PhD Dissertation

International Doctorate School in Information and Communication Technologies

University of Trento

Department of Information Engineering and Computer Science

Modeling and Reasoning about Contextual Requirements: Goal-based Framework

RAIAN ALI

Advisor:

Prof. Paolo Giorgini

Università degli Studi di Trento

March 2010

Abstract

Most of requirements engineering (RE) research ignores, or presumes a uniform nature of, the context in which the system operates. This assumption is no longer valid in emerging computing paradigms, such as Ambient, Pervasive and Ubiquitous Computing, where it is essential to monitor and adapt to an inherently varying context. There is a strong relationship between requirements and context. Context might be considered to determine the set of requirements relevant to a system, to derive the alternatives the system can adopt to reach these requirements, and to assess the quality of each alternative. A RE framework that explicitly captures and analyzes this relationship is still missing.

Before influencing the behavior of software, context influences the behavior of users. It influences users' goals and their choices to reach these goals. Capturing this latest influence is essential for a software developed to meet users' requirements in different contexts. In this thesis, we propose a goal-oriented RE modeling and reasoning framework for systems operating in and reflecting varying contexts. To this end, we develop a conceptual modeling language, the *contextual goal model*, that captures the relationship between context and requirements at the goal level and provides constructs to analyze context. We develop a set of reasoning mechanisms to analyze contextual goal models addressing various problems: the consistency of context, the derivation of requirements in different contexts, the detection of conflicts between requirements happening as a consequence of changes in the context they lead to, and the derivation of a set of requirements that leads to a system developed with minimum costs and operable in all of the analyzed contexts. We develop a formal framework, CASE tool, and methodological process to assist analysts in using our modeling and reasoning RE framework.

We evaluate our proposed RE framework by applying it on two systems: smart home for patient with dementia and museum-guide mobile information system. Our contribution to RE research is a RE framework specialized for emerging computing paradigms that weave together software and context. It allows us to overcome the limitation of existing RE frameworks that ignore, or presume a uniform nature of, the context in which the system operates.

Keywords *[Requirements Engineering, Context Analysis, Goal Modeling, Reasoning about Contextual Requirements]*

Acknowledgments

I would like to thank my advisor Paolo Giorgini who has offered me a continuous and excellent guidance during my PhD study. I would like also to thank all members of the Tropos research group at the University of Trento for all their feedback and for the very useful scientific discussions and in particular John Mylopoulos and my colleagues Fabiano Dalpiaz, Sameh Abdel-Naby, Alberto Griggio, Andres Franzen, Amit Chopra, Vitor Souza, Yudis Asnar, and Nauman Qureshi. Special thanks are due to Bashar Nuseibeh for all the support he gave to me when I was a visiting research student at the Open University. Special thanks are also due to my PhD examination committee (Jaelson Castor, Oscar Pastor, and Yijun Yu) for the very useful comments about my thesis. I am also thankful for the academic staff of my bachelors study at the University of Latakia and in particular to Nasser A. Nasser for encouraging my interest in research. I also thank visiting professors at the University of Trento who gave useful feedbacks about my research (Eric Yu, Haris Mouratidis, Andrea Omicini, and Alex Borgida).

Contents

Chapter 1

Introduction

The recent advances in computing and communication technologies such as sensor systems, positioning systems, mobile devices, and so on, have led to the emergence of computing paradigms such as ambient intelligence, pervasive, ubiquitous and context-aware computing. A core element of these emerging paradigms is context. The notion of context has been used in different ways and in different computer science disciplines such as artificial intelligence, natural language processing, and recently mobile and ubiquitous computing. Since requirements are user needs that could vary according to the environment the users are in, we refer by context to the reification of such environment [1].

Context has a strong influence on system requirements: it can be a factor in deciding requirements to meet, choosing among possible ways to meet the requirements, and assessing the quality of each of these ways. On the other hand, the system itself may cause changes in the context as a result of meeting its requirements. However, in spite of the mutual influence between context and requirements, context is either ignored or presumed uniform in RE literature and is considered mainly during the later stages of software development (Architecture [2], Runtime Adaptation [3], HCI [4], Services [5]). A RE framework for modeling and analyzing the requirements of systems reflecting their context is still missing.

Requirements can be contextual. Weaving together computing and human‘s living environment implies the awareness of the varying states of such environment and how variations in the environment states influence the computing system. We advocate that the environment influences users' decisions first: it influences what they require, the possible ways they achieve their requirements through, and the quality of each way. The designed software needs to reflect users adaptation to their environment as a preliminary step to derive what functionalities to execute. For example, in a smart email sys-

tem, the network bandwidth and the computing device capabilities influence the system decision of downloading high/low quality image or video attachments. However, there is even an earlier adaptation to the environment that is user adaptation. For example, usually users do not need to see emails with attachments when they are driving, in a meeting rooms, or using mobile phones with small screens. The system has to reflect user intentions to read emails first before deciding whether to download high quality or compressed attachments.

Context has been used in different ways in computer science literature. It has been used in natural language processing, communication, and image processing to name few. Recently, the notion of context has become common in emerging computing paradigms such as Pervasive, Ubiquitous, Ambient Computing. Although this notion tends to have a common sense in this community, there are several definitions of it. The definition we are going to develop in this thesis is close to the one given in [1] where context is considered as the reification of the environment. The environment is defined as whatever in the world provides a surrounding in which the system is supposed to operate. This definition emphasizes the *world* that is broadly accepted as a core element in requirements engineering literature [6, 7, 8].

Software systems are means to reach user requirements and they are not requirements per se [6, 9, 10]. One important source of requirements is the stakeholders' 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 to *"involve the patient in social activities"* (G_1) whenever *"the patient is feeling bored and it has been long time since his last social activity"* (C_1). The caregiver can satisfy goal G_1 both by *"taking the patient for a trip in the city"* ($G_{1.1}$) or *"asking a relative or an old friend of the patient to come"* ($G_{1.2}$). Goal $G_{1.1}$ is adoptable only if *"the city is not crowded"* ($C_{1.1}$), since people with dementia usually get anxious in crowded places. Goal $G_{1.2}$ is adoptable only if *"the patient has relatives or friends that can come"* ($C_{1.2}$). The requirements model of a software that supports 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: (i) if $C_1 \wedge C_{1.1}$ then $G_{1.1}$, and (ii) if $C_1 \wedge C_{1.2}$ then $G_{1.2}$.

Goal models have been proposed in the RE literature (i^* [11], Tropos [12, 13], and KAOS [14]) to represent high level goals and possible alternatives to satisfy them besides the quality of each alternative through the notion of softgoals. Moreover, goal models have been used to represent the rationale of both humans and software systems [15], and they have been shown helpful for adaptive systems engineering in particular [16, 17].

This thesis addresses the problem of modeling and analyzing requirements for systems operating in and reflecting varying contexts. It proposes a goal-based modeling language to express the requirements at an early level, the intentional level, that makes explicit the *why* of requirements [14]. Besides the why of requirements, capturing the relationship between context and goals makes explicit the *where/when* of requirements as well. The requirements are complete if they sufficiently establish the goals they are refining [18, 19]. Considering systems reflecting their varying contexts, the requirements are complete if they sufficiently establish their goals in those varying contexts. In other words, we make explicit the influence of the world variability on goals and goal refinements.

1.1 Research Baseline

The work in this thesis is, mainly, based on the following baselines:

- *Context is a partial state of the world.* The world is whatever provides a surrounding in which an actor, possibly the system, lives. The relevance of the parts of the world depends on the decisions and actions an actor takes.

  Example 1. Lets us consider a smart home system (actor) that is responsible of managing home on behalf of habitants. One of the objectives of a smart home is the refreshment of air inside home. The decision about the activation of this objective depends on the context. In a context like "humidity inside home is high or windows have not been opened for long time", the system may decide to refresh the air. Such context describes, partially, the world. The world elements relevant to this decision are the humidity level and the windows state. The system may open the windows in order to refresh air or may, alternatively, turn a ventilator on. The decision about the adoptable alternative depends on the context in turn. In a context like "it is sunny and not windy outside" the system may open the windows, otherwise it may turn the ventilator on. The action of opening the windows may change the context in turn. The windows become opened and the light level may become high.

- *Requirements can be contextual.* Context can influence the user requirements, the alternatives to meet them, and the quality of each alternative. This implies that the software has to reflect user rationale concerning what requirements to meet, and how to meet them, in

different contexts. This reflection is essential for a valid software that meets user expectations in different contexts.

Example 2. Let us consider a promotion staff in a shopping mall who is responsible of promoting a product by giving free samples of it to customers. The staff activates the promotion of the product to a customer in a context like "the customer looks interested in the product". A staff may give a sample in two ways: giving a physical sample directly to the customer, or giving a code to the customer to use it for getting the sample through a dedicated machine in the mall. Context may help to decide which of these two alternatives is adoptable. The first alternative is adoptable in a context like "customer has no experience in using automated machines and staff still has samples of the product". The second alternative is adoptable in a context like "user knows how to use automated machines or there is no free sample with staff anymore". The quality of each of these two alternatives depends on the context. The second alternative, from the perspective of a quality measure like "customer comfort", is good if a context like "one of the dedicated machines is free and close enough to the customer's location" holds, otherwise it is a poor quality alternative. A system designed to autonomously manage the promotion process should reflect this rationale of a promotion staff. It may, for example, direct staff to give samples to customers or may send code to customers through SMS depending on the context.

- *Variants are the cornerstone for adaptability.* A system with one variant can not be adaptable. From the perspective of requirements, adaptability is a selection between variants to meet user requirements. Adaptability to context is the selection of variants that fit to context.

  Example 3. Let us consider a museum guide system designed to assist visitors during their visit and ensure, at the same time, the adherence of museum rules. One of the requirements of the system is to keep people far enough from the pieces of art and prevent them of touching these pieces. If the system has only one way to meet this requirements, such as frequent reminder through public speakers, then the consideration of context has little sense. It may, at the most, influence the activation of the reminder that itself implies variability, i.e. to activate or not to activate. The system may have another variant to meet this requirement such as the light based alarm. The selection between these two variants can be based on the context. For example,

the main requirement is activated in a context like "there is a visitor who touched or he is so closed to a piece of art". The light-based alarm can be adopted in a context like "there is an audio presentation in the room of the piece of art", while the voice-based alarm can be used in the other contexts.

- *Software is a means to reach humans' goals.* The seminal works by [18, 14, 10], emphasized the importance of knowing the goals behind software systems and answering the question "why do we need a software". In other words, software is not a goal by itself, rather it is a means for reaching some human' goals. Humans' goals and the way through which they reach these goals can be influenced by the context. Consequently, software is not necessarily uniform in reaching human goals but can be context-dependent. Software has to reflect the rational of human in deciding what goals to reach and which way to reach them in different contexts. This reflection is preliminary to derive a useful functionalities to execute.

  Example 4. Let us consider a caregiver for people with dementia problems. One of the goals of a caregiver is to maintain the safety of the patient in case of anxiety attacks. This goal of a cargiver becomes active when the context "symptoms of anxiety are clear" holds. The caregiver may have two alternatives to deal with such anxiety and keep the patient safe. The first one is by calming the patient down that is adoptable when the context "anxiety seems to be moderate or the patient dementia stage is still basic" holds. The second one is by preventing the patient from getting out by closing entrances and calling other colleagues to help in giving medication. This second alternative is adoptable when the context "the patient is extremely anxious and his dementia disease is severe" holds. A smart home, as a software system, is a means to reach the caregiver goals. The smart home has to reflect the goals of caregiver and the way through which he reaches these goals in different contexts. Smart home may calm the patient down, in the correspondent context, by playing relaxation music. Alternatively, it may actuate motors to close and lock the entrances, and issue a public speaker message to call caregivers to give medication.

- *Context needs a systematic way to be correctly specified.* The specification of context means the specification of the way an actor, possibly the system, can verify if it holds. Context is a state of the world that is the case. A state of the world may not be visible per se but could

be an abstraction of visible facts in the world. Discovering these facts and the way through which they are composed to verify the truth of a state of the world may be complex. This complexity necessitates a systematic way to specify a context correctly.

Example 5. A context like C_1="the customer is interested in the product" is relevant when deciding if a product has to be promoted to a customer. However, this context is not visible per se, rather it is derived from visible facts in the environment. Facts that indicate this context can be multiple and the logical composition of these facts to derive the truth value of such context can also be complex. One of the different ways to derive C_1 can be through the context $C_{1.1}$= "the customer's behavior indicates interest in the product", or the context $C_{1.2}$="his purchase history indicates interest in the product". In turn $C_{1.1}$ is not monitorable per se but can be derived from some facts such as "he is watching the product for long time" and "he does not watch much other products that are of the same category".

1.2 Research Question

In several emerging computing paradigms, such as Pervasive, Ubiquitous, and Ambient computing, requirements are not absolute but can be context-dependent. Context may influence requirements in different ways. It may activate a set of requirements, make adoptable a set of alternatives to meet the activated requirements, and influence the quality of each of such alternatives. On the other hand, the system may cause changes in the context as a consequence of meeting its requirements. Therefore the influence between context and software is mutual.

A requirements engineering framework for systems operating in and reflecting varying contexts is missing. In most requirements engineering research, context is either ignored or presumed uniform. The assumption of uniform context is obviously not valid when we weave software with a human's living context. Systems are now expected to reflect varying contexts they may operate in. This thesis aims to develop a requirements engineering framework that weaves requirements with context. It, mainly, addresses the following questions:

- Context may influence the requirements and be influenced by the functionalities that the system may follow to meet its requirements. This raises the question:

  - *How can we capture the relationship between requirements and context?. In other words, how can we model the mutual influence between requirements and context?.*

- The specification of context is not always a trivial task. Some complex, or unclear, contexts may require a systematic way to reach their specification correctly. This raises another research question that this thesis deals with:

  - *How can we systematically identify the way an actor (possibly the system) can judge if a context holds? In other words, how can we materialize context by discovering visible facts in the system environment that the context reifies?*

- A contextual-requirements model may include complex specification of (several) context(s). Such specification may be a subject of modeling errors that make it inconsistent. This raises another main question of this thesis:

  - *How can we check the consistency of a context specification?*

- The system, for meeting its requirements, may carry out a set functionalities that lead to changes in the context. The changes of the context may lead to conflicts among the different functionalities that the system may execute to reach its set of requirements. This raises the question:

  - *How can we check a designed contextual requirements model to detect and assess the severity of conflicts, manifested via changes on the context, between different system functionalities?*

- Systems, at runtime, need to monitor their context and adapt themselves to its current state. This means that the system has to choose amongst variants to meet its requirements. In other words, the system needs to make a meta-computation to derive useful functionalities to execute in each different context. This rises the question:

  - *How can we specify a way to adapt the requirements model to multiple contexts and allow the system to autonomously do this adaptation?*

- Having numerous variants is desirable for reaching higher degree of flexibility that allows, amongst other things, the system to adapt to multiple contexts. From the other side, supporting a large number of variants can be problematic in terms of extra time and costs needed to establish them. In other words, there is often a tradeoff between flexibility and feasibility. This raises the question:

  - *how can we reason about a designed contextual requirements model, that contains a large number of variants, to elicit those variants allowing the system to meet its requirements in all considered contexts with minimum development costs?*

1.3 Contribution of the Thesis

In literature, there is a gap between variability of context and requirements. In this thesis, we try to reduce this gap and allow for expressing of and reasoning about requirements for varying contexts. More precisely, the contribution of this thesis is:

- Conceptual modeling language for representing contextual requirements: we propose the *contextual goal model* to capture context-based variability in requirements at the goal level. We identify a set of variation points at Tropos goal model where context may intervene to decide the adoptable alternatives for satisfying a goal. We also propose a set of modeling constructs to analyze context. This analysis helps to discover the information that the system needs to capture from its environment and how this information is composed to judge if an analyzed context holds.

- Reasoning techniques: we develop two reasoning techniques concerning the requirements derivation for varying contexts. The first one concerns the automatic derivation of goal model variants that reflect context and user priorities at runtime. The second is for processing a contextual goal model to extract the variants leading to a system developed with minimal costs and operable in all considered contexts. This reasoning is useful at design time to decide the core requirements the system has to meet when there are budget or timing constraints. We develop another two automated reasoning techniques to detect design errors in a contextual goal model. More concretely, we check it for the consistency of context specification and for the conflicts caused by changes in the context the satisfaction of goals leads to.

- Logical framework and CASE tool: we formalize our proposed contextual model and use off-the-shelf reasoners as a part of implementing our reasoning mechanisms. We develop a prototype tool, called RE-Context, that incorporates the formalism and the reasoning mechanisms. We propose a methodological process to construct and reason about contextual goal models. We also apply our process on two contextual goals models of two context-dependent systems and report the obtained results.

1.4 Structure of the Thesis

The thesis is structured as follows:

- **Chapter 2** presents an overview of the state of the art of the research question of this thesis.

- **Chapter 3** proposes the *contextual goal model* to express requirements for varying contexts. The chapter starts with the main principles of weaving together requirements and context. It overviews Tropos goal modeling and its main constructs. We then motivate the integration between context and goals. A set of variation points of Tropos goal model is defined. The variation points are classified based on the semantic of the relation between context and goal model on each of them. A set of modeling constructs to analyze context is proposed. These constructs are motivated by the need to analyze contexts and discover ways to verify them based on visible facts in the environment. An analogy between context analysis and goal analysis is discussed. A general discussion about our proposed model is presented.

- **Chapter 4** proposes a set of reasoning techniques about our proposed contextual goal model. The first category of reasoning techniques addresses the problem of detecting modeling errors in a designed contextual goal model. It checks the context specified for each variant of a contextual goal model to decide if it is consistent. We give a classification and semantics for the inconsistency of contexts. We also check each variant of the goal model for conflicting changes on the context caused by that variant executable tasks. The other category of reasoning techniques addresses the problem of deriving the variants of a contextual goal model at design time and runtime. For the design time, we develop a reasoning technique to decide the variants the system-to-be has to support to enable it of reaching stakeholder's goals in all considered

contexts with minimum development costs. The other reasoning technique addresses the runtime derivation of the contextual goal model variants that fit to a monitored context and user prioritization.

- **Chapter 5** proposes a logical framework and CASE tool to support our reasoning mechanisms. We translate our contextual goal model into Datalog that enables us to derive all the alternatives for the satisfaction of a goal. We translate context specification into a Boolean formula as a step to use SAT solver for verifying context consistency. A prototype CASE tool, called RE-Context, is developed to support the four kinds of reasoning that are proposed in Chapter 4. A methodological process that is followed to construct and reason about contextual goal models is also proposed.

- **Chapter 6** describes the results we got by applying our framework on two case studies. We explain the application of our proposed framework on two systems: a smart home for patients with dementia problems, and a museum-guide mobile information system. We report the results we got in terms of developed models, reasoning results, qualitative feedbacks reported by software engineers, performance analysis of our developed CASE tool.

- **Chapter 7** concludes the thesis and presents a set of problems as directions for a future work. We discuss the generality of our approach applying its principles on another variability model that is Feature Model. We also present an initial work towards an integrated RE framework for contextual requirements. We outline a set of problems concerning the RE of system operating in and reflecting varying contexts such as: viewpoints in context specification, contextual security requirements, optimizing monitoring requirements, and lifelong adaption to context.

1.5 Published Work

- Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. *Location-based Software Modeling and Analysis: Tropos-based Approach.* In the Proceedings of the 27th International Conference on Conceptual Modeling (ER 08), Springer LNCS 5231, Pages 169-182. Barcelona, Spain. October 20-23, 2008.

- Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. *Location-based Variability for Mobile Information Systems.* In the Proceedings Of the 20th International Conference on Advanced Information Systems Engineering (CAiSE 08), Springer LNCS 5074, Pages 575-578. Montpellier, France. June 16-17, 2008.

- Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. *A Goal Modeling Framework for Self-Contextualizable Software*. In the Proceedings of the 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD09), Springer LNBIP 29. Pages 326-338. Amsterdam, The Netherlands, 8-9 June, 2009.

- Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. *Modeling and Analyzing Location-based Requirements: Goal-oriented Approach*. International Journal of Computer Science and Software Technology (IJCSST). Vol. 2, Nr. 2, July-December (2009).

- Raian Ali, Fabiano Dalpiaz and Paolo Giorgini. *Modeling and Analyzing Variability for Mobile Information Systems*. In proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2008), Springer LNCS 5073, Pages 291-306. Perugia, Italy, June 30th - July 3rd, 2008.

- Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. *Goal-based Self-Contextualization*. In the Forum of the 21st International Conference on Advanced Information Systems (CAiSE 09 - Forum). CEUR-WS Vol-453, Pages 37-42 .Amsterdam, the Netherlands, 8-12 June, 2009.

- Raian Ali, Yijun Yu, Ruzanna Chitchyan, Armstrong Nhlabatsi, and Paolo Giorgini. *Towards a Unified Framework for Contextual Variability in Requirements*. In the proceedings of the 3rd International Workshop on Software Product Management (IWSPM09), In conjunction with 17th IEEE International Requirements Engineering Conference (RE09). Atlanta, Georgia, USA. September 1, 2009.

- Raian Ali, Ruzanna Chitchyan, and Paolo Giorgini. *Context for Goal-level Product Line Derivation*. In Proceedings of 3rd International Workshop on Dynamic Software Product Lines (DSPL09) co-located with the 13th International Software Product Line Conference (SPLC09). San Francisco, California, USA.August 24 - 28, 2009.

- Raian Ali, Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, John Mylopoulos, and Vitor E. Silva Souza. *The Evolution of Tropos: Contexts, Commitments and Adaptivity*. In the proceedings of the 4th International i* Workshop, co-located with the 22nd International Conference on Advanced Information Systems Engineering (CAiSE 10). Hammamet, Tunisia, 07-08 June, 2010.

- Fabiano Dalpiaz, Raian Ali, Yudistira Asnar, Volha Bryl, Paolo Giorgini. *Applying Tropos to Socio-Technical System Design and Runtime Configuration*. In the Proceedings of the 9th WOA workshop, From Objects to Agents (Dagli Oggetti Agli Agenti), ISBN 978-88-6122. Palermo, Italy, 17-18 Novembre 2008.

- Raian Ali, Sameh Abdel-Naby, Antonio Mana, Antonio Munoz and Paolo Giorgini. *Agent Oriented AmI Engineering*. In the Proceedings of the Ambient Intelligence Developments Conference (AmI.D07), Springer ISBN: 978-2-287-78543-6, Pages 166-179. Sophia Antipolis, French Riviera, France, September 17-19, 2007.

- Sameh Abdel-Naby, Paolo Giorgini, and Raian Ali. *Towards Integrating Agents with Objects Tracing Systems in AmI*. In the 5th European Workshop on Multi-Agent Systems (EUMAS'07), Hammamet, Tunisia. December 13-14, 2007.

Chapter 2

State of the Art

2.1 Context-Awareness

The advances in information and communication technology has led to new computing paradigms such as Ambient Intelligence [20, 21], Pervasive Computing [22, 23], Ubiquitous Computing [24, 25], Context-Aware systems [26, 27], and so on. The target computing systems in these paradigms are able to reach users' requirements transparently and avoid them the explicit interaction with computers. In other words, the main motivation in these paradigms is decoupling user from computing devices [28]. Users should get their needs met without an explicit request to a computing system. These computing paradigms are the application area for which our modeling and reasoning framework is designed. A core element in these paradigms is *context*. In this section, we review the literature and discuss several definitions of context, show different classification of contextual information, and finally we outline different approaches concerning context modeling.

2.1.1 Context definition

Context has been defined in different computer science disciplines such as artificial intelligence (for a survey see [29]). It has also been defined in the literature of context-aware computing in variant ways. Here we outline several definitions of context and then we make an observation about the common characteristics of context.

- One of the earliest work that explicitly considered context and context aware computing is the work by Schilit et al. [30, 31]. In that work context has been considered in terms of attributes of elements in the physical environment. The work emphasizes that context is more

than a physical location that the user is in. It may also includes different changing factors such as lighting, noise level, network connectivity, communication costs, communication bandwidth. Context can also refer to the social environment of a user such as whether he is with his manager or with a co-worker, and so on. However, in this work, context is defined by examples. Although it gives intuition to what context is, this approach makes it difficult to judge precisely if an information is a part of context.

- Dey [32], observed that defining context by examples or by giving synonyms to it (such as situation or environment) does not help to decide what context is. The work also made another observation about several other definitions that defined context based of the kind of the targeted applications. Dey gives a definition of context that refers to the world relevant to the interaction between humans and computing: "*Context is any information that can be used to characterize the situation of an entity.*" An entity is defined as "*a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*". In [33], Dey et.al, extend this definition by giving examples about the dimensions of context: *"context is typically the location, identity, and state of people, groups, and computational and physical objects"*.

  This definition emphasizes that context is what is relevant to an interaction between the users and an application. However, these is also another limitation for context. Context may influence the users' needs before influencing their interaction with applications. The application has to reflect the influence of context on users in order to derive useful execution course. In [32], Dey also gives a definition of context-awareness as following: *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*.

- In [34], Chen observed that context has two main aspects that are the characteristics of the surrounding of an application that strongly influence the behavior of that application, and the surrounding that is relevant but not critical. Context is defined as *"the set of environmental states and settings that either determines an applications behavior or in which an application event occurs and is interesting to the user"*. Based on this, the authors classifies context awareness computing in two kinds:

  - Active context awareness: where the application monitors context and adapts its behavior when a change in the context occurs.
  - Passive context awareness: where the application monitors changes in the context and presents them to the user like presenting an information about a close restaurant.

- Finkelstien et al. [1], give another definition of the context. The definition emphasizes the uncontrollable nature of the context. Context is defined as the reification of the environment. The environment is defined as whatever in the world provides a surrounding in which the machine is supposed to operate. Alternative definition of the environment is whatever over which we have no control. The context in this work tends to be a technical context such as the network bandwidth, display characteristics, the spatia-relations between devices, and so on.

- Yau et al [35, 36], define context as "any detectable attribute of a device, its interaction with external devices, and/or its surrounding environment". The context considered here is that describing a computing device such as the battery level, the geographical location, the history of interaction of this device with other and so on. The same work define context-awareness as "the ability of a device to detect its current contexts and changes in any contextual data". Again, the adaptability to context is limited to the one that concerns the technical environment of a computing device.

2.1.2 Context dimensions

Contextual information can be classified into different dimensions. Krogstie et el. [37, 38], observed that mobility is strongly related to changes in the context. The changes in the context could potentially affect the objectives and the tasks of mobile information systems. In the same works, Krogstie et al. classify context information as follows:

- The spatio-temporal context: it describes aspects related to time and space. It contains attributes like time, location, direction, speed and track.

- The environment context: it captures the entities that surround the user, for example, physical objects, services, temperature, light, humidity, and noise.

- The personal context: it describes the user state. It consists of the physiological and the mental contexts. The physiological context may contain information like pulse, blood pressure and weight, as well as personal abilities and preferences, while the mental context may include elements such as mood, expertise, anger, and stress.

- The task context: it describes what the user is doing. The task context may be described with explicit goals or the tasks and task breakdown structures.

- The social context: it describes the social aspects of the user context. It may, for instance, contain information about friends, neighbors, coworkers and relatives. The role that the user plays is an important aspect of the social context. A role may describe the user's status in this role and the tasks that the user may perform. The term social mobility refers to the ways in which individuals can move across different social contexts and social roles and still be supported by technology and services.

- The information context: it describes the information space that is available at a given time.

In [39], Henricksen et.al, context has been divided based on changeability into two main categories:

- Static context: it refers to a property that is fixed regarding one entity, like for example: device type.

- Dynamic context: that refers to the context that changes over time such as wheatear, temperature, user mood and so on. The dynamic context is further subdivided into three categories:

  - Sensed context: it is the context that can be captured by sensors such as location, temperature degree, humidity level, blood pressure, and so on.
  - Derived context: it is the context that can be derived using a derivation function. An example of the derived context is the "Is Located Near" relationship. An object is located near to another one is derived from two sensed location contexts which are the coordination parameters.
  - Profile context: this context covers information supplied by users. For example, user name, organization, and the name of people user is working with and so on.

Schmidt et al. [40, 41], uses 3-D model to view context, the dimensions of this model:

1. Self: e.g. device state, physiological, cognitive
2. Environment: the physical and the social environment.
3. Activity: the behavior and task being done.

The work also provides another hierarchial categorization of context. The two root categories are:

- Human factors: these factors are related to a human as individuals, as social environment, and as activities they are doing.
- Physical environment factors: these factors concern the physical conditions, infrastructure, location of the system. Conditions, for example, are further classified into dimensions like: light, pressure, accelerations, audio, temperature, humidity and so on. The dimension light is in turn characterized by dimensions such as level, flickering, color, wave length, and so on.

In his PhD thesis [42], Shilit has classified context information into:

- Physical objects: the main entity that context information are centered around is a physical object. It can be a human or an artefact. Examples of this kind are: mobile phone, windows, furniture, user and so on.
- Conditions and states: some objects have dynamic state, such as users who might be busy, sleeping, driving and so on, and mobile phone which may be off, on, busy, and so on. Some other objects has a static state and does not make part of a variable context such as furniture. Obviously the decision between dynamic and static states is to a large extent a design decision.
- Spatial relations: each object occupies a space and exists in a location. The location may be presented as coordinates in a system or presence in a containing place. For example, two objects may have spatial relation such as “close to”, “on”, “inside” and so on.
- Phenomena: it is an emerging state of the world that makes part of the context. Such a state is not attached to a specific object. For example, “noisy”, “financial crisis”, and so on.

- Ways and means: it concerns the ways to achieve tasks. For example, the ways through which a user can get instructions about the use of an information terminal in an airport, or the ways user can make the check-in through.

- Customizations: it is user, or a user group, given commands or preferences. For example, "do not allow calls during meeting", "turn the mobile phone to silent during meeting" and so on.

Zimmermann et al. [43], classified contextual information into:

- Individuality context: it concerns properties and attributes describing an entity. An entity can be natural, human, group, and artificial.

- Activity context: it concerns the tasks an entity is involved in.

- Spatio-temporal context: it refers to the time and the location of an entity.

- Relations context: concerns any possible relations an entity may establish with other entities.

We emphasize that the definition of context and its dimensions depend on the application area. Taking business process modeling and execution as another domain that addressed contextualization, the considered context is different from that of mobile, pervasive, context-aware computing. For example, Rosemann et al. [44, 45], classify context with regards to its relation to a business process into four kinds:

- Immediate context: this context includes the elements of the organization that are beyond the pure control flow and facilitate the execution of a process directly. For example, the kind of data required, the source of data, the organizations resources that are in charge with the next activity and so on.

- Internal context: refers to the elements of the organization that have indirect influence on the business process. For example, norms and values, concerns and interests, strategy, structure and culture and so on.

- External context: it refers to elements that are in a system wider than the organization and not under the control of the organization and that have influence on the business process. For example, it includes categories of context elements related to suppliers, competitors, investors and customers.

- Environment context: this context refers to those elements in the world that are outside the business network but still have influence on it. For example, the weather factor may influence the organization activities as the call volume may need to be increased during the storm season. Time of the day, week, month, and year may also influence the activities, for example, on Saturday many people make shopping, so more products has to be available in the store.

From the above definitions, we could outline the following characteristics of context:

- *Context is a surrounding for an entity*: in all the definition, we see that context is seen as a surrounding of an actor. This actor could be a person or a machine i.e., the context-aware system. This means that a context is seen from some perspective and not given as an absolute concept.

- *Context is not fully controllable*: context refers to what is in the world and that is not under the full control of an actor. The actor is supposed to reflect to the uncontrollable changes in context, as possible. The elements in the world that are fully controlled by an entity are not part of the context but part of the actor itself.

- *Context influences certain decisions and actions*: context has influence the decisions and the actions an actor needs to take. The actor that observes context will reflect the changes in that context. Those parts of the world that do not influence the choices or the actions of an actor, are not part of the context and do not need to be observed.

- *Context has a subjective nature*: following the previous point, what makes an element in the world of an actor part of the context is the decision that actor needs to take.

- *Context has a volatile nature*: the elements in the world that are uniform are not part of the context. An actor does not need to observe them when it takes a decision. One main idea of context-awareness is the variable nature of the world an actor lives in and that influence that actor's decisions.

2.1.3 Context Modeling

Various modeling approaches for representing context have been proposed (for a survey see [46, 47]). Henricksen et al. [39], propose an object-oriented

representation of context information in pervasive computing. They extend Entity-Relationship diagrams with constructs to allow for more expressiveness in capturing context information. The association entities-entities and entities-attributes are classified into static and dynamic. The dynamic association is classified into sensed, profiled, and derived. An example of a static association is that between a mobile device and its type. An example of sensed association is that between a person and its location (location is sensed). An example of profiled association is that between a person and his supervisor or his device. An example of derived association is that the spatio-relation between two entities that can be derived from their locations.

The associations between entities-entities and entities-attributes are classified according to their structure into four kinds. The simple association that happens when an entity, which owns an association, does not participate in it more than once. For example, the device has one device type. An association is composite if it is not simple and this kind includes the other three kinds of associations. The collection association means that an entity can be associated simultaneously with multiple attributes values and/or other entities. For example, a person may be sitting with more than one friend at the same time. The alternative association refers to an association between an entity and a number of other entities/attirbutes, where at one moment there is only one active association. For example, the relation between a TV and channel. The temporal association is the same as alternative associations but attached to a time interval. For example, a person may be doing activity within a certain time. In [48], Henricksen et al. use the Object-Role Model (ORM [49]) instead of ER model for the purpose of efficient formalization and transformation to logical models.

Another approach in modeling context uses ontologies. CONON is an example of this category of context modeling techniques proposed by Wang et al. [50, 51]. The main idea of this ontological approach is by proposing a general ontology (upper ontology) for context and then specializing it for a specific application (domain-specific ontology). The upper ontology captures concepts that are common in a large variety of context-aware applications such as person, activity, location, and computing entities. This ontology is then specialized for each application. For example if we talk about smart home for health care, then the persons could be the patient and the caregiver, the location can be "bed", "kitchen", "garden" and so on, the devices can be the "DVD player", "TV", "cellular phone", and the activity can be "sleeping", "watching TV", "taking medicine" and so on. In this work, the Web Ontology Language (OWL [52]) is used as a formalization that allows for automated reasoning. To this end, reasoning rules has to be defined. For example, to conclude a situation like "patient is taking a shower" some predicates has

to hold such as “patient is in the bathroom”, “bathroom doors are closed”, “water heater is on”, and so on.

Another example of ontology-based context modeling is CoBrA [53]. It provides a set of classes common in pervasive computing such as the place, agent, activity, location. These classes are refined into subclasses as well. For example the place is specialized into campus, building, restroom, room and so on and the class agent that can be a person, a software agent, a role and so on. The ontology proposes also a set of attributes related to each class. For example an agent could have a name, address, intends to perform and so on. An activity could have attributes such as starting time, end time, location, ..etc. Similar to CONON, CoBrA is formalized and reasoned about using OWL.

Several other approaches for modeling context exists. CMP is a UML Context Modeling Profile for Mobile Distributed Systems proposed by Simons in [54]. It provides a UML profile tailored for contextual information. To this end it proposes specialized classes that help for better capturing and visualizing of contextual information. It uses Object Constraint Language (OCL [55]) to assert the right usage of the proposed constructs and associations. In [56], a context is considered as atomic situation and the ontology proposed is for hierarchial representation of a compound/complex situations. Description logic (DL) reasoner over OWL are used for different purposes such as consistency checking of an ontology and knowledge inferring.

2.2 Goal-oriented Requirements Engineering

In this section, we briefly discuss goal-oriented requirements engineering (GORE) research. We outline the main concepts and key ideas behind GORE. We list the main motivation of GORE and we finally review a variety of ongoing research in GORE.

2.2.1 Main concepts

Although different approaches in requirements engineering used goal modeling in different ways, a set of concepts is almost core in all of them. Here we review a list of the main GORE concept:

The concept of goal: the main concept in GORE is goals. Goals have been defined in multiple places in the literature of requirements engineering. In [57], Darimont and Lamsweerde define goal as “an objective the composite system should meet”. Anton [58], defines goals as “the high level objectives of the business, organization, or system. They capture why the system is

needed and guide decisions at various levels within the enterprise". Plihon et al. [59], consider goal as a synonym of intention and define it as "a future system state or behavior to avoid, maintain, attain, cease, etc.". Rolland et al. [60], refines the last definition into "a goal is something that some stakeholder hopes to achieve in the future.". Bresciani et al. [12], define goal as "representation of actors' strategic interests" where actor is defined as "an entity that has strategic goals and intentionality within the system or the organizational settings". Pohl et al. [61], define goals as "the objectives an actor wants to achieve when requesting a certain service", while an actor or user type "represents a set of users that have some common characteristics with respects of why and how they use the system".

The concept of softgoals: a category of goals is called softgoals and defined by Mylopoulos et al. [15] as "goals that do not have a clear-cut criterion for their satisfaction. We will say that softgoals are satisficed when there is sufficient positive and little negative evidence for this claim, and that they are unsatisficeable when there is sufficient negative evidence and little positive support for their satisficeability". Often, softgoals are treated as quality measures to differentiate between multiple strategies to reach goals. They also can be used to express user preferences over those strategies as done in [62].

The owners of goals: goals does not exist as separate entities. They are owned by an (intentional) entity. Dardenne et al. [14], classify goals with regard to the ownership dimension into: private that an individual agent owns, and system that are the goals of a system as whole. Similarly, Pohl et al. [61], classify goals into business goals and personal goals. There is no explicit classification of goals based on ownership in Tropos GORE methodology [12]. Tropos projects a system/organization as a set of interdependent actors and associate goals to them. There is no explicit notion of a whole system or individual actors goals.

The source of goals: identifying or eliciting goals can be done through ways such as scenarios, use cases, interviews, corporate mission statements, policy statements, an so on [63, 64]. Eliciting goals via scenarios is one of the intensively studied techniques. Scenarios help for identifying goals and the different ways through which they may be implemented or reached [65]. Use cases may be considered as implementation of stakeholders goals [66]. However, as observed by Constantine et al. [67], although use cases can be goal driven, but the focus is on those goals concerning the direct interaction between a user and a system. In other words, there could be goals that are not directly related to that interaction.

The refinement of goals: goals are not executable, they tend to be interests or intentions. Goals may be ultimately reached by a set or alternative

sets of executable processes [12, 14]. The refinement is AND/OR refinement. When we refine a goal through AND-Decomposition into subgoals, this means that reaching all subgoals implies the achievement of the refined one. When we refine a goal into a set of goals through an OR-Decomposition, this means that reaching one of these subgoals is enough to reach the refined goals. The refinement through OR creates a space of alternatives to reach the root analyzed goal, i.e. variability. An anatomy to the elements of a goal statement that can be refined through OR to give a space of alternative is explained in [68].

2.2.2 Main motivations

The main reason for goal-oriented requirements engineering is to explicitly capture the "Why" in requirements. Requirements are, first of all, user's requisites over the world and are not a set of software functionalities. A software is a means to reach user goals. The starting point of software development has to be at the level of stakeholders' strategic goals. The analysis of such goals may indicate a need and possibility for a software that helps to reach such goals. Consequently, goals allow for a rationale for software functionalities and answer the basic question "why do we need a software". In what follows, we list a set of main motivations for goal-oriented requirements engineering [9, 10]:

1. Requirements acquisition: goals guide and give rationale to what a software has to execute. Analyzing goals by asking questions such as, "why", "how", "how else", "how good is", helps to reveal stakeholders' requirements. Moreover, stakeholders become aware of the space of possible solutions to reach their goals instead of committing, in less comprehensive way, to a certain technology without discovering what the other possibilities are.

2. Positioning requirements in the organizational domain: goals allow us to bridge the gap between software and the organization it is intended to operate in. Goal-based requirements analysis helps to derive a software that addresses certain problems in an organization. It overcomes the limitation of other approaches, such as object oriented analysis, that start immediately by studying what the system has to do, but do not justify why it has to do that.

3. Explaining and clarifying requirements: goal models, in general, allow for refinement of high level goals by asking how, and how else, and how good each alternative is. This incremental and iterative process

helps stakeholders to clarify their potentially ambiguous requirements. Moreover, this refinement explains the requirements to stakeholders by answering their questions such as "why do we need to do this", and "how do we reach this" and so on.

4. Managing conflicts: stakeholders are different in their interests and priorities of these interests. This may lead to conflicts about what requirements to reach and how, and how well, to reach them. Even a single stakeholder may have conflicts between a set of requirements. Goals allow for a natural, high level, abstraction that may be exploited to manage those conflicts. For example, achieving a goal may deny the achievement of another. Moreover, a way to reach a goal can be assessed differently by different stakeholders.

5. Tracking the design: goals and the refinement of goals lead to several design options. In other words, the system may implement a subset of these options. Goals can be used to assess if the design or even the final system is correctly specified/developed. Goals provide a criteria to evaluate and track the system at later stages. For example, in autonomic computing, the final system may switch between different execution course to keep user goals satisfied.

6. Avoiding irrelevant requirements: goals provide a criteria for software pertinence. A software functionality is pertinent if it is a part of reaching a goal of stakeholder, otherwise its implementation is not justified.

7. Isolating stable from volatile aspects in requirements: goals tend to be persistent or stable while software, as a way to reach goals, are subject of evolvement or changes. The ways to reach goals may be influenced by the environment in which the system will operate while goals may be stable. In other words, we may have a goal that needs to be reached in different environments while the way to reach it may be different in each of them.

2.2.3 Ongoing research

Besides the well established research on goal modeling such as goal elicitation, goal refinement, obstacle analysis, goals and scenarios, and so on (for a survey see [69]), recent research has addressed new areas where goal modeling can be useful for. In what following, we briefly review these new research directions in goal modeling:

- Security modeling and analysis [70, 71, 72]: this research focuses on the security requirements at a level higher than the software, the organizational level. It provides modeling concepts that stand for high level security requirements and trust requirements between actors (intentional entities) and their implementation. i*/Tropos has been shown a suitable abstraction for an organizational system, especially the early requirements phase where actors depend on each other for goal to be achieved, tasks to be executed, or resources to be furnished. Secure Tropos extends i*/Tropos with concepts to catch security and trust requirements between those actors. While security in traditional i*/Tropos and goal oriented analysis in general is left vague or at most treated as a softgoal; Secure Tropos treats security requirements as a first class entity at this early level of analysis. To this end, Secure Tropos provides modeling concepts for the entitlement between actors and their capabilities. To model the transfer of entitlement and responsibilities between actor and the expectation of an actor about the other actors, Secure Tropos provides the delegation, trust, distrust relations. Secure Tropos engineering framework also provides an automated reasoning. One main purpose of the reasoning is to verify the consistency between security and trust entitlement. Secure Tropos uses Datalog to formalize and reason about its models.

- Risk analysis [73, 74, 75]: goal models have been shown very useful to capture and analyze the objectives of stakeholders and identify and justify the requirements of a software system. However, capturing these objectives and analyzing them has to take into consideration any unexpected or uncertain situation that might be an obstacle when reaching stakeholder objectives. Goal-Risk framework addresses the last question. It extends goal model with constructs to capture uncertain events and their impact on the goals achievement. The negative impacts introduce risks that face the goal achievement. Goal-Risk framework also deals with the treatment of a potential risk. It analyzes the possible treatments in terms of possible actions performed to uncertain events of negative impact. An automated reasoning about the Goal-Risk model is then developed. It allows the analyst of a software system to detect important properties about the designed system regarding what risks are there and how they are treated.

- Automated design [76, 77, 78]: this research provides automated support to explore the space of alternatives during the requirements analysis and design of an information system and find an optimal or good

enough set of delegations between actors and assignment of goals to them. To this end, the research exploits artificial intelligence planning techniques to generate all the possible design structures. Having the alternative design structures generated, they might be then evaluated against the local strategies of system actors. The evaluation part exploits ideas from game theory. The reason for such choice is that each actor has its own strategic interests and try, first of all, to maximize their local utilities. The evaluation of alternative design structure might seek for the one that is good enough for the maximum number of those actors. A prototype tool (P-Tool) is designed to implement the approach and support the analyst in selecting a good enough design-structure alternative.

- Normative i* modeling [79, 80, 81]: when eliciting the requirements, the analyst needs to consider the laws and regulation of the organization where the designed system will operate. This research aims at generating a set of requirements for a new system that are complaint with a given law. To this end, this research develops a conceptual framework that extends goal models with concepts from the legal domain and their relations to goal model concepts. It also provides a systematic process to generate a set of requirements that are compliant with a collection of legal prescriptions and addresses the problem to solve at the same time. Therefore, the proposed Normative i* framework allows for capturing laws at an early level of analysis, the intentional level, where an early impact of laws occurs.

- Goal-oriented testing [82, 83, 84]: the purpose of testing a software system is to be sure that the system that has been developed, or is being developed, is good enough. A main question that concerns the decision about a good enough software is: does the software fulfill its requirements?. This research adopts testing as a software development activity that involves a main source of requirements, the goals. Goal-Oriented Software Testing (GOST), is a testing methodology through which the analyst can derive test cases from requirements captured by goal models, and design specifications. GOST is based on Tropos agent-oriented software engineering (AOSE) methodology and targeted for MAS as implementation technology. Briefly, GOST contributes to existing AOSE methodologies by providing: (i) a testing process model, which complements the development methodology by drawing a connection between goals and test cases and (ii) a systematic way for deriving test cases from goal analysis.

- Commitments and goal models [85, 86]: i* emphasizes the social nature of requirements fulfillment that actors often depend on others to achieve their goals. However, the notion of dependencies in i* is an intentional one, not social. An actor may depend on another for some goal; however that in itself does not amount to any social expectation that the latter will fulfill the goal. More precisely, i* gives no account of which dependencies are legitimate from a social perspective.

  For example, Alice is new to Trento and depends on Barbara, her only acquaintance in the city, to show her around the city; Barbara may also know that Alice depends on her. However, unless Barbara actually commits (via an explicit act of communication) to showing Alice around the city, we (as the society, the observer) do not form the legitimate expectation that Barbara will show Alice around Trento. Until Barbara actually commits to Alice for showing her around the city, the dependency exists only in the minds of the actors. When Barbara commits to Alice, there exists an objective and verifiable relation between her and Alice. This objectivity is a key aspect of being social in systems involving autonomous and heterogeneous actors. Moreover, even if Alice somehow knew that Barbara has the goal of showing her around the city (because, maybe it was mentioned in Barbara's TODO list that Alice happened to read), that does not amount to any legitimate expectation that Barbara will actually do so. Barbara may not act upon her goals, or she may simply change her mind. People and organizations want commitments from each other to guard precisely against such uncertainties: one does not have to care what another's goals are as long as the latter has committed to the former for a state of the world corresponding to that goal. We often mention loose coupling as highly desirable among system components; with commitments, we get the loosest possible coupling among actors.

  The centrality of commitments has long been known to multiagent systems researchers [87]. This research seeks to apply the results there to requirements engineering. It formalizes the notion of how an agent may exploit commitments to fulfill its goals, and vice versa . In doing so, we have completely done away with dependencies as they are conceptualized in i*. Replacing dependencies with commitments is critical to keeping goal-oriented requirements modeling relevant to the engineering of open systems.

2.3 Requirements Driven Variability

2.3.1 Goal-based variability

Goal models allows for analyzing goals and identifying different ways to reach them. This motivates several research questions concerning the acquisition of the space of alternatives to goal satisfaction and the selection amongst them either at design time or at runtime. Starting with goals gives rational to the adjustment of the system in terms of switching between plans guided by stakeholder goals. In other words, the variability is in the actions to reach the goals that are, often, stable. In what follows, we review some research directions that are using goals in the requirements acquisition and design of high-variability software:

- Designing self-adaptive systems [88, 89, 90, 91]: self-adaptive systems is characterized by autonomy in reaching their objective at runtime. This implies a repository of plans the system supports and has the ability to choose between in different environments. A software engineering methodology for building self-adaptive systems is developed. The methodology allows to define and to model the autonomous diagnosis and decision making going to be done at runtime. In order to design self-* systems, one of the major dimensions to consider is the dynamic environment in which the system lives.

  Tropos4AS (Tropos for self-adaptive systems) is in extended version of Tropos proposed in this thread of research. Tropos4AS supports software engineers in developing self-adaptive systems, incorporating principles from goal-oriented requirements engineering, environment modeling and the BDI agent architecture. More precisely, Tropos4AS extends Tropos by constructs to model environment conditions and capture their influence on the behavior of a software agent. It also extends Tropos goal model by constructs that capture the failure in one of its alternatives along with the recovery activities. It uses the alternatives to goal satisfaction to generate alternative configurations that will be implemented in the system-to-be. The main result of Tropos4AS is the systematic transformation of design model into implementation. Therefore, the system at runtime will use the design time models as a reference that guide its behavior.

- Architectures for self-reconfigurable socio-technical systems [92, 93]: A Socio-Technical System (STS) is an interplay of humans, organizations and technical systems. STSs are a type of distributed systems

where a number of autonomous and intentional actors interact in order to achieve the respective objectives. STSs are characterized by dynamism, unpredictability and weak controllability. The operational environment is subject to sudden and unexpected changes, actors may join and leave the system at will, social dependencies between actors are at risk because of actors' autonomy, and actors may fail in achieving their goals. The interests of the actors can be supported technologically by introducing a software architecture that monitors the actors behavior, diagnoses failures against correct behavioral models, and reacts to failures via compensation actions. This research proposes an architecture intended to make STSs self-reconfigurable. The architecture becomes an integral component of an STS; the goal model itself serves as a description of correct behavior. The architecture observes the actions performed by participating actors, compares the monitored data against goal models, and enacts reconfigurations in response to failures.

The algorithms the architecture exploits are based on the specialization of the Belief-Desire-Intention paradigm. An agent participating in an STS behaves correctly if, whenever a goal is activated, it will select and execute a plan that eventually leads to the achievement of that goal. Failures occur if the agent does not follow the plan correctly, if the agent does not execute anything, or if a dependee does not bring about the dependum for the depender. Reconfiguration actions take into account the autonomy and uncontrollability of the participants: the architecture can (i) perform some real action by controlling actuators; (ii) remind or suggest actions to actors; and (iii) assign some responsibilities to external agents.

- Acquiring variability in fulfilling requirements [94, 68]: for supporting high-variability the system is supposed to have relatively large number of alternatives. The work discusses a systematic way to generate these alternatives at the goal level. More specifically, it discusses how to generate alternatives through OR-decomposing a goal. A semantic anatomy of what is OR-decomposed is given. A goal can be associated with a set of concerns and the OR decomposition can consider each of these concerns (Agentive, Dative, Objective, Factitive, Process, Locational, Temporal, Conditional, Extent).

  For example, the Agentive concern refers to the agents that are in charge with reaching the goals, the Dative concern refers to the agents who will be affected by the fulfillment of a goal, the Objective concerns

refers to the objects that will be affected by the activities done to reach the goal, the Locational concern refers to the spatial locations where the activity to reach the goal is established, and the Temporal concerns refers to the time in which such activities are perform.

- Goal-driven software customization [95, 62]: this work complements the previous one. While goal models allows for a space of alternatives to reach goals, a criteria to customize the goal model is needed. This research allows for customizing goal models based on two dimensions. The first one is the skills of user that may be required for each of the tasks (leaf goals) of goal model, and the user's preferences that may rank goal satisfaction alternatives differently.

  To illustrate the customization based on user skills, let us consider the example of conveying information to a user. For example, to show information to a user interactively through his PDA, this requires a skill like knowing well about how to use touch screen and respond to interactive presentation techniques. If user does not have these skills the information can be delivered as a video-like presentation. To illustrate the user preferences dimension in ranking the alternatives, let us take the information delivery process by an assistance staff in a shopping mall to a customer. This process can be done remotely or in person. Remote calling is positive from the perspective of staff comfort and negative from the perspective of the quality of information delivery as whole. Delivering information in person is as opposite to remote calling for the last two quality measures (staff comfort, and quality of information). If the mall adminstration appreciates staff comfort more than the quality of delivered information, then remote calling will be adopted and vice versa.

2.3.2 Problem Frames variability

Problem Frames is a requirements engineering approach that is for analyzing software problems and their context [96]. This approach adopts three descriptions to characterize a problem. The first one is W: the description of the context (context here refers to known domain properties of the world). The second is R: which is the required domain properties. The third one is S: the specification of what the software (machine) has to do to reach R. Problem Frames do support a kind of variability through the notion of variant frames. A variant frame is a variant of a basic problem frame where additional problem domain is added, or the control characteristics of a shared

phenomenon are changed. This allows for variability when the problem does not fit to the basic frame.

Salifu et al. [97, 98], investigate the use of problem descriptions to represent and analyze variability in context-aware software. The work recognizes the link between requirements and context as a basic step in designing context-aware systems. The work applies Problem Frames approach to analyze different specifications that can satisfy the core requirements under different contexts. The problem description captures the relationship between contexts, requirements, and the specification (machine). For each different context there could be a different problem description. This leads to identification of variant problems that are variations of the original problem that are adapted for a particular context. A context-aware system is a composition of the specifications to the variant problem.

The specification of the system in this way allows it to be adaptive to the context. A context change that violates the requirement triggers a switching action to an alternative specification for restoring the satisfaction of requirements. In other words, the requirements has to be always satisfied, and when the monitoring systems observes a failure in meeting the requirements because of changes in the context the current configuration fits to, this will lead to looking for alternative problem variants that fits to the new context.

2.3.3 Feature models variability

Software product line variability modeling, mainly feature models [99, 100], concerns modeling a variety of possible configurations of the software functionalities to allow for a systematic way of tailoring a product upon stakeholder choices. Features are characteristics of the system, and feature model represents the variability of these characteristics for configuring a family of software products. However, there is still a gap between each variant and the context where it can, or has to, be adopted. Speaking in terms of feature modeling, context can determine if a feature is mandatory or optional or even redundant. For example, for an email editing system, encryption could be an optional feature if the system is to operate within one organization where staff members trust each other. On the other hand, such feature could be mandatory if the editor is for users who will write emails from a public network.

Modeling and analyzing variability in software product lines is a broad topic and for the purpose of this thesis we focus on two directions: the feature variability and context, and the integration between goal-modeling approaches and features:

- Feature variability and context: in product lines engineering, context can be a main factor in deciding what product variants to derive. In other words, context influences the need for, the applicability and the appropriateness of, each variant. We use the term self-contextualizability to denote a system ability to adapt to context in order to keep its objectives satisfied. Self-contextualizable product line is a product line that incorporates the reasoning needed to derive product variants fitting to their contexts. Consequently, the relation between products variants and context has to be explicitly captured and reasoned on to derive contextualized products. Context influences the set of features to be included in a software product variant. Considering context at the design time can model a feature as mandatory or optional, whilst at the runtime context needs to be considered when switching to an alternative feature. Feature models can be enriched with context towards more systematic derivation of software variants. A recent work by Hartmann et al. [101] studies the relation between context and features to support the engineering of software supply chains. Context can determine if a feature is mandatory or optional or even redundant.

- Goals and features integration: goal models are used as an intentional ontology that fits well with the early requirements analysis phases. They support analysis of different alternatives for satisfying user needs. As proposed in [102], a goal model is a good starting point for feature model construction as it justifies feature configurations in terms of stakeholder goals. In this thesis, we advocate a perspective on a goal model as a core domain model: it acts as the source of all stakeholder-related variability. Goal models justify existence of all functional requirements (hard goals) and quality measures (soft goals) of a software system in terms of stakeholder intentions. Thus, variability in intentions is a primer source of system variability. Of course, goals are not the sole source of system variability - they pertain to variability of the problem domain. Technical solutions devised to satisfy these goals will add their own variability dimensions. The latter however, are meaningless without the former, as a system will be useful only if it is addressing some set of stakeholder needs. Recent work has emerged on the relation between goal models and feature models. For example, Antonio et al. [103], develop IStarLPS framework that adopt i* framework for software product lines. It provides a systematic way to identify feature from i* goal models towards more comprehensive requirements engineering for software product lines.

2.4 Chapter Summary

In this chapter, we have reviewed the state of the art of several research areas related to this thesis. We have reviewed several definitions of context in context-aware computing and discussed several classifications of contextual information and several modeling techniques for representing context. We have reviewed the main concepts and motivations in goal-oriented requirements engineering and a variety of ongoing research. Finally, we discussed main-stream requirements engineering approaches in capturing and managing system variability together with context.

Chapter 3

Contextual Goal Model

In this chapter, we propose *contextual goal models* to capture the relation between variants for goal salification and context. As we have introduced in Chapter 1, context may have influence on human's goals and their choices to reach such goals. The relationship between context and goals is essential for software developed to meet users needs in different contexts. Software has to reflect user adaptation to context, while reaching his goals, to derive a set of functionalities to execute. We explain the principles our modeling is based on, give an overview of Tropos goal modeling, and then propose an extension to it:

- we define a set of variation points on Tropos goal model where context might intervene to decide applicable alternatives for goal satisfaction.
- we define a set of constructs to analyze context. This analysis is for discovering the information that a system needs to capture of its environment to judge if an analyzed context holds.

3.1 Weaving Requirements with Context

Most RE literature ignores, or presumes a uniform nature of, the context in which the system operates. However, new advances in communication, computing, and sensing technologies have introduced novel fields and applications areas where this assumption is not anymore valid. For example, applications for ambient intelligence introduce an inherent relation and mutual influence between varying contexts and requirements. Requirements and the system actions to meet them could change according to the varying context. The context, in turn, may change as consequence of the actions the system takes. In order to weave requirements with contexts, we need to support:

1. *a way to represent variability of the system*, or in other words an explicit representation of variants the system can use to adapt its behavior to the changes of the context. For example, possible variants for circulating the air inside the smart home can be: (i) opening the windows and (ii) turning the ventilator on;

2. *an explicit association between variants and contexts*, to allow for a systematic derivation of variants adoptable in each context. For example, if the "humidity level inside home is high" (context), we need to circulate the air. Opening the windows to circulate air (variant) is adoptable if "it is good weather outside" (context) otherwise ventilator could be turned on;

3. *an explicit representation of the influence of context on the quality of each variant*. A variant can be evaluated against a set of quality measures and this evaluation can be context dependent. For example, the variant of opening the windows to circulate air inside home may be evaluated negatively against a quality measure like "patient feels more privacy", but only in certain contexts such as "patient is sleeping";

4. *a systematic analysis of context*. An analysis to discover the information necessary to identify a certain context is needed. For example, "patient is interested in the current TV program" is a context that we may need further analysis to judge if it holds. To this end, we may check if other two contexts "the patient is sitting/lying in front of the TV" and "the patient is not doing any other significant activity but watching the TV" hold, and so on.

5. *an explicit representation of the influence of requirements over the context*. This helps to analyze and reason about the problematic interplay between system's behaviors and changes of the environment. For example, "circulating the air by opening the windows" and "locking external doors and windows to protect home against robbery" leads to a conflict preventing the satisfaction of both requirements together.

Context may influence not only the system, but also the user. This means that the system has to adapt its behavior to context taking into account the ways users themselves may adopt to reach their requirements in different contexts. Goal modeling, that is a mainstream approach in requirements engineering, is very effective to represent users goals and analyze alternative ways for their satisfaction besides the qualities of these alternatives (softgoals [10]). Goal models can be used to capture the adaptability to context

at the intentional level as a preliminary and essential step for the software adaptation to varying contexts.

3.2 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 conceptual model, the contextual goal model. We consider a mobile information system for promoting products to customers inside shopping malls. The customers and sales staff are provided with 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 that may include 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 activate 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 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 a product already chosen or bought 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 pick up the product from.

Another alternative to promote a product is by 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 description of 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 alternative of 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.

3.3 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 [15]. 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".

Here we illustrate the main concepts of Tropos goal modeling that will be used in the rest of the thesis:

- **Actor**: an entity that has strategic goals and intentionality within the system or the organizational settings. For example, a customer represents a human actor, while customer mobile information system represents a system actor.

- **Goal**: represents an actor's strategic interest that has clear-cut definition and clear-cut criteria to judge if it is satisfied. For example, "customers get connected to the mall network" is a goal that we know when it is satisfied and we have specific ways to satisfy it.

- **Task**: is an executable process that represents a way of doing something. For example, "show a demo about the use of information terminals in the mall through user's PDA" is an executable process that is a way to convey information about the use of terminals to customers.

- **Softgoal**: represents actor's strategic interest that has no clear-cut definition and/or no clear-cut criteria to judge if it is satisfied. Softgoals are typically used to model non-functional requirements. For example, "customer is more comfortable" is a goal that we do not know exactly how it can be measured or satisfied.

Tropos allows the above concepts to be related via a set of relationships:

- **Contribution to softgoals**: certain goals/tasks can contribute positively/negatively to softgoals "satisficing". For example, the automated wireless connection of the customer's PDA to the mall network contributes positively to the softgoal "customer is more comfortable".

- **Dependency**: occurs between two actors and indicates that one actor depends on the other in order to attain a goal, execute a task. The former actor is called the depender, while the latter is called the dependee. The object around which the dependency is based is called dependum. For example, the customer might depend on the sales staff for reaching the goal "get informed about a product" or executing a task like "getting a sample of a product".

- **Means-end**: goals can be finally reached by means of executable processes (tasks), i.e., a task is a way of reach a goal. For example, the tasks "show the path to the nearest information terminal" and "trace and guide customer to the nearest terminal" are means to reach the goal "customer arrives to the terminal".

- **Decomposition**: goals and tasks can be refined through AND- or OR-decomposition. If a goal G (a task T) is AND-decomposed into subgoals $G_1, G_2, ..., G_n$ (Subtasks $T_1, T_2, ..., T_n$) then if all of the subgoals (subtasks) are satisfied (executed) so is the goal G (the task T). If a goal G (a task T) is OR-decomposed into subgoals $G_1, G_2, ..., G_n$ (Subtasks $T_1, T_2, ..., T_n$) then if at least one of the subgoals (subtasks) are satisfied (executed) so is the goal G (the task T).

In Fig. 3.1, 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*).

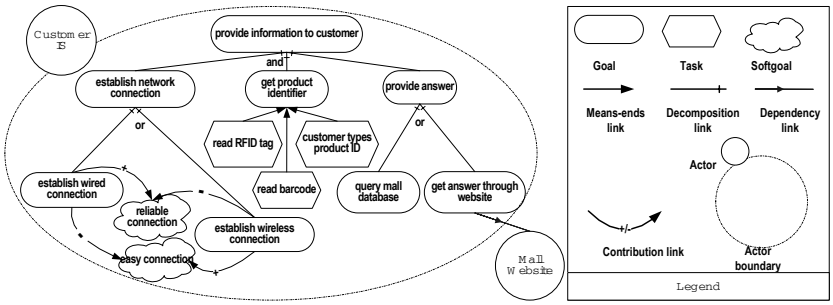


Figure 3.1: Tropos goal model example

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*.

3.4 Context in Requirements

Context has been defined in multiple computer science disciplines especially in artificial intelligence (for a survey see [29]). It has been also defined in the literature of emerging computing paradigms, such as ubiquitous, adaptive, and mobile systems [32, 1, 37], 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 [104]. 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 [6, 9, 15, 8]. 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 [12], 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 to 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 whether 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 sales staff may attempt 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 capable 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 [37]. 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 static 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 shopping malls the system will operate in.

3.5 Weaving Context with Goals

Context has a strong influence on system requirements: it can be a factor in deciding what requirements to meet, choosing among possible ways to meet the requirements, and assessing the quality of each of these ways. On the other hand, the system itself may cause changes in the context as a consequence of meeting its requirements. However, in spite of the mutual influence between context and requirements, context is either ignored or presumed uniform in RE literature. A RE model for systems reflecting their context is still missing. An early analysis of the mutual influence between requirements and context may improve the quality of the software and reduce the probability of revising the design during the development process. Since requirements are expressions of stakeholder's needs, that is originated by their goals, also context should be analyzed along this dimension.

As we mentioned in the previous section, goal analysis answers the question "why is the software needed". We may judge the usefulness and the completeness of the software in terms of its establishment of users goals.

Capturing the relation with context at this level, the goal level, is preliminary and basic step for a final developed system operating and reflecting varying contexts. It answers a bigger question that is: "why and where/when is a software needed".

Context influences user intentions and choices before the software itself. Software has to reflect user adaptation to context for deriving useful functionalities to execute. Goal models, that is a mainstream approach in requirements engineering, represent an intentional ontology that captures users goals and the variant ways to satisfy these goals, together with the quality of each variant through the notion of softgoals [10]. Consequently, goal models have the potential to capture the adaptability to context at the intentional level as a preliminary and essential step for the software adaptation to varying contexts.

Goal analysis allows for different variants (alternatives) to satisfy a goal, but does not specify explicitly when each variant can be adopted. Supporting variants without specifying when to follow each of them raises the question *"why does the system support several variants and not just one?"*. On the other side, the consideration of different contexts the software has to adapt to, without supporting variants raises the question *"what can the system do if context changes?"*. Analyzing different variants for satisfying a goal, and specifying the relation between each variant and the corresponding context help for justifying both the variants and context, and for having a well-defined requirements for varying contexts.

Example 6. For a promotion system inside shopping malls, a sales staff may have a goal like *"product is promoted"* activated in a context like *"customer is not in a hurry and there is enough time to promote the product"*. The sales staff, and depending on the context can choose variant ways to reach this goal. If the context *"customer does not know about the product or the product is local to some region the customer is not from"* holds, the sales staff may give a sample of the product. From the other hand, if the context *"the product complements another one that the customer has"* holds, the sales staff may cross-sell the product. A promotion information system has to reflect the goal and the rationale and the adaptation of the sales staff when promoting a product. In other words, one source of software adaptation is the human's one.

Example 7. For a tour guide, if a context like *"tourist has not had lunch today and time is around lunch hour"* holds, the tour guide will try to reach a goal like *"find a place for tourist to eat"*. Moreover, the context *"tourist is vegetarian"* will limit the list of restaurants from which the guide would choose. A mobile information system, playing the role of tour guide, has to

reflect the guide goals, rationale, and adaptation to context. This reflection is preliminary before executing useful functionality like showing the map to a suitable restaurant and translating the menu.

3.6 Contextual Goal Model: Variation Points

We propose to integrate goal model, as an early requirements model, and context in order to capture the relationship between the goal model variants and context. A **goal model variant** is an And-tree (i.e., deterministic tree) of the goal model hierarchy and represent one alternative to satisfy root goals. We need to specify the relation between each goal model variant and context. In other words, we need to specify the context in which the variant is needed, the context in which the variant can be adopted, and the context in which the variant is good enough from the perspective of each softgoal. Enumerating the variants of a goal model and specifying the context for each of them separately is a hard task for two main reasons:

1. the potentially huge number of goal model variants, i.e., specifying context for each enumerated variant could be extremely time consuming.

2. when a variant contains a large number of nodes, it will be hard for the analyst to comprehend it and, consequently, specifies the context for it in one step.

Fig. 3.2 represents a goal model for the promotion information system explained in Section 3.2. The system is intended to interact with customers and sales staff, through their PDAs, in order to meet the root goal "promote products to customers". The goal model represents the multiple alternatives that the system may adopt to reach such goal. To make the model contextual, we need to explicitly represent the relation between these alternatives and context. Contexts, labeled by $C1..C13$ in the figure, can be related to the following variation points at Tropos goal models:

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 free sample from a machine"* can

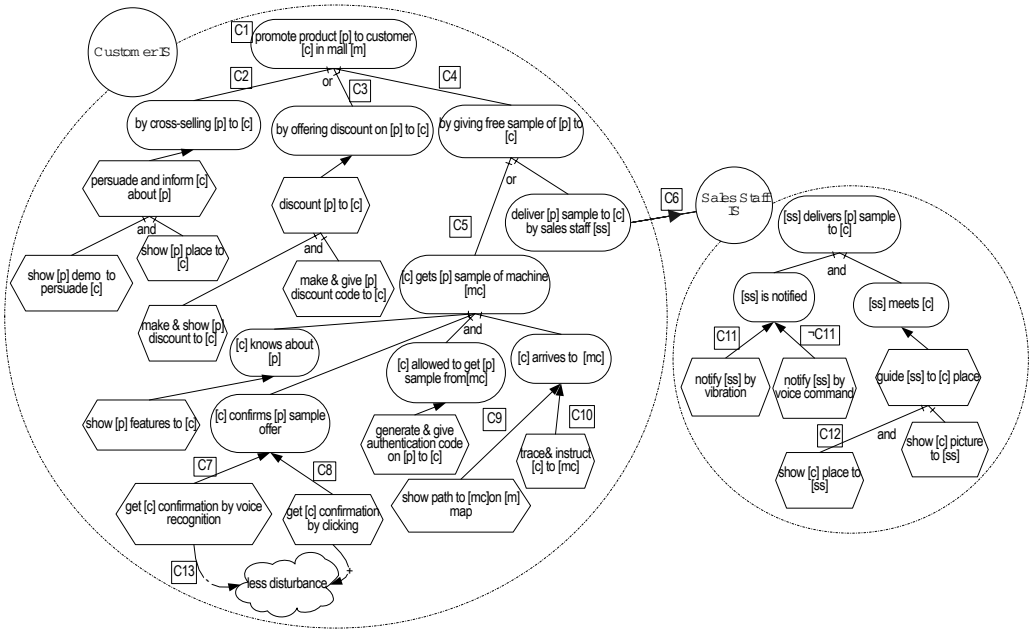


Figure 3.2: A goal model annotated with contexts at its variation points.

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-ma"* 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_{13})

3.7 Context Influence on Goals: A Classification

For systems living in and reflecting varying contexts, there is a strong impact of context on the set of requirements to derive. Context might decide what requirements to meet, how and how well they can be met. Based on its impact on requirements, we classify context in three kinds, each of these kinds is represented at a set of variation points of our proposed contextual goal model:

1. **Activation context**, when the context makes it necessary to achieve (execute) a set of goals (tasks). In our contextual goal model, activation contexts are those contexts at the variation points (i) *root goal* and (ii) *And-decomposition*. They decide if 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.

2. **Required context**, when the context is necessary to adopt a certain way for achieving (executing) a set of activated goals (tasks). The contexts that are at the variation points (i) *Or-decomposition*, (ii) *Means-end*, and (iii) *Actors dependency* are required contexts. They are required to make applicable a variant of the goal model. The required context of a goal model variant is the conjunction of contexts at the variation points of these three kinds.

3. **Quality context**: when the context influences the quality of a variant of the goal model. Only the contexts at the variation point *Contribution to softgoals* are quality contexts. Contributions (links) to softgoals are, indeed, used in Tropos to capture the impact of a goal/task to a quality (i.e., softgoal).

In the rest of this thesis, we refer by the **context of a goal model variant** to the conjunction of activation and required contexts of that variant. Fig. 3.3 shows two goal model variants (from Fig. 3.2) and the contexts associated to them. The classification of context into these three categories allows us, amongst other things, to answer questions like: in a given context, does the system need to meet some requirements? what are the possible ways to meet them? and what is the quality of each of such ways? Moreover, in Chapter 4, we show how we can exploit contextual goal models to analyze and reason about different properties of the system at an early stage of the development.

3.8 Contextual Goal Model: 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. 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.*

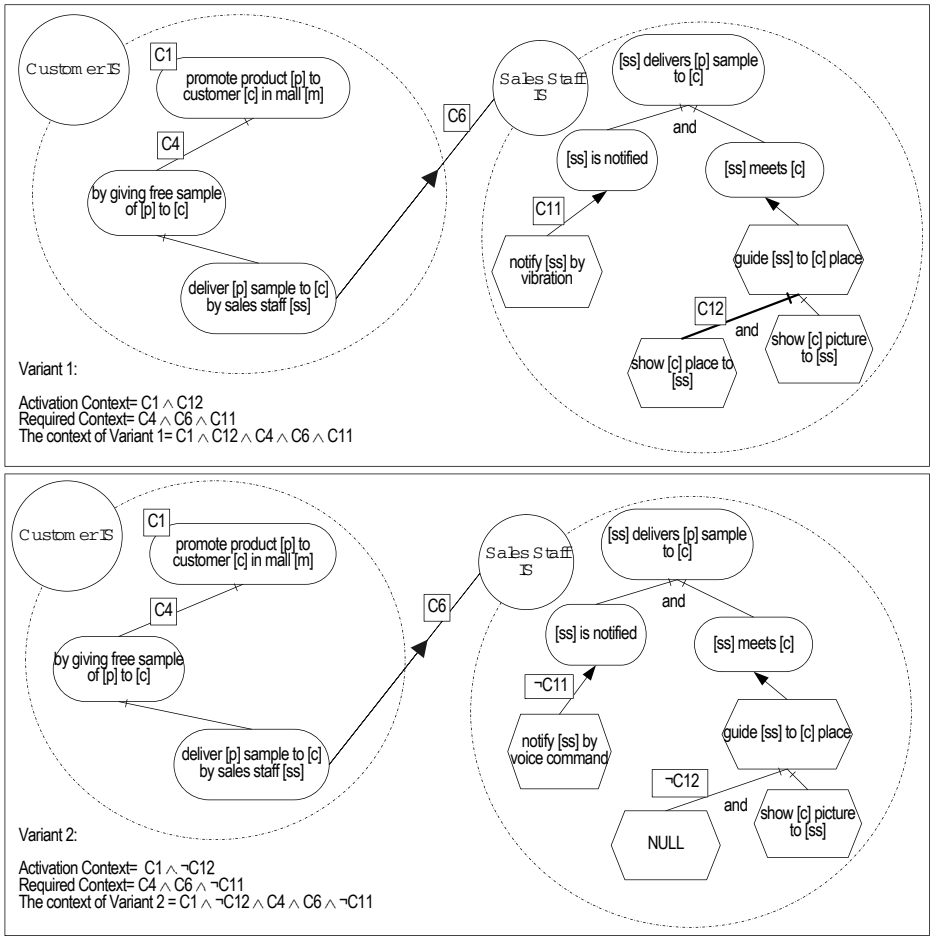


Figure 3.3: Goal model variants and their contexts

Code 1 The EBNF of world predicates formula

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

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:

- the 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.

- the abstract (soft) 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 inter-

ested 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 enough 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 3.2, 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. 3.4, 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.

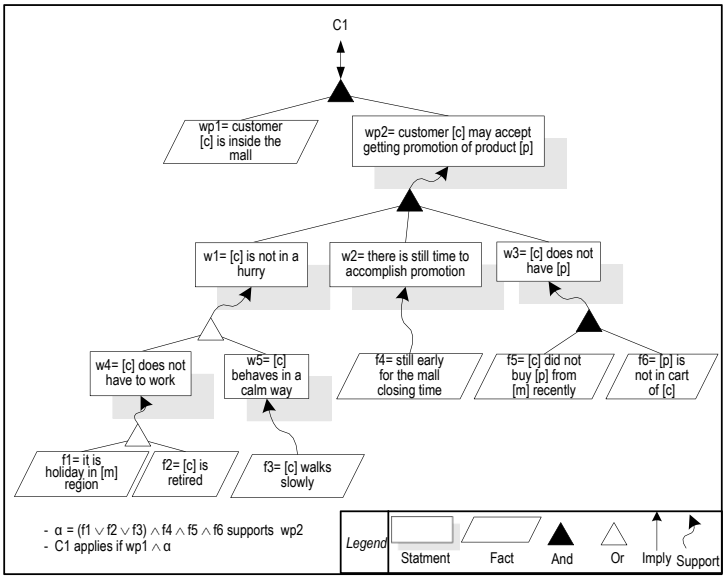


Figure 3.4: The context analysis for C_1

As we mentioned earlier, we consider the relation *support* as a transitive relation. For example, as shown in Fig. 3.4, 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 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 helps the analyst to discover what data the system has to collect of the world. The analysis allows us to identify the facts that has to be verified. 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. 3.4, an analyst could develop a data conceptual model, shown in Fig. 3.5, 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.

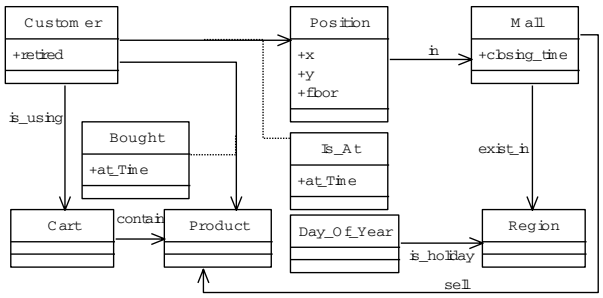


Figure 3.5: The conceptual model of data needed to verify C_1 facts

To illustrate our proposed constructs to analyze context, we show more examples of context analysis and their corresponding data models in Fig. 3.6.

The analogy between goal and context analysis is shown in Fig. 3.7. 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.

3.9 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, hold and therefore the deployed software for that mall can exclude the functionality of voice recognition as a way of interaction with customers.

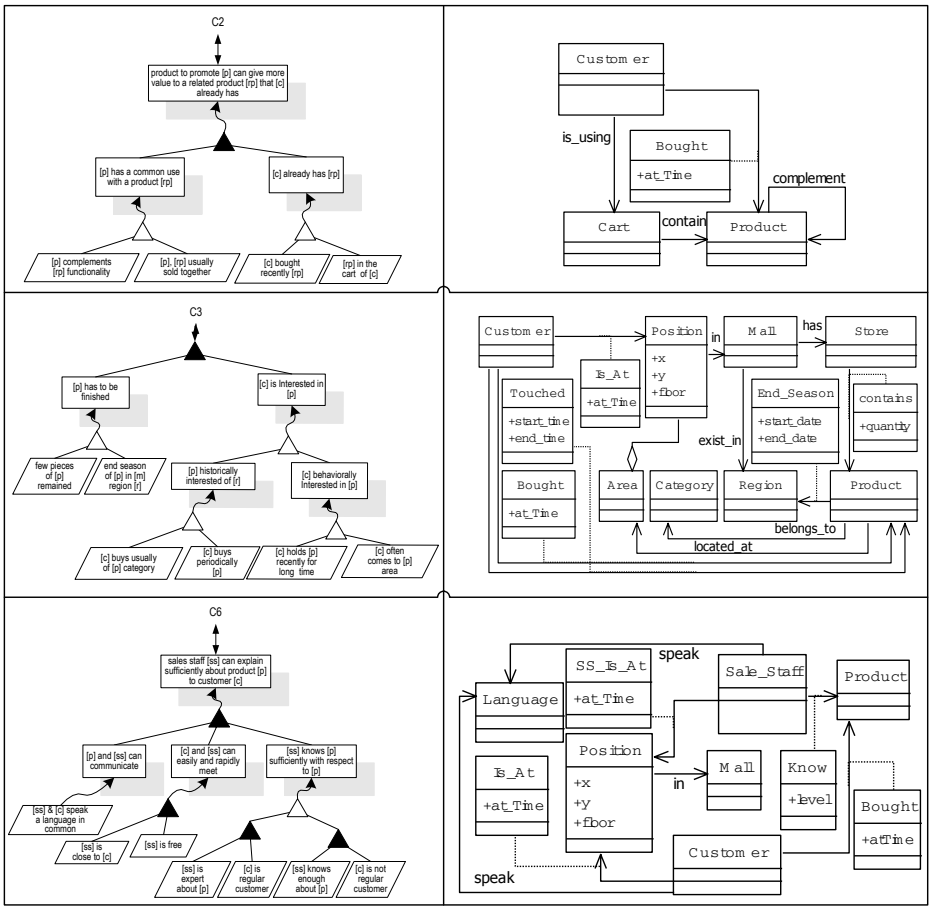


Figure 3.6: Examples of context analysis and elicited data models.

- *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 verifying 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.

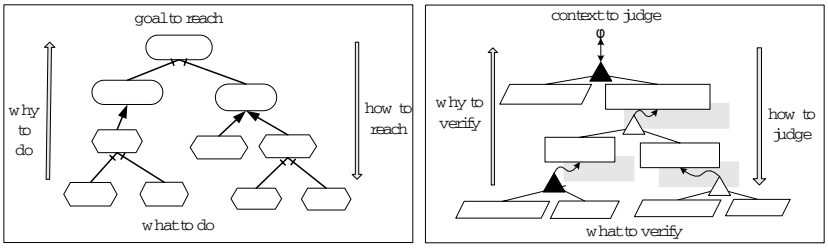


Figure 3.7: The analogy between Goal and Context Analysis

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 8. 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 < 60 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. 3.8a. 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. 3.8b.

In this thesis, we presume that it is always possible to specify the context for each alternative way to meet the requirements. However, in some cases, it could be hard to specify the relation between the alternatives and context at one step for two main reasons:

- some specifications require validation based on the actual operation of the system.
- not every validated relation remains infinitely valid.

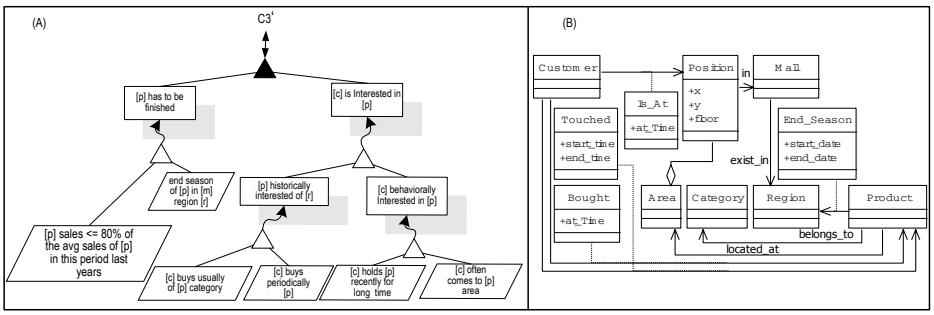


Figure 3.8: A modified context C_3' and its data model.

Consequently, a lifelong contextualization is needed. A possible future work direction could concern developing mechanisms to maximize the autonomous contextualization of requirements continuously. This will allow the system to learn from its experience in certain environment and optimize the way it meets its requirements in. For example, certain people/cultures consider a sales staff coming on person to promote a product as irritating behavior, while some others appreciate it. A system after operating for a period of time in one environment should know how successful each alternative way to promote a product was and contextualize itself autonomously.

3.10 Chapter Summary

In this chapter, we explained our proposed conceptual model. We have motivated the importance of considering context at the goal level as a basic step for software adaptation to context. We have discussed a set of variation points on Tropos goal model where context may intervene to take a context-based decision. We have showed the motivation beyond the variation points that is essentially avoiding the enumeration of all the goal model variants and specifying context for each of them separately. We have classified the semantics of the variation points through classifying context into three categories, each category is associated to a set of variation points: activation, required, quality contexts.

One main contribution of this thesis is the context analysis. Context analysis provides a set of constructs that allow for a systematic way of analyzing context and discovering ways to judge if it holds. In other words, this analysis is to discover what is visible in the environment and that leads to judge if a context holds. The motivation behind context analysis is similar to that of goal analysis. While goals are state of the world to reach, context

is state of the world that is the case. We analyze goals to discover possible executable process to reach them, while we analyze context to discover possible monitorable facts to judge if it holds.

Chapter 4

Reasoning about Contextual Goal Models

In this chapter, we develop the following reasoning techniques about contextual goal models:

- **Reasoning about consistency**: to verify the consistency of a designed contextual goal model, we develop two analysis techniques:
  - *Reasoning about context consistency*: we check the consistency of contexts specified at a contextual goal model. Contexts, both the individuals which are specified at the variation points and the accumulative, such as activation and required contexts, are represented as formulae of world predicates. The logical relations between the variables (world predicates) of that formula may make it inconsistent (unsatisfiable). We process the variants of a contextual goal model to detect the set of variants unadoptable due to inconsistency of their contexts and explain different semantics of such an inconsistency.
  - *Reasoning about conflicts*: we check each variant of a contextual goal model for conflicts between its executable processes (tasks). The conflicts we detect, are those manifested via inconsistent changes on the context that the execution of tasks leads to.
- **Reasoning about variants derivation**: to automate the derivation of a contextual goal model variants, we develop two analysis techniques:
  - *Variants derivation for varying contexts*: this technique concerns the automatic derivation of goal model variants that reflect context and user priorities at runtime.

– *Variant derivation for minimum-cost system*: this technique concerns processing a contextual goal model to extract the variants leading to a system developed with minimal costs and still able to meet users goals in all considered contexts. This reasoning is useful at design time to decide the core requirements the system has to meet when there are budget or timing constraints.

4.1 Reasoning about Consistency

In this section, we develop reasoning about the consistency of contexts specified for goal model variants, and reasoning about the harmful interplay among tasks manifested on context. In fact, the earlier we reason about the system, the better we discover and manage errors leading to a final system different from what stakeholder intended. Considering that goal models fit well to the early stages of the software development [15], the consistency analysis at the goal level represents an early step towards a final system correctly developed.

4.1.1 Running example

We take a running example of a smart home system for people with dementia[1]. A smart home is a ubiquitous computing scenario where computing is integrated with our living environment. In smart homes, context is monitored as an implicit input. This input influences what objectives the system needs to meet and how it meets them. Smart homes are used in areas like elderly support, health care, and entertainment.

We consider here a smart home designed for patients with dementia, a variant of the scenario described in [105] and used in the EU sponsored Serenity project[2]. The smart home supports some daily tasks that the patient might forget to do, such as eating, circulating the air inside the home, taking medicines. Besides their memory impediments, patients with dementia suffer from anxiety attacks. The smart home should manage such situations by making the patient aware of the anxiety attack, or by preventing him from getting out of the house in an unusual way. The home then needs to calm the patient down, and call the caregiver to come and administer a treatment. The smart home supports also some other general tasks, such as preventing a potential robbery of the home (e.g., it can give the illusion that the home is lived in when the patient is out).

[1] In Chapter 6, we will process the entire model of the Smart Home

[2] The Serenity project focuses on System Engineering for Security and Dependability, and involves 15 European R&D partners. Website: http://www.serenity-project.org/

Context influences, and is influenced by, the smart home requirements. For instance, in the context "patient is anxious, behaving in an unusual way, and is on the way to get out the home" the smart home is required to ensure certain exit procedure. In the context "the patient is moderately anxious and his dementia disease is not severe" the smart home could just give an alert as a way to ensure the exit procedure. In the context "patient is very anxious and his dementia is severe" then the smart home could lock the outer doors and windows. This last alternative could cause changes in the smart home environment. The external doors and windows are locked, and there could be less light inside home.

Technology plays a crucial role in smart homes. Environmental sensors allow for the gathering of information concerning temperature, humidity, the level of light, magnetic fields (also installable on doors and windows to check whether they are open or closed), electrical current. Motion sensors can identify if something or somebody is moving in a certain area, whereas accelerators allow to detect sudden movements such as falls. RFID tags can be used as an authentication mechanism by recognizing people and objects transparently. Medical sensors gather information concerning a person's health: pulse oxymeters detect saturation level and heart rate, while smart-shirts provide even more accurate information (EKG, respiration, temperature, ...). Cameras serve a wide variety of purposes such as people tracking.

In Fig. 4.1, we present a partial goal model for the smart home system. We are going to use it as a running example for explaining the reasoning techniques we propose in this section. The description of the contexts annotated at this goal model is shown in Table 4.1. We show an example of context analysis in Fig. 4.2 and in Fig. 4.3 we show the model of data that the smart home system has to collect to verify the facts of the analyzed context.

4.1.2 Reasoning about context consistency

Context analysis allows us to refine contexts at the variation points of the goal model and discover formulae of facts that specify them (for an example see Fig. 4.2). We remind here that only monitorable contexts are allowed in our contextual goal model, i.e., the contexts that are refinable to formulae of facts. The formula of an accumulative context, such as the context of a goal model variant, is the conjunction of the formulae of those individual contexts at each relevant variation points [3]. A formula expressing a context could be inconsistent which could be a modeling error to fix.

In order to check the consistency of a formula specifying a context, we

[3]We have defined the classification of variation points in Chapter 3. See also Fig. 3.3

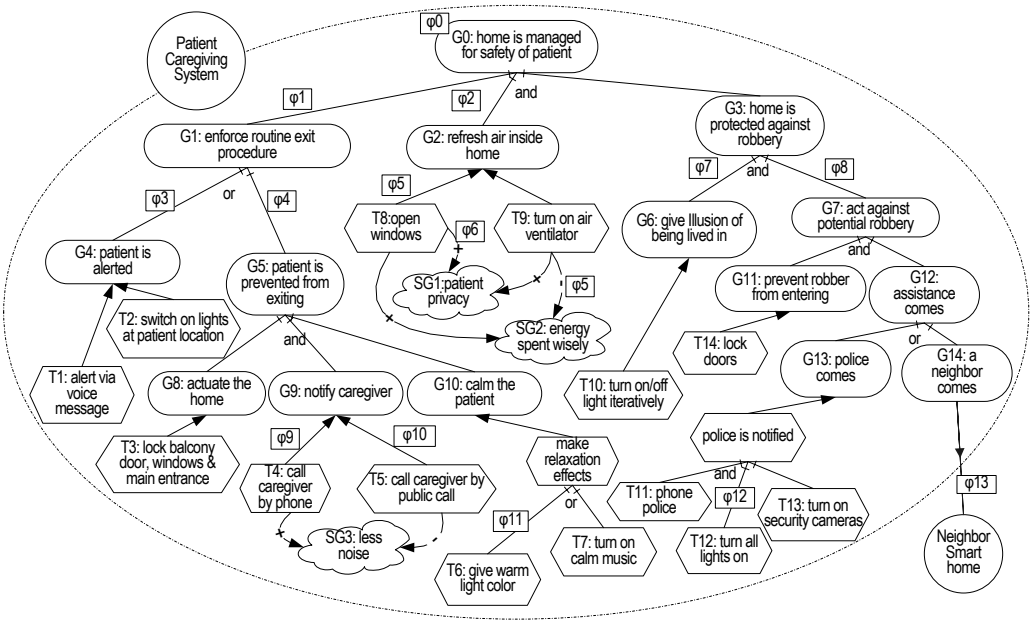


Figure 4.1: Partial contextual goal model for Smart Home system.

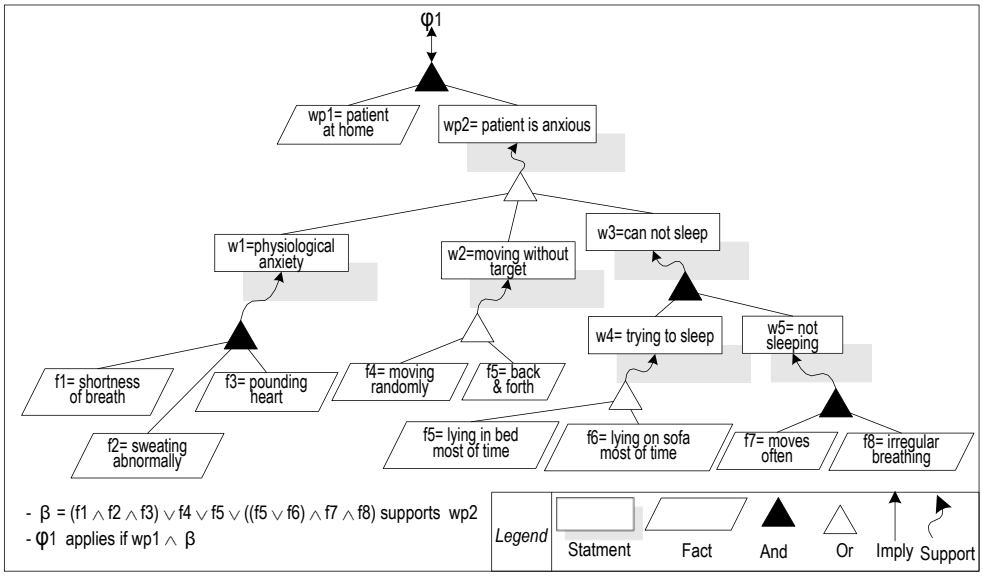


Figure 4.2: The context analysis of φ_1.

	Description	Technology
φ_0	Home is lived in, and the patient is expected to have some dementia problem, and there is no awaken caregiver or healthy relative at home.	Database (info about home and patient), RFID tags (caregiver and relative)
φ_1	Patient is anxious and he is at home.	Smart-shirt or oxymeter, camera with motion recognition
φ_2	Humidity level in the house is too high, or home windows and doors haven't been opened for long time.	Humidity sensor, magnetic sensor (open-close), database
φ_3	The patient dementia disease is not in an advanced stage and he is moderately anxious.	Database (disease status), smart-shirt (anxiety)
φ_4	The patient suffers of advanced dementia, or 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 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 telephony company, phone busy sensor
φ_{10}	It is not night time.	Digital clock
φ_{11}	The light level at patient location is too low or too high.	Light sensor
φ_{12}	It is too dark inside home.	Light sensor
φ_{13}	The neighbor is healthy, is at home, and can see or reach easily the patient home.	Database (health status and house location), GPS/RFID (neighbor position)

Table 4.1: The description of contexts specified at the contextual goal model of Figure 4.1.

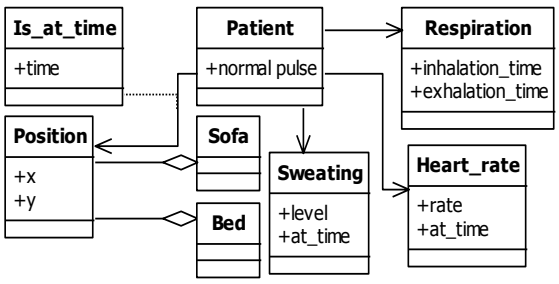


Figure 4.3: The data needed to verify φ_1 facts.

also need to take into consideration all possible contradictions among the variables (world predicates) of that formula. For example, in Figure 4.1 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 that 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 wherever the system operates. 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 whatever institute the system operates in.

2. *Operational environment dependent* relations are true in a particular environment where the system operates without any guarantee that such relations hold in other operational environments. For example, lets us 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 always false. Therefore, we have to consider a special kind of environment dependent relations: $Env \rightarrow world\_predicates\_formula$. For example, if the system operates in an institute for patients with severe dementia exclusively, then the implication $Env \rightarrow \neg wp_3$ where wp_3 =*"patient has basic dementia"* always holds.

We apply SAT-based techniques [106] 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[4], 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. The pseudo-code of the algorithm (*CheckSAT*) is reported in Figure 4.4.

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

Figure 4.4: Checking context consistency under assumptions (CheckSAT)

Obviously, the context at each variation point has to be consistent, otherwise it is a modeling error to fix. The accumulative contexts, such as activation and required contexts for goal model variants, could also be inconsistent. However, the inconsistency of these accumulative contexts does not always indicate a modeling error and fixing or accepting such an inconsistency is an analyst's decision. The compact form of goal models integrates a large number of variants and may, as a side-effect, include variants that are not practically needed and their context inconsistency is acceptable. Moreover, the semantic of context inconsistency depends on the kind of accumulative context in which it happens. In what follows, we illustrate the above ideas via examples taken from the contextual goal model of Figure 4.1.

Example 9. The inconsistency of the activation context of a goal model variant means that the variant is not needed. The variant shown in Figure 4.5 has an inconsistent activation context because of the contradiction between $\varphi_1 = wp_{1.1} \wedge wp_{1.2}$, 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". In this example, the variant is practically inapplicable and 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 in the home area. However, given that these two requirements are not

[4]In this thesis, we suppose that the relations between variables are manually provided.

needed at the same time, the designers could accept the mentioned context inconsistency. In some other cases, inconsistency of activation contexts has to be fixed. Let us suppose that φ_0 is modified to 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 treating it as a modeling error.

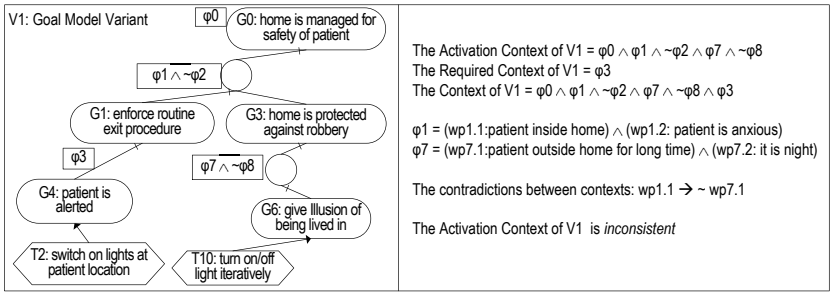


Figure 4.5: A variant with an inconsistent activation context

Example 10. The inconsistency of the required context of a goal model variant that has a consistent activation context means that the variant can be activated but it is unadoptable in any context. In other words, a set of requirements could be activated but a certain way (variant) to meet them is unadoptable. Figure 4.6 shows an example of inconsistent required context. In this example, the administration of the health care institute decides that calling caregiver through institute speakers requires that patient has extreme anxiety, while in the other cases caregiver could be called by phone. Therefore, φ_9 is modified into φ'_9 that adds the fact "patient anxiety is moderate" which make $\varphi'_9 \wedge \varphi_4$ inconsistent. In this new specification, the context required for calling caregiver by phone never holds and designers would decide to fix the inconsistency.

Example 11. 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 but never adopted in the context where it is activated. Figure 4.7 shows an example of 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 make the context of the variant V_3 inconsistent. If T_5 does not appear in other goal model variants with a

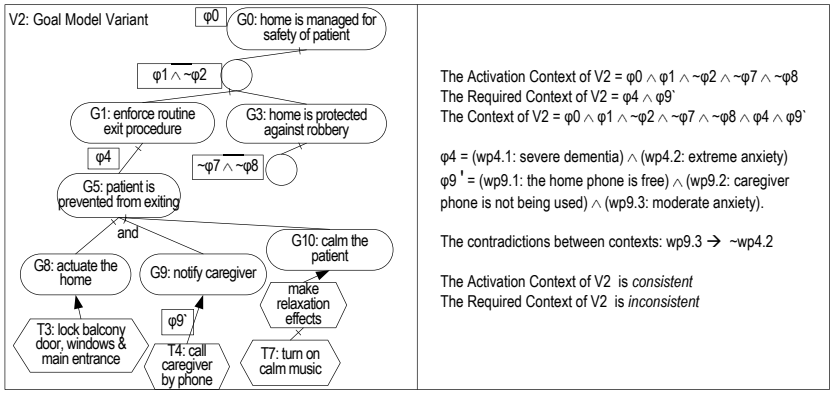


Figure 4.6: A variant with an inconsistent required context

consistent context, one design decision could exclude it from the implemented system.

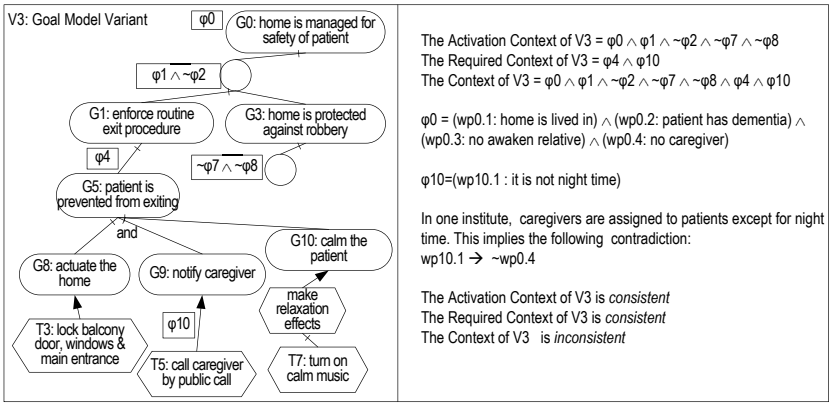


Figure 4.7: A variant with an inconsistent context

Example 12. The inconsistency in quality contexts happens 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. For example, the administration of some institutes could consider calling caregivers through the institute 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 in Fig. 4.1 will be preconditioned by the context φ="it is night time". Since T_5 requires day hours time then

$\varphi_{10} \rightarrow \neg\varphi$ and, therefore, there will be no contribution between T_5 to SG_3 and the designers could just remove this contribution from the model.

4.1.3 Conflict analysis

Adaptability to context indicates a high degree of autonomy and flexibility that the system has for achieving 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 be inconsistent and originate conflicts preventing the right achievement of user's goals. Understanding conflicts is preliminary for their resolution and requires to answer questions like:

- Why does a conflict occur? In other words, what are the conflicting tasks and the goals behind them?
- What is the context in which a conflict occurs?
- Is there any alternative to avoid the conflict?
- What are the core conflicts that the system, at certain context, can not avoid? In other words, which conflicts are severe?

Most conflicts manifest themselves on a subject that is an object in the environment where the system operates [107]. In this thesis, we focus on two kinds of conflicts:

- Conflicting changes: this conflict happens when two or more system executable processes (tasks in a goal model) try simultaneously to change an 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 different states, *"closed"* and *"open"* respectively. If these two tasks execute in parallel, a conflicting change occurs.
- Exclusive possessing: this conflict happens when two or more executable processes need an exclusive possessing of an environment object. 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 possessing conflict occurs.

Detecting conflicts

To analyze the two kinds of conflicts that we have mentioned, we need to enrich contextual goal models with two kinds of information:

- The effect of tasks execution on the system operational environment: we need to specify explicitly the influence of tasks execution on the objects in the system environment. For each object that the system interacts with, we need to define if the execution of a task changes the state of that object or requires an exclusive possession on it. In Figure 4.8, we show the influence of some of Figure 4.1 tasks on the patient's home objects.

Object		States
External Doors	Balcony	{open, closed, locked}
	Main Entrance	
Windows	Living room	{open, closed, locked}
	Bed room	
Lights	Living Room	{on, off, medium}
	Bed Room	
	Balcony	
Siren, Security camera, Ventilator		{on, off}

Task	Object	State	Exclusive
T1	Home speakers		true
T5	Institute speakers		true
T2	Lights	{on}	
T3	External doors	{locked}	
	Windows	{locked}	
T4	Landline		true
T11			true
T8	Windows	{open}	
T5	Institute network		false

Figure 4.8: Objects in the patient's home (a), and the tasks impact on them (b)

- 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 obviously hard and time consuming activity. For this reason, we adopt the extension to goal model proposed in [108] where business process operators are introduced aiming at filling the gap between stakeholder goals and the business process to reach these goals. Out of these operators, we use the parallelism and sequencing operators to derive if two tasks may execute simultaneously. In Figure 4.9, we annotate the smart home contextual goal model, shown in Figure 4.1, with these two kinds of operators.

The algorithm reported in Figure 4.10 processes a contextual goal model and enriches its variants with information concerning adoptability and conflicts. The algorithm extracts the goal model variants having consistent contexts (Line 2-3). The goal model variants with inconsistent contexts are

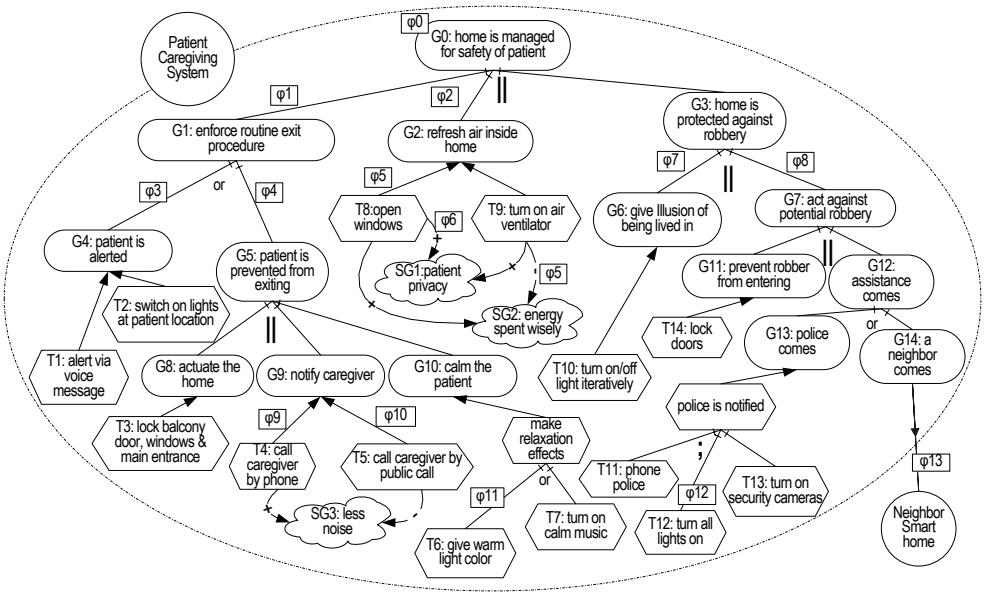


Figure 4.9: Goal model annotated with parallelism (||) and sequence (;) operators

excluded from further processing as they are unadoptable. Then each variant is checked for conflicts between its tasks (Line 6–14). The set of tasks belonging to each variant are 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). Consequently, by this reasoning we detect not only the conflicts between tasks but we also know the goals behind the tasks originating the conflicts and the context in which such conflicts happen.

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 crucially provided. In this section, we develop reasoning to discover when a conflict belongs to this kind, i.e. when it is core. We first give some basic definitions and then develop an algorithm processing a contextual goal model to detect core conflicts.

Input: *S: the set of all goal model variants*
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: $T := V.set\_of\_tasks$
7: $V.conflict\_set := \emptyset$
8: **while** $|T| > 1$ **do**
9: $t_i := pop\_element\_of(T)$
10: $T_{t_i\|} := \{t_j : t_j \in T \wedge in\_parallel(t_i, t_j)\}$
11: $T_{t_i\|conflicting\_changes} := \{(t_i, t_j, o, t_i.o.state, t_j.o.state) : t_j \in T_{t_i\|} \wedge o \in Environment\ Objects \wedge t_i.o.state \neq t_j.o.state\}$
12: $T_{t_i\|exclusive\_possession} := \{(t_i, t_j, o, \text{“}exclusive\text{”}) : t_j \in T_{t_i\|} \wedge o \in Environment\ Objects \wedge t_i.o.exclusive \wedge t_j.o.exclusive\}$
13: $V.conflict\_set := V.conflict\_set \cup T_{t_i\|conflicting\_changes} \cup T_{t_i\|exclusive\_possession}$
14: **end while**
15: **end if**
16: **end for**
17: **return** S

Figure 4.10: Detecting conflicts in contextual goal models

Definition 8 (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: $(\varphi_i \rightarrow \varphi_j) \wedge \neg(\varphi_j \rightarrow \varphi_i)$.*

From this definition, any variant that is not core has a set of core variants applicable in all contexts where it is itself applicable, but not vise versa. A reason for keeping non-core variants is that at certain context they might assure better quality[5]. The core variants are grouped on the base of the equivalence, either direct or under assumptions, of their contexts to construct core groups of variants.

Definition 9 (Core groups set)**.** *A core groups set is a set of core variants partitioned on the base of context equivalence.*

Definition 10 (Core group of variants)**.** *A core group of variants is an element of a core groups set.*

[5]The selection of non-core variants to keep is out of the scope of this thesis.

Example 13. In Figure 4.11, we show two partial goal model variants $\{V_1, V_2\}$. These two variants are two ways for satisfying the 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 not core since there is the variant V_2 that can replace V_1 in all the contexts where V_1 is applicable.

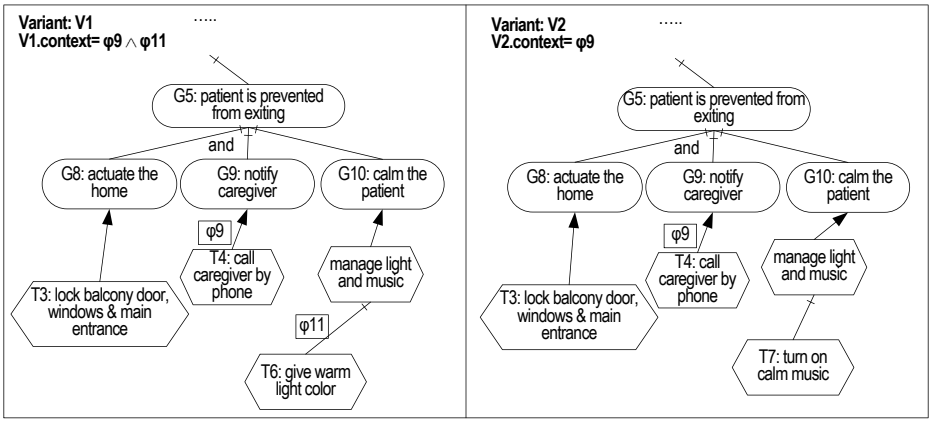


Figure 4.11: An example of a variant (V_1) that is not core

Having a conflict-free variant in a core group of variants means that any conflict in the other variants in the same group is not 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 11 (Conflictual core group of variants)**.** *A conflictual core group of variants is a core group of variants that does not include any conflict-free variant.*

The algorithm reported in Figure 4.12 extracts the core groups of variants in conflict from a contextual goal model. It calls the algorithm shown in Figure 4.10 to enrich each variant with information about adoptability and conflicts occurring in it (Line 1). The algorithm excludes the unadoptable variants, i.e., the variants with inconsistent contexts, as they are obviously not core (Line 2). The algorithm then extracts the core groups of variants (Line 4–11). To this end, the algorithm partitions the set of variants based on context equivalence (Line 6). The algorithm CheckSAT, shown in Figure 4.4, can be also used to check the equivalence between boolean formulae expressing contexts. Given the logical relations (implications) (ξ) between

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: $S' := S' \setminus \{V \in S : V.adoptability = \bot\}$
3: $S'' := \emptyset$
4: **while** $|S'| > 0$ **do**
5: $V := pop\_element(S')$
6: $temp := \{V\} \cup \{V' \in S' : CheckSAT(\neg(V.context \leftrightarrow V'.context)) = \bot\}$
{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** $\exists V \in U : V.conflict\_set = \emptyset$ **then**
14: $S'' := S'' \setminus U$
15: **end if**
16: **end for**
17: **return** S''

Figure 4.12: Extracting the conflictual core groups of variants

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 conflictual and it is excluded from the output set (Line 12–16).

Example 14. As shown in Figure 4.13, 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 the tasks T_3 and T_8 manifested on the environment object "windows". Each of these two tasks changes the state of this object differently as we specified in Figure 4.8. The 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 not 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.

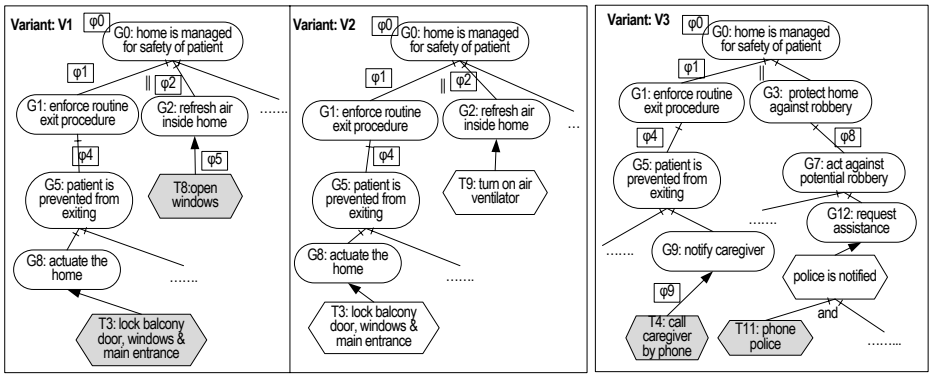


Figure 4.13: Non-core variant with conflict (V_1), its conflict-free alternative (V_2), and variant with core conflict (V_3)

4.2 Reasoning about Variants Derivation

A goal model hierarchy is a compact form that may incorporate a huge number of variants for goal satisfaction. A variant for goal satisfaction is an And-tree of the goal model hierarchy, i.e. a deterministic way to goal satisfaction. Choosing between these variants is a question that arises at two different stages:

- **Design time**: it is often desirable that software systems are provided with alternatives to reach users' needs. Having alternatives is desired for various reasons such as increasing the system ability to recover from errors by adopting alternative solutions and increasing the system variability to enable it to work in different contexts, and so on. From the other hand, developing a large number of variants could lead to various problems. For example, it could lead to more development costs, or it might lead to redundancy in software functionalities. In this thesis, we deal with the management of variants to get a system with a number of variants enabling it to reach user goals in all considered contexts and developed with minimum costs.

- **Runtime**: the introduction of variants to reach user requirements necessities a systematic selection between these variants at runtime as well. In this thesis, we focus on the selection between variants according to the context and user priorities. User priorities are expressed over softgoals, and the variant adopted is the one adoptable in the current context and that contributes more positively to the prioritized softgoals.

4.2.1 Running example

In this thesis, we use a case study of a museum-guide mobile information system developed within the Laboratory of Mobile Application (LaMA[6]) at the University of Trento[7]. The system is expected to enforce the museum rules by notifying visitors to what they should do in the right moment. Moreover, the system has to figure out if the visitor is interested in a certain piece of art and convey suitable information related to that piece of art. Visitors and museum staff are provided with PDAs as communication and explanation devices. The system consists basically of two components: the monitoring component that captures context, and the functional component that carries out actions reflecting each monitored context.

To initiate the process of conveying information about a piece of art to a visitor, the system has to monitor if the visitor is interested in it. This information can be inferred, for instance, if the visitor has been standing in front of the piece of art for long time. If so, the system has to look for the best way to convey information to the visitor. The delivery of information can be done via information terminals, the PDA the visitor has, or a staff member. For each of the possible ways to convey information, the system is supposed to do certain tasks. For example, to use terminals the visitor must be informed about the existence of such a service, guided to it, and informed about the way to use it. To get information through a staff member, the system has to notify the staff member and establish a call with the visitor, or guide the staff to the visitor's place to give information in person.

Concerning the relationship between context and requirements, context can influence decisions about:

- **Requirements to meet**: if the context "visitor is not interested in a piece of art" applies, the mobile information system does not need to activate the information delivery process. Moreover, if the context "visitor is familiar with the use of terminals and knows one of the languages the terminals support" applies, then informing the visitor about the way of using such terminals is not required and the system has only to inform the visitor about the existence of the service and guide him to a free terminal.

- **Ways to meet requirements**: the system could have two variants to convey information about a piece of art via PDAs: video-based and interactive. Each variant could require a valid context. For example,

[6]http://lama.disi.unitn.it/

[7]In Chapter 6, we will process the entire model of the mobile information system

conveying information via an interactive presentation requires that a context like "visitor has good experience in using PDAs" applies.

- **Quality of each way**: considering staff comfort as a quality measure; conveying information to visitors on person is less comfortable for a staff when a context like "visitor is far away from the staff" applies.

In Fig. 4.14, we show a partial goal model of the museum guide system, and in table 4.2, we give a brief description of the contexts annotated at its variation points.

4.2.2 Deriving variants for varying contexts

Goal models support variants to goal satisfaction. The selection of the variants to include in the system-to-be is a design decision that could be based on different criteria. For example, the decision could be based on minimizing the development costs as we propose in the next section. However, if we want the system to be flexible and highly variable, the developed system should support multiple variants to goal satisfaction [109]. When the system is supported by multiple variants, it may find more than one applicable variant at certain contexts at runtime. Therefore, the system will need a criteria to decide what variant to adopt. One criteria could be the user's prioritization over the variants. Asking users to specify their prioritization over the goal model variants directly has two main difficulties:

- the potentially huge number of model variants, i.e., specifying prioritizations over the enumerated variants could be extremely time consuming.

- when the variants contain a large number of nodes, it could be hard for users to comprehend the variants and the differences between them.

Instead of asking users to provide prioritization over the variants themselves, prioritization can be expressed over the quality measures, i.e., softgoals. Users can express prioritization on softgoals and bypass the large number of goal model variants. Besides avoiding us dealing with the large number of enumerated variants, softgoals allow users to express prioritization using their own terms. For example, users can easily state that *"more comfort"* has high priority while *"less disturbance"* is not such important. The quality contexts of a variant are those between each softgoal and that variant. The truth value of quality contexts determines the quality of each

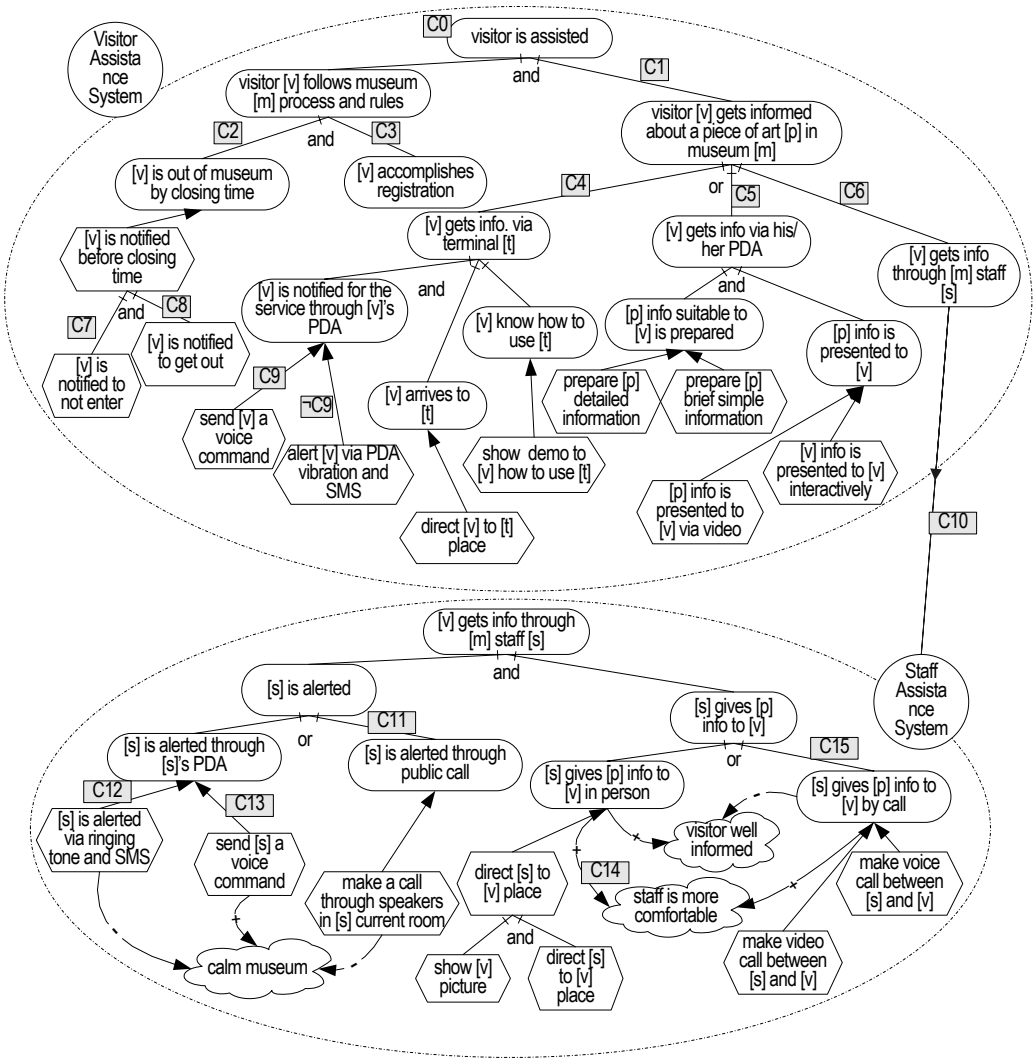


Figure 4.14: Partial contextual goal model for the Museum-guide system

variant. In other words, a variant can be of different qualities in different contexts.

We adopt an approach similar to the one proposed in [92] to specify prioritization over softgoals. We consider binary contributions to softgoals (positive or negative). Stakeholders can specify the priority of each softgoal by selecting an integer in the range $[0,n]$. Priority 0 corresponds to *"the user does not care about the softgoal"*, priority n means *"the user considers the soft-goal very important"*. The priority of a variant is computed by the formula:

	Description
C_0	the visitor has to be inside the museum area including parking places and the public square in front of museum, and moreover, the visitor should have accepted the autonomous assistance by the mobile information system
C_1	the visitor is interested in getting explanation about the piece of art, and he is in the gallery building
C_2	closing time is approaching
C_3	visitor has entered the museum building
C_4	terminal is free and close to the visitor and he/she is able to use and interact with it
C_5	the piece of the art information are not so complicated, and the visitor has the ability and the knowledge to use PDAs
C_6	the visitor is not able to use PDA and not familiar with terminals, or that the visitor is classified as an important visitor
C_7	the visitor is on the way to enter the museum building
C_8	the visitor is still inside the museum building and is not walking towards the exit
C_9	the visitor puts the headphones on, and is not talking to somebody or using his PDA for a call
C_{10}	there is a staff that is free and talks a language common to the visitor, and knows enough about the considered piece of art comparing to the visitor knowledge
C_{11}	the room does not include audio art contents
C_{12}	staff is not calling
C_{13}	staff is not calling and not putting the headphone on
C_{14}	if the visitor is close to the staff
C_{15}	the staff's PDA is not being used for a call

Table 4.2: The descriptions of context specified at the contextual goal model of Fig 4.14

$$priority(v) = \sum_{sg \in v} percentPos(v, sg) \times priority(sg) - \sum_{sg \in v} percentNeg(v, sg) \times priority(sg)$$

The function $percentPos(v, sg)$ ($percentNeg(v, sg)$) refers to the percentage of the positive (negative) contributions with respect to the total number

of contributions from the variant v to the softgoal sg. We use the percentage to uniformly deal with softgoals with disparate numbers of contribution links. Every contribution link is treated as an evidence about the positive or negative satisfaction of a softgoal. Consequently, the derivation of goal model variants for a given context and user prioritization is a two steps process that the system follows at runtime:

1. *Deriving the variants applicable in the current context*: the truth values of contexts at the variation points decide the set of goal model variants that are applicable. As we have shown earlier, context analysis allows us to discover a formula of facts that specifies a context (see Fig. 4.2). The system, at runtime, has to monitor the environment and collect data (Fig. 4.3) and compute the truth value of the formulae of facts at each variation point of the goal model. This, in turn, filters the space of goal model variants leaving those that are applicable in the current context.

2. *Ranking the applicable variants based on user's prioritization*: at certain contexts, there could be more than one applicable goal model variant. In other words, there could be more than one variant to meet the same requirements. To select between them, user prioritization could be considered by the system at runtime. To this end, users are asked, at design time, to prioritize the set of softgoals. The system computes the value of contextual contributions and the priority of each applicable variant according to the formulae above. The adopted variant is the one with the highest priority, i.e., the one that better contributes to the highly prioritized softgoals.

Example 15. Suppose that the current context allows for the two model variants partially shown in Fig. 4.15. The system has the possibility to guide a staff to meet a visitor in person (variant V') and the possibility to establish a call between them so as to communicate remotely (variant V''). Delivering information in person to a visitor contributes negatively to the softgoal *"staff feels more comfortable"*, as the staff is not close to the visitor (presuming that C_{14} is false), and positively to the softgoal *"visitor is well-informed"*. The second alternative variant, delivering the information by a remote call, contributes conversely to the two mentioned softgoals. If the museum adminstration gives staff comfort a priority higher than the quality of information delivered to visitors, then the variant V' would be adopted, and vice versa.

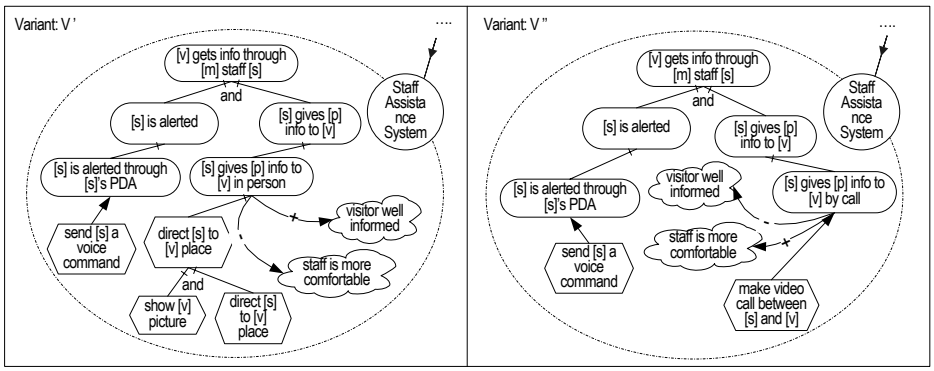


Figure 4.15: Variants of different qualities

4.2.3 Deriving variants for minimum costs

In the previous section, we have studied the derivation of goal model variants for a given context and user priorities. Such reasoning is of high importance for systems that support multiple goal model variants and where more than one variant is adoptable in certain contexts. On the other side, and for reasons such as budget and timing constraints, we may want a system developed with minimum costs sacrificing the quality and flexibility gained by supporting the whole set of goal model variants. In other words, the system has to support a set of variants that is enough to meet users' goals in all considered contexts and developed with minimum costs. To this end, we have developed a reasoning in three steps to be used at design time: (i) we exclude the variants that are unadoptable because of unsatisfiability in their contexts; (ii) we exclude the variants that can be always replaced by others; (iii) and finally, we reason about the remaining variants to extract those leading to a system developed with minimum costs and that is able to meet user goals in all analyzed contexts.

Deriving the unadoptable goal model variants

A goal model variant can be unadoptable as a consequence to the inconsistency of its context. We need to check such inconsistency early to save costs and fix errors given that unadoptable variants may lead to software parts that are never used or incorrectly specified. As we have shown in Section 4.1.2, inconsistency sometimes originates from the compact form of goal model hierarchy. The compact form of goal models could lead to a huge number of variants, that some of them are indeed unadoptable and their contexts inconsistency can be accepted. I.e. inconsistency is not always a

modeling error and a design decision has to be taken either to fix it or to accept the exclusion of a variant. Here we give another example of a variant with inconsistent context.

Example 16. The variant shown in Fig. 4.16 has an unsatisfiable context due to the contradiction between C_7 ("the visitor is on the way to enter the museum shortly before the closing time"), and C_1 ("the visitor is in the gallery building and interested in getting explanation about a piece of art"). A design decision has to be taken to accept this kind of unsatisfiability, i.e. to confirm that the model variant is indeed not needed, or to modify the model and fix it. In fact, and in this particular example, the unsatisfiability is not a modeling error but it is a side-effect of the goal model hierarchy. This hierarchy compactly represents a large number of variants in one model and it, at the same time, may include variants that are never applicable. The tasks of the unadoptable variants, such as the variant of our example, could appear in other variants with satisfiable contexts and, therefore, these tasks are not necessarily unusable if implemented in the final system. A task could be implemented in the system-to-be if it appears in, at least, one goal model variant with a satisfiable context.

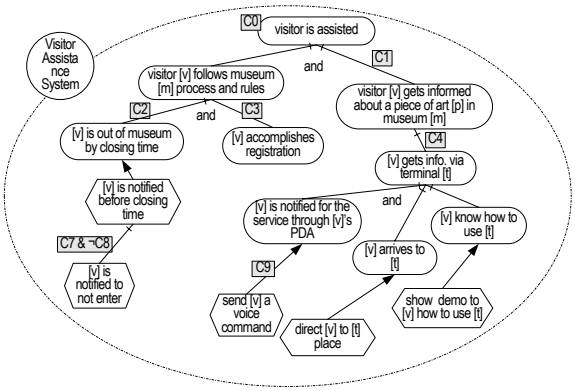


Figure 4.16: Goal model variant with an inconsistent context

Deriving the (non-)core goal model variants

Core requirements are system requisites that can not be bargained on. There could be different perspectives to categorize requirements into core and non-core. Concerning a system supported by variants to operate in and reflect varying contexts, the variants having no alternative variants at certain contexts are core. Discovering core variants is useful for several reasons. It helps

to know the parts of the system that are critical and whose failure can not be remedied by adopting other variants at certain contexts. Also, it helps to know the part of the system that needs to be developed first and can not be delayed to get a system operable in all considered contexts. The latter reason is the focus of this thesis.

In Section 4.1.3, we have given the definition of Core Variants and Core Group of Variants. We have also developed a reasoning about a contextual goal model that leads to derive them. A reason for keeping the non-core variants is that, in some contexts, they could lead to better satisfaction of quality measures than the core variants, i.e. better satisfaction of the softgoals. The discovery of these core groups of variants can give the analyst more information of the requirements that need more attention and more priority when there is a need to implement the core part of the system first because of timing or budget constraints. Moreover, these core groups of variants can be further processed to select a variant from each group seeking for a system developed with minimal costs. Here, we give another example of the non-core variants taken from the museum guide system:

Example 17. In Fig 4.17, we show two partial contextual goal model variants $\{V_1, V_2\}$ each including a different set of tasks to implement. Both contexts of the two variants are satisfiable and $V_2.context \rightarrow V_1.context \wedge \neg(V_1.context \rightarrow V_2.context)$. This means that V_2 is non-core since there is always the variant V_1 that can replace it in all considered contexts. In the space of these two partial variants, the task *"send [s] a voice command"* and *"make voice call between [s] and [v]"* are non-core, while the tasks *"[s] is alerted via ringing tone and SMS"*, *"show [v] picture"*, and *"direct [v] to [s] place"* are core and essential to implement in order to achieve the goal *"[v] gets info through [m] staff [s]"* in all considered contexts.

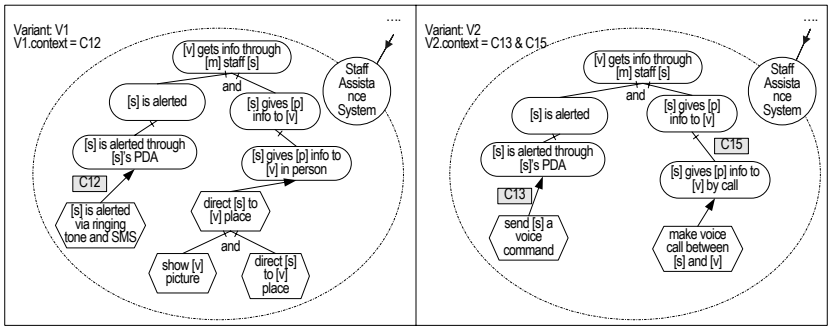


Figure 4.17: Non-core variant (V_2)

Deriving the variants for minimal costs system

Developing a system that supports multiple variants to reach its requirements is desirable for several reasons such as flexibility and fault tolerance. In a previous section, we have shown how such approach can accommodate the priorities of different users. For different reasons, such as timing and budget constraints, we may be required to develop just an operable system, i.e. a system that operates in all considered contexts. In this section, we develop the final step of the reasoning about a contextual goal model to derive a subset of its leaf tasks that leads to a system able to operate in all considered contexts and developed with minimum costs. These tasks may not implement the whole set of goal model variants, but those that are implemented will allow the system to reach its goals in all considered contexts.

Costs are not related to goals but to tasks as tasks represent executable processes while goals are just desires of an actor. Each task needs certain development resources (equipments, programmers, software packages, and so on). Each of these resources has a cost. We need to specify the resources needed for each task development and the costs of each resource to enable our target reasoning. A resource may be a part of the development of multiple tasks which means that the development costs of tasks may overlap. For example, both of the tasks *"direct visitor to terminal location"* and *"direct staff to visitor location"* need almost the same resources. They both need a positioning system, communication system, and preparing a digital map of the museum. The development of the two tasks *"piece of art information is presented to visitor via video"* and *"piece of art information is presented to visitor interactively"* share the resources of gathering data about the pieces of art and preparing pictures, videos, and audio explanation to be presented, and programming the presentation.

Defining the resources needed for each task and the costs of these resources is the basic step to decide which tasks to develop. The second step is getting the core groups of variants of the contextual goal model (we have already explained this reasoning). Then we need to identify a subset of tasks that implements, at least, one variant of each core group of variants targeting for a minimal total cost.

Definition 12 (Operable set of tasks)**.** *S is an operable set of tasks for a core groups set CS iff for each $CG \in CS$, $\exists v \in CG \wedge v.tasks \subseteq S$*

Definition 13 (Min-cost set of tasks)**.** *S is a min-cost set of tasks iff S is an operable set of tasks and $\nexists$ another operable set of tasks S' with lower development costs.*

A naive approach to extract a min-cost set of tasks can be to compute the cartesian product of the core groups of variants and then selecting the combination of variants of minimum cost. Such approach is obviously time consuming and suffers of exponential blow-up. Moreover, our experiments evidenced that it can not deal even with small-medium size goal models. Thus, we need to replace the naive approach with an optimized algorithm. We can significantly reduce the complexity of our reasoning by exploiting the nature of the problem as shown in the algorithm reported in Fig. 4.18. First, the algorithm calculates the set of tasks that are mandatory for all possible combinations of variants (Lines 1–4). A task is mandatory if it is included in all the variants of, at least, one core group of variants. To reduce the number of core groups of variants to be involved in the cartesian product, the algorithm makes two processing and produces a reduced core groups set. A core group that includes, at least, a variant implementable using a subset of the mandatory tasks will be excluded (Line 5–9). Some core groups of variants become equivalent after excluding the mandatory tasks of the variants belonging to them and we unify such equivalent groups to reduce the number of core groups that will be included in the forthcoming cartesian product (Line 10–11). The rest of the algorithm deal with the cartesian product of the core groups of variants belonging to the reduced core groups set (S) and returns the min-cost set of tasks (Lines 12–14).

Example 18. In Fig. 4.19, we show a part of the goal model shown in Fig. 4.14. We provide estimations for the costs of each task development aside. We show the set of variants after excluding the non-core variants as we explained in the last section. The remaining variants are grouped based on context equivalence to create core groups of variants. The relation between tasks based on the shared resources are reported. Include(T_1, T_2): the work done to gather simple information of the pieces of art is included in that needed for gathering more detailed information. Intersect(T_3, T_4, A): the interactive presentation (T_4) includes videos (the resource A) that are also needed for video-based presentation (T_3). Intersect(T_3, T_5, B), Intersect(T_4, T_5, B): all these tasks need a server and PDA for communication (the resource B). Intersect (T_4, T_8, C): we presume that T_8 is interactive which means that both of T_8 and T_4 require PDA with touch screen and the corresponding programming packages for getting user input in this way (the resource C). After this specification, we show the set of tasks to develop and the variant that are implemented on them and the final minimized costs.

Input: S: *core groups set*
Output: T: *a min-cost set of tasks for* S

1: $MT := \emptyset$ $\{MT$ *stands for mandatory tasks*$\}$
2: **for all** $CG \in S$ **do**
3: $\quad MT := MT \cup \{\bigcap\{v.tasks : v \in CG\}\}$
4: **end for**

5: **for all** $CG \in S$ **do**
6: $\quad$ **if** $\exists v \in CG : v.tasks \subseteq MT$ **then**
7: $\quad\quad S := S \setminus \{CG\}$
8: $\quad$ **end if**
9: **end for**

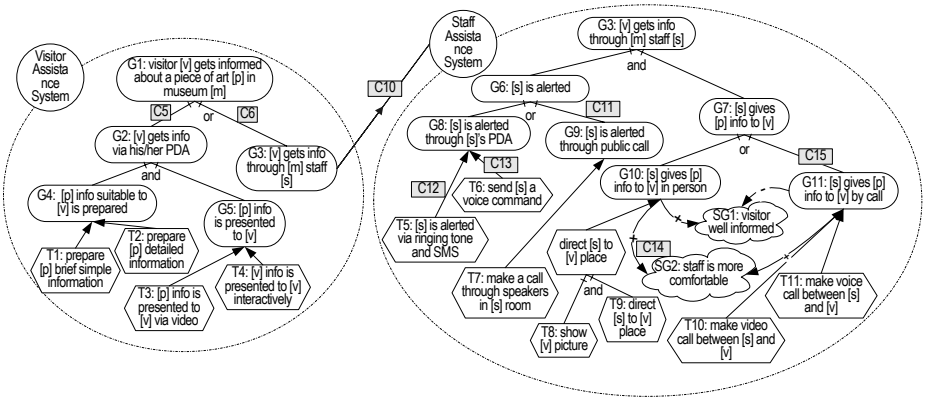
10: $S := Exclude\_Mandatory\_Tasks(S, MT)$
11: $S := UnifyEquivalent(S)$

12: $P := \{\{S.CG_1.variants\} \times \ldots \times \{S.CG_n.variants\}\}$
13: $mincost := p \in P : costs(p.tasks \cup MT)$ *is minimum*
14: **return** $mincost.tasks \cup MT$

Figure 4.18: Extracting the min-cost set of tasks.

4.3 Chapter Summary

In this chapter, we have proposed a set or reasoning mechanisms to detect modeling errors in a contextual goal model and to manage the derivation process variants to goal satisfaction. The validation was to discover inconsistency in the contexts of each variant and for discovering cases when the parallel execution of two executable processes in one variant can lead to a conflict. We have provided reasoning to support the derivation of variants at design and runtime. For design time, we developed a reasoning that allows the derivation of variants ensuring a system valid in all considered context and developed with minimum costs. For runtime derivation, we developed a reasoning for choosing the variants adoptable in a given context and compliant with user priorities.



The non-core variant	The variants excluding the non-core variants	The core groups of variants	The cost relations	The min-cost core requirements
NV1= {T6, T10} **NV2**= {T6, T11} Both can be replaced by **V2** due to the implications: C13→C12 and the trivial C15 → true.	**V1**= {T1, T3} **V2**= {T1, T4} **V3**= {T2, T3} **V4**= {T2, T4} **V5**= {T5, T8, T9} **V6**= {T7, T8, T9}	**Core1**= {V1, V2, V3, V4} **Core2**= {V5} **Core3**= {V6}	**Cost**(T1,30) , **Cost**(T2,40), **Cost**(T3,60) , **Cost** (T4,80), **Cost**(T5,25) , **Cost**(T6,35), **Cost**(T7,50) , **Cost** (T8,30), **Cost** (T9,50) , **Cost** (T10,50), **Cost** (T11, 30). **Include**(T2, T1), **Intersect**(T3, T4, 40), **Intersect** (T3, T5, 20), **Intersect**(T4, T5, 20), **Intersect** (T4, T9, 30) **Cost of developing all tasks= 340**	**The tasks to develop**= {T1, T4, T5, T7, T8, T9} **Costs=** 215 **The variants implemented:** { V2, V5, V6}

Figure 4.19: Example for minimum-cost variants extraction.

Chapter 5

Automated Support Tool and Methodological Process

In this chapter, we develop an automated support tool and a methodological process to help analysts for constructing and reasoning about contextual goal models. RE-Context is the tool we developed in order to provide automated reasoning about contextual goal models. The tool is a prototype: the algorithms proposed in this thesis are fully implemented, but it does not provide a graphical user interface yet. Moreover, some of the algorithms are naively implemented; these algorithms can certainly be improved and optimized. Throughout this chapter we provide a detailed description of RE-Context. First, we describe its architecture, showing how it integrates with state-of-the-art reasoning tools. Second, we describe how the algorithms proposed in this thesis are implemented. Third, we explain the input format that the tool requires. We finally show a systematic analysis process to capture and reason about contextual goal models.

5.1 RE-Context: Automated Support Tool

5.1.1 Architecture

We describe here the logical architecture of RE-Context. Figure 5.1 presents a schema depicting the structure of RE-Context. It's easy to notice that RE-Context is not a monolithic program, but it is rather a combined system that brings together three main components: a Java program, DLV[1] and MathSat4[2].

[1] http://www.dbai.tuwien.ac.at/proj/dlv/
[2] http://mathsat4.disi.unitn.it/

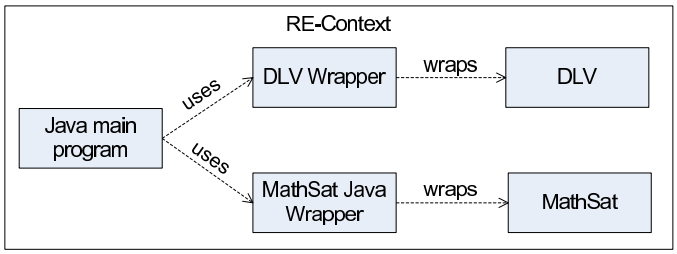


Figure 5.1: Logical architecture of RE-Context.

The core part of RE-Context is developed using Java. The Java program includes many algorithms and exploits the functionalities of the automated reasoning tools DLV and mathsat in order to carry out its functions. In particular, the Java code of RE-Context uses two wrappers to effectively interact with the automated reasoning tools: the DLV-Wrapper[3] and a mathsat Java wrapper provided by the authors of mathsat.

Let's examine more in detail how DLV and mathsat are used in RE-Context:

- DLV is used for variants generation and to compute the minimum development cost. During variants generation DLV computes not only the set of tasks and contexts for every variant, but also contribution to soft-goals and conflicts regarding the usage of resources.

- MathSat is used to perform checks concerning context, namely to identify inconsistent variant (if the context formula is inconsistent), to detect equivalent contexts, and to verify implications between contexts.

5.1.2 Functionality

In this section, we describe the functionalities that RE-Context provides. Figure 5.2 summarizes the main steps that RE-Context carries out; the diagram shows the activities connected by input/output flows.

Variants generation. The first step that should be carried out is the generation of the variants starting from a contextual goal model. In order to carry out this activity, RE-Context interacts with both mathsat and dlv. First, the mathsat environment should be initialized; initialization takes place just once. Then, the mathsat rules that represent relations between contexts —e.g. implications, equivalences, negated implications—are loaded from file `ass.msat`. Then, the contextual goal model and a set of auxiliary rules are

[3]http://www.mat.unical.it/wrapper/index.html

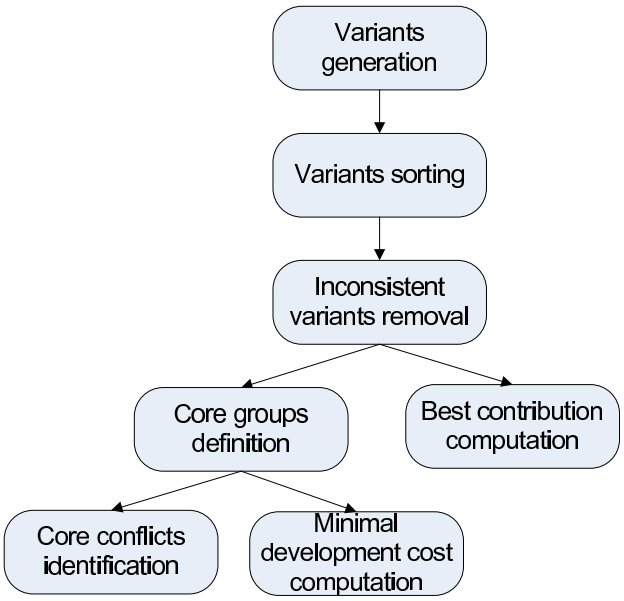


Figure 5.2: The algorithms in RE-Context linked by input/output flows.

loaded from two files in the DLV format: `vargen.dlv` and `rules.dlv`. The rules are used to identify conflicts, to propagate information about sequencing of goals top-down, to compute contribution to soft-goals. After these loading activities are completed, DLV is executed in order to derive all existing variants. DLV outputs all the models that satisfy a set of rules and constraints; here, every model is a variant. DLV is executed from Java using the DLV-Wrapper interface: this allows for iterating through the models in a simple way. Every model is examined and the information describing the variant is stored to memory; the retrieved information consists of the tasks, the contexts, the conflicts, and the contributions.

Variants sorting. The generated variants are sorted based on the number of tasks (from the smallest to the biggest). The sorting is executed using Java sorting operations on collections. This step is performed in order to boost the performance of the tool when dealing with inconsistent variants. Indeed, if a smaller variant is inconsistent, then also a superset of that variant (i.e., a variant including all the tasks of the inconsistent variant) will be inconsistent. In such a way, it is possible to mark as inconsistent all the supersets of that variant. The ordering is necessary to maximize the effectiveness of this strategy. Suppose we have three variants (let's consider only tasks here) $V_1 = \{t_1, t_2, t_3, t_4\}$, $V_2 = \{t_1, t_2, t_3\}$, $V_3 = \{t_1, t_2\}$ and there is an inconsistency V_3 context. Without ordering, this would require three consistency checks; with ordering, one consistency check would be sufficient.

Inconsistent variants removal. This activity identifies the inconsistent variants and removes them from the set of adoptable goal model variants. Variants removal is not mandatory: as shown earlier in the thesis, inconsistencies might be fixed by the analyst and inconsistency check repeated. The inconsistency of one variant is verified via mathsat: all the contexts of the considered variant are put together as a conjunction and the SAT solver is run to verify if there is a model (truth assignment) for that formula. If there is a model, then the variant is consistent; otherwise, it is inconsistent. The set of variants is analyzed iteratively and exploiting the principle described before: if an inconsistency is found, all the following variants are compared to the current one on the basis of the tasks that compose the variant. If a variant is a superset of an inconsistent variant, it is marked as inconsistent as well. While iterating through the set, if a variant is already marked as inconsistent it needn't be examined using mathsat. During the inconsistency identification process, RE-Context can be configured to discover the kind of context the inconsistency occurs in. After inconsistency identification is completed, inconsistent variants are removed from the set of variants.

Ranking variants. An activity that is performed after removing inconsistent variants is to compute the variant having the highest priority, i.e., the best contribution to the prioritized soft-goals. This activity is quite simple and consists of a sequential scanning of all the consistent variants. The auxiliary DLV rules allow for the computation of the contribution to a single soft-goal; RE-Context exploits Java to compute the weighted sum of the contributions and compare the value for the current variant against the maximum value obtained so far.

Core groups identification. Starting from the consistent set of variants it is possible to define the core groups of variants. The algorithm implemented in RE-Context is based on three sequential steps.

1. *syntactic* grouping consists of scanning all the variants and grouping them into sets according to their syntactic equivalence. By syntactic equivalence we mean that their contexts are exactly the same, namely the two sets of contexts contain the same elements. This first iteration allows for a first grouping without the execution of external tools (which are slower).

2. *semantic* grouping is intended to reduce the number of core groups by performing equivalence checks using mathsat. The core groups obtained after syntactic grouping are compared one to another by checking the equivalence of their first variants. For instance, let $context(CG_1) = \{c_1, c_2\}$ and $context(CG_2) = \{c_1, c_3\}$; let the contexts be $c_1 =$

$a \wedge b$, $c_2 = b \wedge c$, $c_3 = d$; let the following relation between variable hold $(b \wedge c) \leftrightarrow d$. This assumed relation can be absolute or dependent on the operational environment of the system as we have already explained in Chapter 4. The context of core group CG_1 and CG_2 are not syntactically equivalent, but they are equivalent under the assumed relations.

3. *non-core* variants removal is in charge of removing those core groups that are non-core ones. This check is performed by RE-Context by invoking mathsat. The algorithm works by iterating through all core groups and checking the implications between their context to filter those that can be replaced by others in all considered contexts.

Core conflicts identification. In our framework, conflicts are defined with respect to the usage of resources. The conflicts of each individual variant have already been computed by DLV during variants generation. Therefore, the identification of core conflicts reduces to the analysis of every core group and check if all variants in that core group have a conflict. If it is the case, that core group is conflictual. If there is at least one conflict-free variant, that core group is not conflictual.

Minimal development cost computation. The purpose of this activity is to find a set of tasks that allows for a system operable in all analyzed context and developed with minimum costs. This corresponds to implementing, at least, one variant from each core group; so, the naive algorithm tries all combinations of variants. Clearly, this approach is very expensive computationally. RE-Context applies some optimization techniques that significantly improve the performance in practice. The algorithm proceeds in three steps:

1. all core groups are sequentially scanned in order to define *shared tasks*. Shared tasks are those tasks that are mandatory for a core group, regardless of the chosen variant. For instance, if $CG_1 = \{V_1, V_2\}$, $V_1 = \{t_1, t_2, t_3, t_4\}$, and $V_2 = \{t_1, t_3, t_5, t_6\}$, the set of shared tasks for this core set is $\{t_1, t_3\}$. The shared tasks of all core groups are added to a set which represents the set of tasks that have to be necessarily implemented.

2. All the core groups are scanned and, for every variant, the overall shared tasks are removed. If there is a variant with zero tasks remained then the core group is removed from further processing.

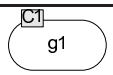
3. The naive algorithm (pick one variant per core set) is now applied to the reduced set of core groups. In theory, the computation might

be still very hard; in practice, we have noticed that the preprocessing dramatically speeds up the process. We exploit DLV to generate all possible combinations of the variants in the remained core groups.

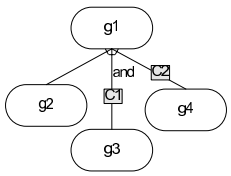
5.1.3 Input Format

We describe here the input format for RE-Context, which allows for specifying a contextual goal model, information about task development costs and resource conflicts, contexts and relations between contexts. The input format expresses this information in a very low-level; nevertheless it is not difficult to generate a graphical editor and implement a translation engine.

The first chunk of input information to be encoded regards the contextual goal model. We provide below the guidelines that show how different constructs can be mapped to the RE-Context format. The input format for contextual goal models is DLV based, therefore it consist of a set of inference rules of the type *head* :- *tail*. Given that we exploit disjunctive datalog, *head* is a disjunction of predicates and atoms separated by the "v" symbol, whereas *tail* is a conjunction of predicates and atoms separated by the "," symbol.

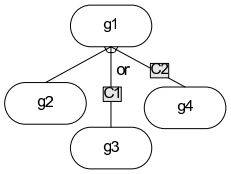
Goal activation		ach(g1) :- c(1).

Goal activation is represented by a single rule which says that, if the context is true, then the goal should be achieved. When executing variants generation, this rule requires also to assume that the activation context is true. Anyway, an activation context set to false would produce no variant at all.

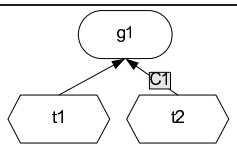
And-decomposition		achFrom(g2,g1) :- ach(g1). achFrom(g3,g1) v -c(1,s) :- ach(g1). c(1,s) :- achFrom(g3,g1). achFrom(g4,g1) v -c(2,s) :- ach(g1). c(2,s) :- achFrom(g4,g1).

Contextual AND-decomposition is mapped to more than one rule. In particular, a non-contextual branch requires one rule, whereas a contextual branch requires two rules. In the example, the non-contextual branch from g_1 to g_2 is represented by the first rule, which states that if g_1 should be achieved, then g_2 also has to be achieved. The binary predicate *achFrom* replaces the unary predicate *ach* to keep track of the goal tree structure. The contextual branch from g_1 to g_3 is represented by the second and third rules. The second rule states that, if g_1 has to be achieved, then either g_3

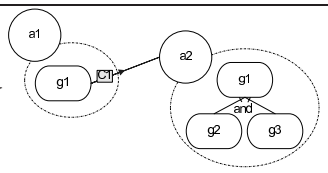
must be achieved or the activation context c_1 must be false. The third rule says that if g_3 must be achieved, then the activation context c_1 must be true.

Or-decomposition		achFrom(g2,g1) v achFrom(g3,g1) v achFrom(g4,g1) :- ach(g1). c(1,r) :- achFrom(g3,g1). c(2,r) :- achFrom(g4,g1).

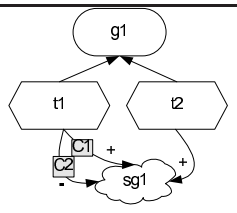
Contextual OR-decomposition is modeled via two types of rules. One rule deals with non-contextual decomposition: the first rule in the example states that, if g_1 must be achieved, then one among g_2, g_3 and g_4 must be achieved. Then, every contextual branch needs a clause saying that the associated context should be true. The second rule says that if g_3 has to be achieved, then the required context c_1 must be true.

Means-end		todoFrom(t1,g1) v todoFrom(t2,g1) :- ach(g1). c(1,r) :- todoFrom(t2,g1).

Means-end decomposition is represented similarly to OR-decomposition. The only difference is the usage of the predicate *todoFrom* instead of *achFrom*. The reason for using this predicate is that RE-Context deals with tasks while performing the forthcoming reasoning, therefore the tool can filter the predicates of interest and get rid of auxiliary ones.

Dependency		c(1,s) :- ach(g1). achFrom(g2,g1) :- ach(g1). achFrom(g3,g1) :- ach(g1).

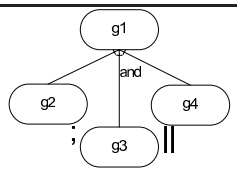
Contextual dependencies from one actor to another one is modeled in a simple way. Indeed, contextual dependencies become a further constraint. In the example, the first rule states that if g_1 has to be achieved, then the activation context c_1 must be true. The delegated goal is then decomposed as a non-delegated goal.

Contribution		softgoal(sg1,4). contrib(t1,sg1,pos) :- c(1,r). contrib(t1,sg1,neg) :- c(2,r). contrib(t2,sg1,pos).

The modeling of contribution to soft-goals requires additional information

about soft-goals. Indeed, every soft-goal should be given a weight from 1 to 5, where 5 means that the soft-goal is very important. The first rule in the example says that sg_1 has priority 4. Contextual contribution is shown in the second and third rule: the contribution of t_1 to sg_1 is positive if context c_1 holds; the same contribution is negative if c_2 holds. Non-contextual contribution is represented as in the fourth rule, saying that the contribution from t_2 to sg_1 is always positive.

Sequentiality and parallelism



sequence(g2,g3).
parallel(g3,g4).

Sequentiality and parallelism constraints between goals and tasks are represented in the input format of RE-Context using two binary predicates: *sequence* and *parallel*, respectively. In the example, the first rule says that g_2 and g_3 should be executed sequentially; more precisely, that g_2 should precede g_3. The second rule says that g_3 and g_4 are to be executed in parallel.

A second piece of information that is represented using datalog rules is that concerning the usage of resources. RE-Context supports two main types of information concerning resource usage:

1. *Exclusive possession*: this kind of information permits to express when a certain resource cannot be used by more than one task in parallel. For example, the following code says that resource x cannot be used simultaneously and that tasks $t1$ and $t2$ require it. If $t1$ and $t2$ are in parallel, then this will produce a conflict; otherwise there will be no conflict.

   resource(x).
   requires(t1,x).
   requires(t2,x).
   exclusiveUsage(x).

2. *Conflicting changes*: this conflict arises when two tasks change the status of a resource to different values in parallel. In the following example, resource x is being changed by parallel tasks $t1$ and $t2$. The first task changes its value to $v1$, the second task to $v2$.

   resource(x).
   changes(t1,x,v1).
   changes(t2,x,v2).
   parallel(t1,t2).

actuationConflict(R,G1,G2) :- resource(R), changes(G1,R,V1), changes(G2,R,V2), G1!=G2, V1!=V2, todo(G1), todo(G2), not sequence(G1,G2), not sequence(G2,G1). exclusivePossession(R,G1,G2) :- resource(R), exclusiveUsage(R), requires(G1,R), requires(G2,R), G1!=G2, todo(G1), todo(G2), not sequence(G1,G2), not sequence(G2,G1).
parallel(A,B) :- parallel(B,A). parallel(A,C) :- parallel(A,B), parallel(B,C). sequence(A,C) :- sequence(A,B), sequence(B,C). sequence(A,B) :- achFrom(A,C), achFrom(B,D), sequence(C,D), not parallel(A,B). sequence(A,B) :- todoFrom(A,C), todoFrom(B,D), sequence(C,D), not parallel(A,B). sequence(A,B) :- achFrom(A,C), todoFrom(B,D), sequence(C,D), not parallel(A,B). sequence(A,B) :- todoFrom(A,C), achFrom(B,D), sequence(C,D), not parallel(A,B).
contribTo(S,V,pos) :- softgoal(S,Priority), ♯ int(Priority), Priority!=0, ♯int(V), ♯int(V1), V1=♯countTask : contrib(Task,S1,pos), todo(Task), S1=S, ♯int(V2), V2=♯countGoal : contrib(Goal,S2,pos), ach(Goal), S2=S, ♯int(V3), V3=V1+V2, V=V3*Priority. contribTo(S,V,neg) :- softgoal(S,Priority), ♯int(Priority), Priority!=0, ♯int(V), ♯int(V1), V1=♯countTask : contrib(Task,S1,neg), todo(Task), S1=S, ♯int(V2), V2=♯countGoal : contrib(Goal,S2,neg), ach(Goal), S2=S, ♯int(V3), V3=V1+V2, V=V3*Priority.
ach(G) :- achFrom(G,_). todo(T) :- todoFrom(T,_).

Table 5.1: Auxiliary datalog rules that preprocess the RE-Context input.

A third chunk of information represented using datalog is that expressing development costs. The cost of a task depends on the development resources that are needed to implement that task: humans, computers, rooms, software tools, and so on. Some development resources are in common to more than one task; therefore, the input for RE-Context is based on the cost of development resources. In the example below, task t_1 is split into the development resources a and b; task t_2 is split into b and c. Hence, the two tasks share the development resource b. The cost of a is 10, the cost of b is 20, the cost of c is 30.

```
todoGroup(a) :- todo(t1).
todoGroup(b) :- todo(t1).
todoGroup(b) :- todo(t2).
todoGroup(c) :- todo(t2).
cost(a,10) :- todoGroup(a).
cost(b,20) :- todoGroup(b).
cost(c,30) :- todoGroup(c).
```

A number of auxiliary datalog predicates is necessary to preprocess the input before it is examined by RE-Context. These auxiliary rules are shown

in Table 5.1 and described below:

- The first two rules are ternary predicates that indicate actuation and exclusive possession conflicts, respectively. The rules are expressed in such a way that the *requires* and *changes* predicates are considered only if the corresponding task should be done.

- The second set of rules propagates top-down information concerning parallelism and sequentiality: (i) parallelism is symmetric; (ii) parallelism is transitive; (iii) sequentiality is transitive; (iv) two goals A and B are sequential if their parent goals are sequential and the there is no explicit parallelism between A and B; the following three rules are variants of the previous one where A and B can be any combination of tasks and goals.

- The third set of rules derives the positive and negative contribution to a softgoal for a given variant. We need two predicates since DLV does not support negative values.

- The last set of rules defines the unary predicates *ach* and *todo* from the binary predicates *achFrom* and *todoFrom*, respectively. The unary predicates are used in the other sets of rules and by RE-Context.

Apart from all the information expressed using the datalog syntax, there is some input information which is expressed using the mathsat syntax. First, RE-Context needs a file that contains the mapping from the symbolic context names (the c predicate) in Datalog to the actual context definition in terms of formulas over facts. For instance, the following code says that context c_0 is defined as the formula $a \wedge b$; c_1 as $c \wedge (d \vee e)$; c_2 as $d \wedge e$.

c(0) FORMULA a AND b
c(1) FORMULA c AND (d OR e)
c(2) FORMULA d AND e

The last piece of input information for RE-Context is a mathsat file which defines relations between context facts. The following example defines four facts (a, b, c, d) and defines a formula with two relations between such facts: $a \rightarrow \neg d$, $c \rightarrow d$.

VAR a,b,c,d: BOOLEAN FORMULA
(a → NOT d) AND
(c → d)

5.2 Methodological Process

In this section, we explain a methodological process to follow for constructing a contextual goal model for a system operating in and reflecting multiple contexts. The overall picture is depicted in the activity diagram in Figure 5.3. Four macro-activities are identified: goal analysis, context analysis, reasoning about contextual goal models, and identifying monitoring requirements.

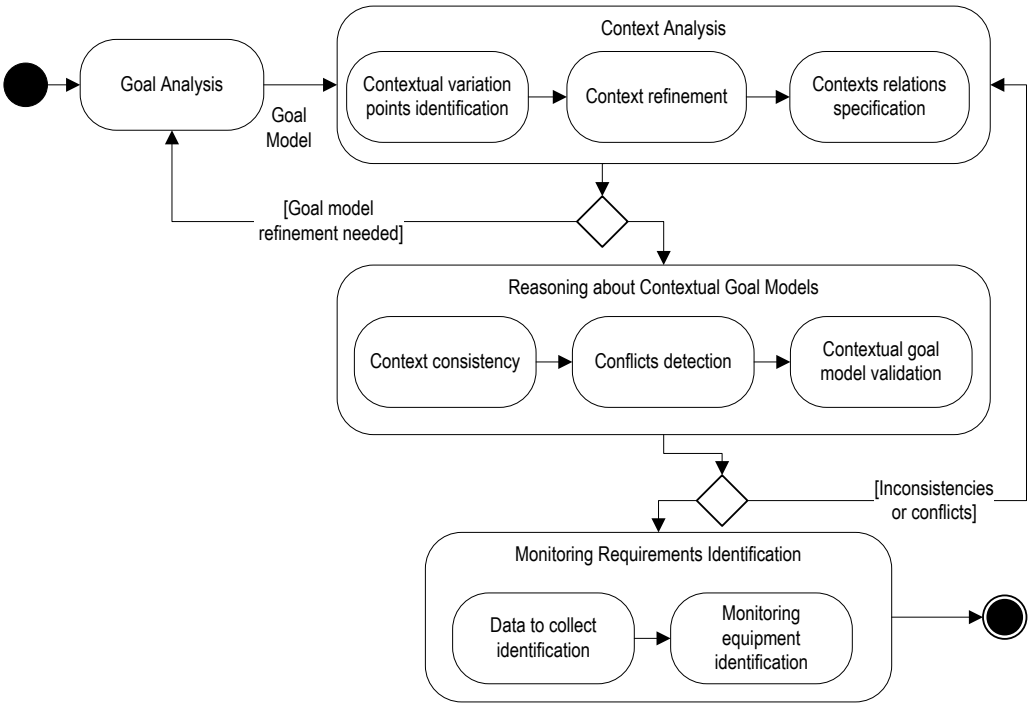


Figure 5.3: Analysis process for contextual goal modeling

1. **Goal analysis**: in this activity, high level goals are defined and analyzed. Goals can be iteratively identified through scenarios [60]. Moreover, an intentional variability taxonomy [68] 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**: this activity weaves goal modeling with context aiming to link the requirements, at the goal level, to the context in which they are activated and adoptable. Context analysis activity is composed of:

(a) *Contextual variation points identification*: for each variation point at 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 context independent. 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 Figures 4.1, 4.14 and the contexts associated with them are described as shown in Tables 4.1, 4.2.

(b) *Context refinement*: in this activity, the contexts at each contextual variation point are analyzed. The analysis is for identifying the way the contexts can be verified through. 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 analyzed context holds, as shown in the example of Figure 4.2. Moreover, in this activity, the analyst has to deal with different views of different stakeholders about the analyzed contexts. Stakeholder may define context differently, and even in contradictory ways. In case of inconsistency between stakeholder on context refinement, the analyst has to look for reconciliation among them 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 higher 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 levels such as the facts level. We still do not provide an automated support for the specification of these relations and presume that this activity is done manually.

3. **Reasoning about contextual goal models**: this activity is supported by our developed automated reasoning tool (RE-Context). The tool allows reasoning about contextual goal models for different reasons. It analyzes a contextual goal model in order to detect inconsistency in contexts specified on it and potential conflicts among its executable

processes (tasks). Moreover, the automated reasoning tool allows us to validate 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. Here we detail the three kind of reasoning done in this activity:

(a) *Reasoning about context consistency*: this reasoning is to check 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 are subject to design decisions as we have explained in Chapter 4. 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 mark their inconsistency as accepted as well.

(b) *Reasoning about conflicts*: in order to 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-decomposi tions. 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 correctly.

(c) *Validating contextual goal models*: this reasoning is to ensure that the contextual goal model reflects stakeholder' expectation of the system in different contexts and in compliance with their priories. To this end, the the analyst can ask stakeholder to give a context and then show them the correspondent goal model variants. Alternatively, the analyst may ask stakeholder to derive the variant in a given context and compare it with the one obtained by our automated analysis. This test might be done for the whole goal model or parts of it. Moreover, user prioritization might be con-

sidered 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 [92, 62] and as we have shown in Chapter 4.

The reasoning about contextual goal model can be done iteratively to facilitate the correction of discovered errors. Whenever a new refinement of goal/context is done, we can start with the above reasoning techniques. With regards to the reasoning for conflicts, it may be hard to decide the interaction between goal model variants and the objects in the system environment until we reach the tasks level. For this reason, we suggest to analyze parts of the goal model that have high probability of being in conflict first. We refine such parts until the tasks level and then we run the conflict analysis reasoning on them.

4. **Identifying monitoring requirements**: after context analysis and reasoning terminate, the analyst can identify the monitoring requirements. Monitoring requirements are fundamental to develop a contextual MobIS. 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 analyzing the facts obtained by context analysis, the analyst can identify the data needed to verify them as shown in Fig 3.5, 4.3. 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 reusability 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. Moreover, we have used class diagrams to represent the data conceptual model in this thesis. Recently, some other models have been proposed to represent the data to monitor in context-aware systems, e.g., the models proposed in [50, 56, 48]. The analyst may select one of these models instead of class diagrams if needed.

   (b) *Identifying the monitoring equipments*: for systems operating in and reflecting varying context it might be essential to specify the equipments to install and to use in order to enable data collection. This activity is for the specification of 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 4.1, we have given a brief description of the equipments needed for each of the contexts at the goal model. However, such specification of equipments becomes more precise after knowing the data to monitor in the environment, i.e, after the previous activity.

5.3 Chapter Summary

In this chapter we have described our automated support tool (RE-Context) and showed an analysis process to construct a contextual goal model. We have described the architecture and the functionality of our tool and explained what input it needs to do the target reasoning explained in Chapter 4. We have finally showed the process to capture contextual goal model and reason about them using the automated support too. The analysis process is iterative to allow the analysts for fixing modeling errors and validating the model. We also explained the models that should be the result of each of the discussed activities in our proposed process.

Chapter 6

Evaluation

In this chapter, we apply our modeling and reasoning framework on two scenarios of systems operating in and reflecting varying contexts. The first one concerns a smart home for people with dementia problems and the second concerns a museum-guide to assist visitors inside a museum. To this end, we have organized two lab sessions and invited a group of requirements analysts to apply our framework. The analysts have modeled the requirements of the two systems, smart home and museum-guide, using our contextual goal model. We show the outcome of the sessions in terms of the developed models developed and observation and feedback made by communicating with the participating analysts. We also report the outcome of applying our reasoning mechanisms on the developed models. We finally show the performance of our reasoning algorithms with respect the size of goal models and discuss its scalability.

6.1 Smart Home System

We consider a smart home system designed to support the life of people with dementia problems located at the campus of a health care institute. This system is a variation of the system described in [105] and studied as a part of the Serenity project[1]. The system is designed to assist the daily life of patients and provide continuous care about them ensuring their safety and comfort. To this end, smart home has to act in response to the context and may need to communicate with different parties: the caregivers in the health care institute, the police center, and the Medical Emergency Rescue Center (MERC). The smart home is mainly responsible of fulfilling the following categories of requirements on behalf of a caregiver:

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

- **Ensuring healthy environment inside home**: patients with dementia suffer from severe problems with memory and thinking. As a consequence of these problems, a patient with dementia may forget to keep the home environment healthy. For example, he may forget to refresh the air regularly, adjust the temperature, turn the oven off after using it, turn the electricity off when getting out, and so on. Smart home has to act autonomously on behalf of a caregiver to keep the environment of the home healthy.

- **Managing anxiety attacks**: people with dementia may suffer also from sudden anxiety or panic attacks and they may behave in an unusual way. A caregiver has to calm the patient down and prevent him of hurting himself. Smart home has to monitor the patient for anxiety attack symptoms and act on behalf of a caregiver when such symptoms become apparent. If the symptoms indicate a moderate anxiety, then the smart home may just try to make the patient aware of himself and his movements. In case of severe anxiety, smart home has to prevent patient of getting out to protect him. It also has to calm the patient down by play a relaxation music and adjust the light level and colors to create a peaceful environment for the patient. At the same time, smart home has to call a caregiver to come and treat the patient.

- **Communicating with caregivers**: smart home is required to notify and guide the caregiver to the patient's place when needed. The way used to notify a caregiver and to guide him to a patient's place has to consider the context of both the caregiver and the patient. Smart home has also to provide a caregiver with real time information about the patient's situation while he is on the way to the patient's place. The information that the caregiver should get, while walking toward a patient's place, has also to include the medical profile of the patient. This will help caregivers to estimate the severity of the problem and ask for additional help if needed.

- **Familiarizing patients with coming guests**: people with dementia may get agitated when they have guests and meet people even if they already know them. A caregiver may act to avoid such agitation by introducing the guests to patients before they arrive. A smart home will be responsible of this objective, i.e., it has to familiarize patients with guests coming to visit them after some time. One way to do such familiarization could be by showing a video or pictures about the guests coming or about some previous meeting the patients had with them sometime ago.

- **Protecting home against potential robbers**: as for any home, protecting against robbery is an objective that the smart home could be responsible of. Smart home has to give illusion that the home is lived in when the patient is out. This would help to prevent a robber of getting inside home. From the other hand, smart home has to act against any potential robber when he is inside the home area. Smart home has to monitor always the home area and figure out if a person is there in a suspicious way. If such occurs, smart home has to call and communicate with police.

- **Communicating with police**: smart home, when needed, has to communicate with and help police to protect home for different reasons such as robbery. Smart home has to notify police and enable police men of monitoring the home remotely. It may turn light and security cameras on and give control to police men. It also has to check when police men arrive home and allow them to enter quickly.

- **Entertaining patients and involving them in social activities**: patients may feel lonely and need to be entertained and socialized. A caregiver may do this by arranging a meeting between the patient and his relatives regularly. Alternatively, the caregiver may accompany the patient to social events organized at the health care institute or in the city. A caregiver may entertain patients by several ways such as playing films or selecting a TV channel showing a program of interest to the patient. Smart home has to do the activities a human caregiver may make to socialize and entertain a patient.

- **Rescuing patients in emergency cases**: in case of severe health problems, such as falling down suddenly, a caregiver may call the MERC to come and rescue the patient. MERC has to be notified and guided to the patient's home. In the mean time, MERC has to be informed about the current situation of the patient and his medical record. Upon arrival, a caregiver will open external doors to MERC and facilitate their entering to the patient's home. A smart home will act on behalf of a caregiver reflecting his objectives and rationale in case of severe health problems with patients.

Context plays a major role in deciding what requirements a caregiver has to meet and the way through which he may meet them. A caregiver that is assigned to a patient with dementia will keep watching the relevant contexts and take decision based on them. For example, a caregiver will try to socialize a patient when a context like "the patient is feeling bored and a long time

passed since his last social activity" holds. Caregiver will decide the right way to socialize a patient taking context into account. He may suggest the patient to go to a social event in the city with a friend of him if a context like "patient used to like going with a friend to such kind of social events and the event is not crowded" holds, as people with dementia get agitated of crowded places. The caregiver may, alternatively, organize the patient's attendance of a social event held at the institute if a context like "there is a social event in the next few days that matches with the patient's interest and profile" holds. The smart home has to reflect this rationale of a caregiver when socializing a patient.

6.1.1 Contextual Goal Model of Smart Home

To apply our modeling and reasoning framework, we have invited a group of five requirements analysts to participate in an evaluation session. All the participants have good experience in goal modeling and ubiquitous computing systems. We explained to them our framework including the conceptual model and the reasoning techniques. A domain expert with a practical experience in a smart-home owned by ITEA[2] has explained the scenario to all participants and answered their questions along the session. The participants were asked to model the smart home requirements using our contextual goal model. In this section, we outline the models developed collaboratively during the session, while the results of the reasoning techniques and the feedback gotten form this experiment will be reported in Section. 6.3.

Fig. 6.1, represents the contextual goal model of the Smart Home for patient with dementia problems. The model projects the system as a set of interdependent actors that together establish the main goal of the system about patients care giving. The description of contexts specified at this model is listed in Fig. 6.2. The last figure also contains the logical relations between these contexts. These logical relations will be used for the forthcoming reasoning.

The main actors of the contextual goal model of smart home system are the following:

- **Caregiving system actor**: this system actor is responsible for the support of daily life of patients when they are at home. It gives care to them ensuring their safety and comfort. It maintains a healthy environment at home, and we take the goal "fresh air inside home" as an example of this category of goals. It also manages anxiety attacks

[2]ITEA is the Social Housing Agency of the Province of Trento. www.itea.tn.it

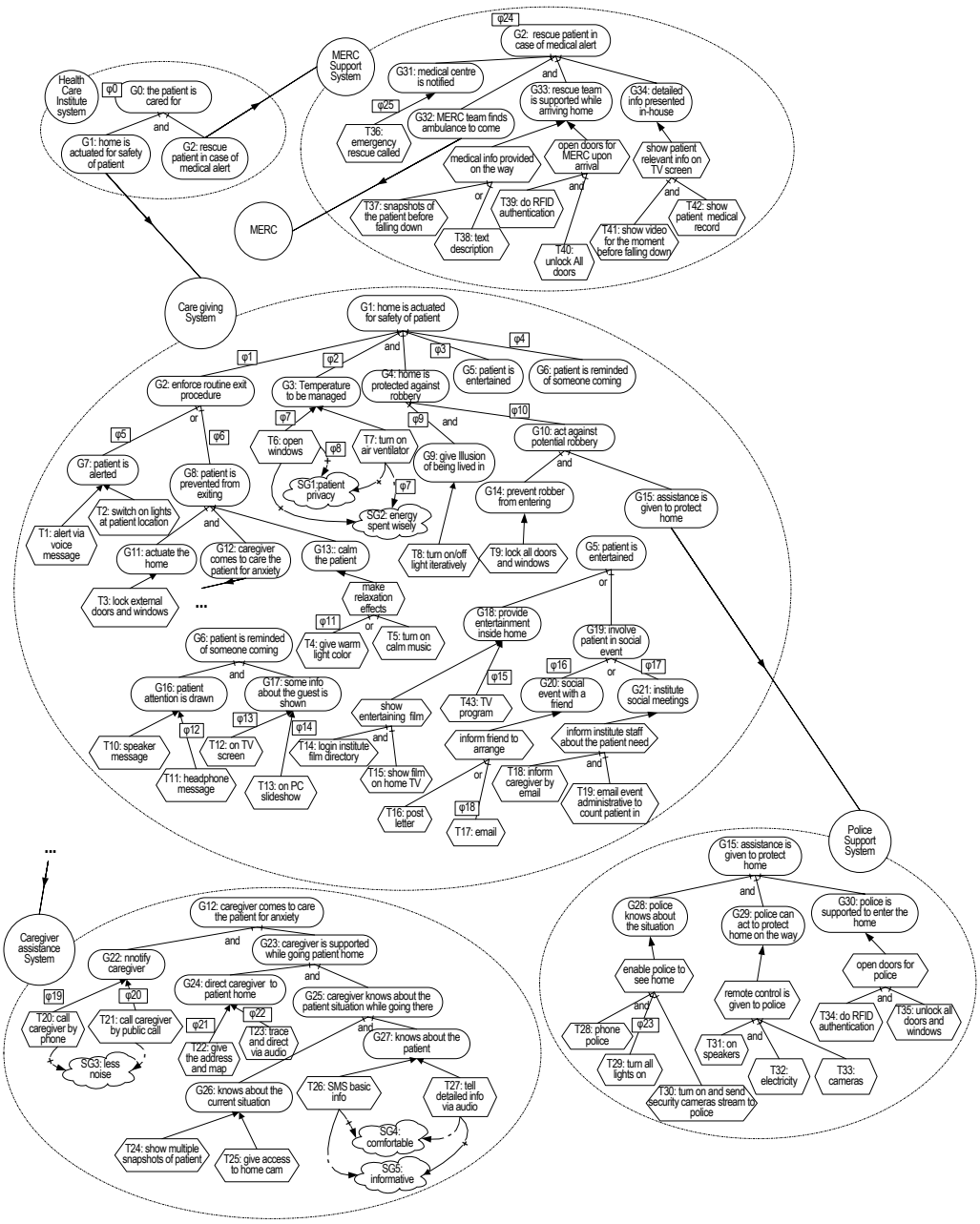


Figure 6.1: A contextual goal model for Smart Home.

both moderate and sever ones. This last goal may require a caregiver to come at home. To this end, this actor may depend on another system actor "caregiver assistance system" to reach the goal "caregiver

Context	Description	Logical Relations
Φ0	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.	
Φ1	Patient is anxious and he is inside home	
Φ2	Humidity level in the house is too high or home windows and doors have not been opened for long time.	
Φ3	Patient is feeling bored and long time passed without any entertainment activity and he is inside home.	
Φ4	There is somebody to visit the patient and the patient usually gets anxious when meeting people suddenly.	
Φ5	The patient dementia disease is not in an advanced stage and he is moderately anxious.	Φ5→¬Φ4
Φ6	The patient suffers from an advanced dementia or he seems to be extremely anxious	
Φ7	It is sunny and not very windy day	
Φ8	The patient is outside home.	Φ9→ Φ8 Φ8→ ¬ (Φ1v Φ3 v Φ4 v Φ24)
Φ9	The patient is outside home since long time and it is night time.	Φ9→ ¬ (Φ1 v Φ3 v Φ4 v Φ24)
Φ10	A person is trying to get into the yard in a suspicious way	
Φ11	The light level at the patient location is too low or too high.	Φ23→Φ11
Φ12	The patient puts the headphones on.	
Φ13	Patient is in the living room or in a place where he/she can notice if the TV is On.	
Φ14	Patient is working on the PC or in a place where he/she can notice if something is being shown on PC.	
Φ15	A TV program that can be interesting to patient is being shown on some supported channel.	
Φ16	The patient would like to attend a forthcoming event with a friend.	
Φ17	There is a social event in the institute that can be of interest to patient.	
Φ18	Friend has provided an email address and is checking his email regularly.	
Φ19	The phone is free and the caregiver is not using his/her phone for a call.	
Φ20	It is not night time.	Φ20→¬Φ9
Φ21	Caregiver is familiar with the institute campus or knows where the patient's home well.	Φ21→¬Φ22
Φ22	Caregiver rarely reached patient home or a home close to it and not familiar with institute campus yet.	
Φ23	The lights are off.	Φ23→Φ11
Φ24	The patient health turns bad and s/he has fallen down.	
Φ25	The MERC is reachable and online.	

Figure 6.2: The descriptions of contexts specified at Fig. 6.1 and some logical relations between them.

comes to treat the patient for anxiety". This system actor also acts on behalf of a caregiver to protect home from robbery. It gives illusion that the home is lived in to confuse potential robbers and acts when a robber comes inside the home area. A part of the actions, that this actor does in the last context, is the communication with police. This actor depends on another actor that is "police support system" to reach the goal "assistance comes to protect the home". Finally, this actor is responsible of socializing the patient, entertaining him, and familiarizing him with guests coming after short time.

- **Caregiver assistance system actor**: this system actor is responsible of communicating with and assisting caregiver to reach a patient's home. To this end, this actor notifies the caregiver in a context-dependent manner. For example, it may use the public speakers of the institute in the context "it is not late in the night", or by phoning him in the context "the home phone is free and the caregiver is not calling using his phone". It also guides the caregiver to a patient home in context-dependent way. If the caregiver is familiar with the campus and the patient, this actor will just give the address, otherwise it will trace and direct the caregiver step by step until arrival. This actor is also responsible of providing a caregiver, while coming to a patient's place, with real time information about the patient's situation and with his profile. This information is to save the time of a caregiver and help him to take better decisions.

- **Police support system actor:** this system actor is responsible of communicating and assisting police to come to a patient's home when needed. For example, protecting the home against robbery or some other emergency situations such as fire. The system has to inform police about the situation by calling them and turning the home lights and cameras on to enable police men of checking the situation remotely. The actor has to give control on electricity, cameras, and speakers to police in order to remotely act and protect home while coming. Moreover, this actor will authenticate police men when they arrive and let them enter to home to accomplish their task.

- **MERC support system actor**: this system actor is to communicate and facilitate the task of MERC for rescuing patients who got a severe health problems. This actor has to inform and guide MERC to patient's place. It also has to provide information to them while coming and while they are inside home. This information will be helpful to better understand the health problem of the patient and to act in a timely fashion.

The contextual goal model captures also the quality of goal model variants through their contributions to softgoals. For example, from the perspective of a quality measure such as "less noisy institute", calling caregiver by phone is assessed positively while calling him by speakers is assessed negatively. Some of these contributions are not always absolute but can be contextual. For example, opening the windows to refresh the air inside home is assessed positively from the perspective of the quality measure "patient privacy" when a context like "patient is outside home" holds. The value of this contextual

contribution is computed at runtime and the system computes it based on monitoring the current location of the patient.

An example of the context analysis that has been done when developing the contextual goal model of smart home is shown in Fig. 6.3. It concerns the context ϕ_{16}="patient would like to attend a forthcoming social event with a friend". Obviously, this context is not monitorable as it is but the system may derive it from some facts visible in the environment. The system may check if the patient is interested in the friend and the event and if patient would like to attend the event together with that friend. Each of these subcontexts need also further analysis to reach ways to verify them. For example, patient could be interested in a friend if he often reviews pictures and videos of himself with that friend, reviews that friend's social network profile, or exchanges emails and SMSs with him. All of these information can be monitored by the system and they can give evidence to the context analyzed.

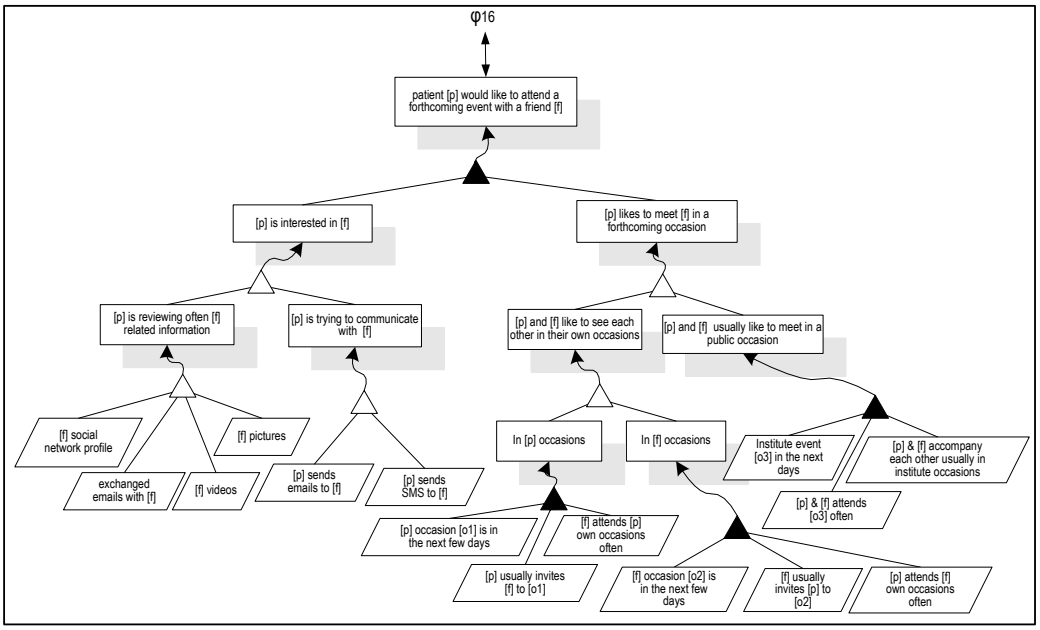


Figure 6.3: A context analysis example from Smart Home (φ_{16})

By analyzing the leaf facts of the context analysis shown in Fig. 6.3, we could elicit the conceptual data model shown in Fig. 6.4. The truth values of facts are computed based on these data. For example, the fact "patient sends emails to a friend regularly" is computed, mainly, based on the data of the class "Exchanged_Emails" that the system updates whenever a patient

writes an email. The process of verifying the root context φ_{16}, is based on collecting the data shown Fig. 6.4. The system has to collect such data and verify the leaf facts and propagate the computed truth values in a bottom-up way to judge if the analyzed context holds.

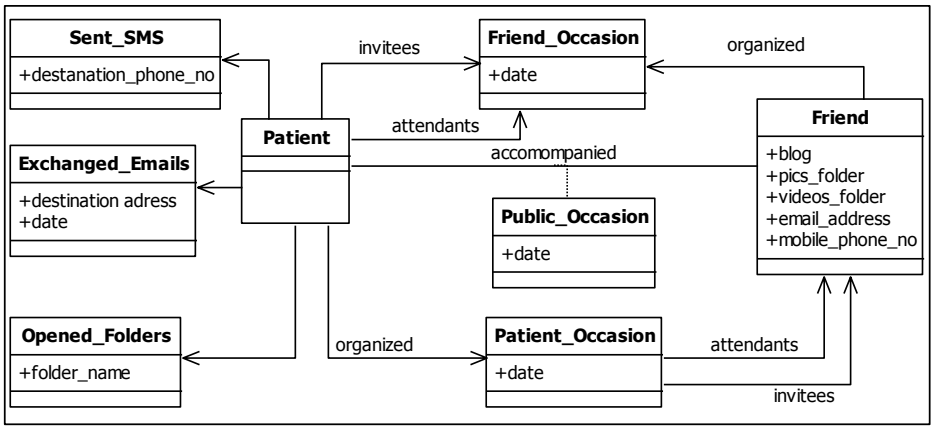


Figure 6.4: Data elicited by context analysis shown in Fig. 6.3

To analyze the contextual goal model for conflicts occurring among its tasks, we need to enrich it with two more other information that are (i) the parallelism and sequence operators between its nodes, and (ii) the influence of tasks on the home objects as we have explained in Chapter 4. In Fig. 6.5, we present the contextual goal model of smart home enriched with parallelism and sequence operators. The parallelism operator is graphically represented as ($\|$) and the sequence operator as (;). A top-down propagation of these operators will decide if two leaf tasks need to execute in parallel or in sequence. The specification of these operators at the higher levels of goal model hierarchy helps us to avoid the specification directly at the tasks level which is, generally, more hard and time consuming job.

With regards to the specification of parallelism and sequence operators, we remark here that the operators may cross-cut several decompositions and they are not necessarily specifiable directly at the subgoals (subtasks) of a single AND-decomposition. In other words, the specification of these operators at higher levels may not be always possible and we may need to specify them at more fine-grained levels of the goal model hierarchy. Although, this will not change our way of formalizing the model, it requires more manual specification of these operators. We would need more automated support to reduce the size of input provided manually by analyst. This input is essential to enable our target reasoning about conflicts.

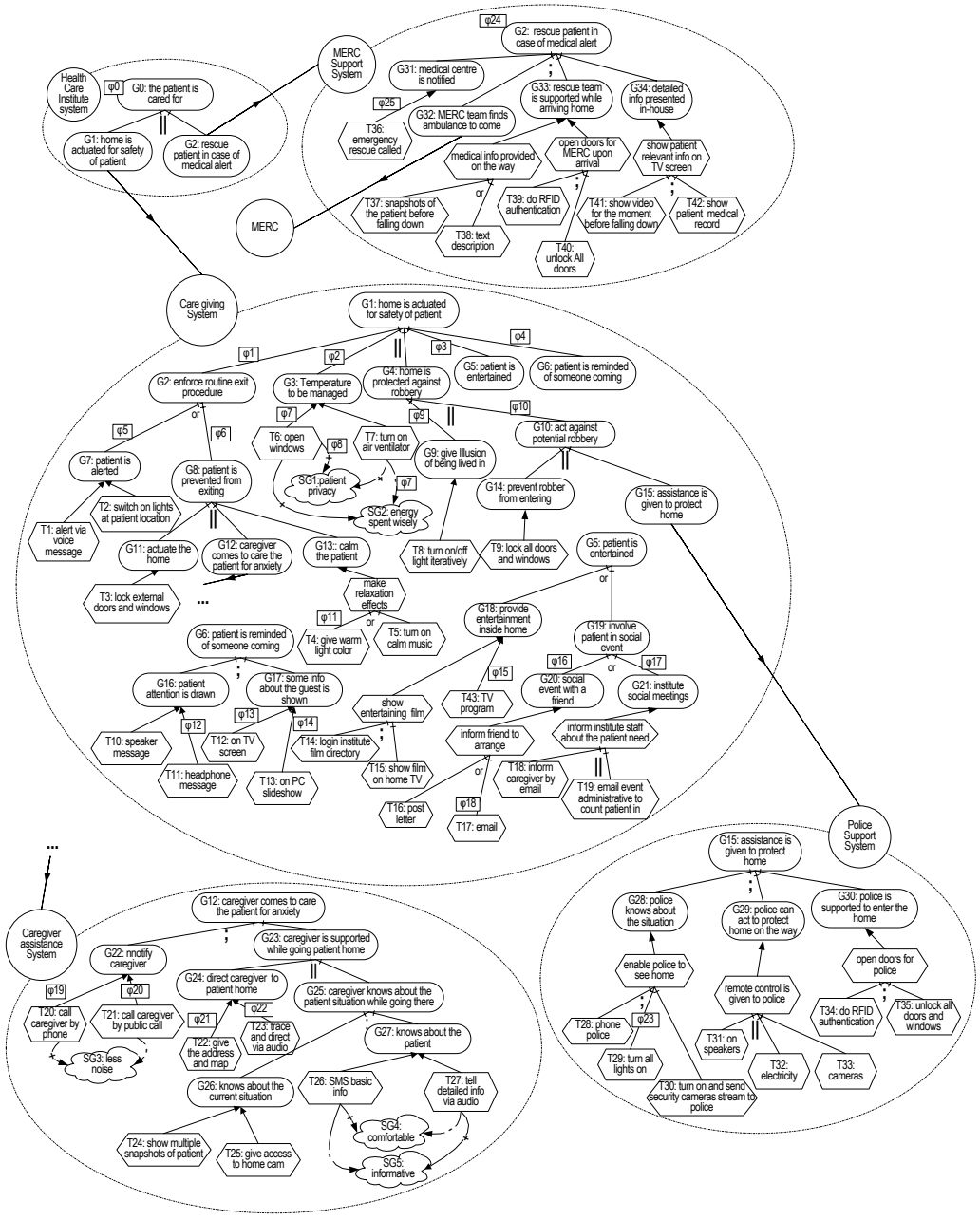


Figure 6.5: The contextual goal model of Smart Home annotated with parallel and sequence operators

The other information needed to enable the reasoning about conflicts concerns the influence of tasks on the system environment objects. The objects of patient's home and the influences of the leaf tasks of the goal model on them are listed in Fig. 6.6. For example, taking the object "external windows", there are four tasks that may change its state. T_3= "lock external doors and windows" will turn the state of the windows into "locked". The task T_6= "open windows" will change the state of windows into "opened". The task T_9= "lock all doors and windows" will turn this state into "locked" and the task T_{35}= "unlock all doors and windows" will turn it into "unlocked". Obviously, when T_3 and T_6 need to execute simultaneously, a conflict manifested on the object "external windows" will occur.

Environment Object	Tasks Vs. State changes /Exclusive possession of objects
Landline phone	T20.ex, T28.ex, T36.ex,
External doors	T3.locked , T9.locekd, T35.unlocked , T40.unlocked
Internal Doors	T9.locekd, T35.unlocked, T40.unlocked
External Windows	T3.locked, T6.opened, T9.locekd, T35.unlocked
Internal windows	T6.opened, T9.locked, T35.unlocked
TV Screen	T12.ex, T15.ex, T41.ex, T42.ex
Home speakers	T1.ex, T5.ex, T10.ex
Home lights	T2.on, T4.on.warm, T8.on, T8.off , T29.on
Ventilator	T7.on
Patient PDA Audio	T11.ex
PC Screen	T13.ex
Institute speakers	T21.ex
Caregiver PDA Audio	T27.ex, T23.ex
Caregiver PDA Screen	T22.ex, T24.ex, T25.ex, T26.ex.

Figure 6.6: The influence of Smart Home tasks on patient's home objects.

The changes over the context captured in Fig. 6.6 are those concerning the physical effect that the tasks of goal model lead to. Capturing this influence is enough for the kinds of conflicts that we address in this thesis, the conflicting changes and the exclusive possessing. However, the changes over the context themselves may be indirect and need further analysis. In other words, an analysis is needed to verify if a change did occur. For example, if the purpose of the tasks of showing a movie on the TV of a smart home is to relax a patient, then a diagnosis of the context "patient feels relaxed" has to be made. The diagnosis will need a set of visible facts that allow it to judge if this context holds. We may be able to use our context analysis constructs to make this analysis. However, this diagnosis and the mechanisms that the system may use in case of failures in changing the context are out of the scope of this thesis.

A partial description of the resources needed for developing each leaf task in the contextual goal model of smart home and the correspondent costs are shown in Fig. 6.7. This specification is essential for the reasoning that

concerns the development of minimum cost system that is operable in all considered contexts. Tasks may share the same development resources which means that a task may take benefit from the development of the others to reduce the total costs. For example, both tasks T_{42} ="showing medical record of patient on TV" and T_{12} = "show guests information on TV" share the resources of TV and the database management system. They, however, differ in the acquisition and manipulation of patient and guests data respectively.

We remark that we have considered only the costs of developing a task. Another interesting cost can be that related to the operation of a task. For example, the task T_{12} = "show guests information on TV" requires iterative job done by human operator to insert the data of new guests during the life of the system. Although the reasoning we proposed may cope with such new information, as it is a cost as well, we have not done this in this thesis. The estimation of operational costs of a task is not simple to decide. Consider, for instance, the operational costs of the same task T_{12}. One of the dimensions to consider when specifying it is the time. We may say, this task costs an employee salary per day. If the system is presumed to live for a number of years, then we can estimate simply the total operational cost of this task. However, this is a simplistic approach since systems usually are maintained over the time and the assumption of certain numbers of year is not often realistic.

6.2 Museum-guide System

In this section, we study a museum guide context-dependent system. The description of the system is provided by the Laboratory of Mobile Application (LaMA[3]) at University of Trento. The main objective of the system is to assist visitors inside museums and ensure the compliance of visitors' behavior with the rules of the museum they are visiting. To this end, the system is supposed to act on behalf of a museum staff and take actions in the right context. The system is intended to communicate with three parties that are the visitor, the assistance staff, and the rule enforcement staff. The main communication device is PDAs held by these three parties. The museum-guide is mainly responsible of fulfilling the following categories of requirements:

- **Supporting the museum rules**: visitor behavior inside the museum has to respect several rules. Some pieces of art should not be pictured, and the system has to act as a rule enforcement staff to ensure that

[3]http://lama.disi.unitn.it/

The tasks groups based on cost equivalence	Group name	Cost	Resource Description
T1, T5, T10	TA: Communicate via home speakers	Cost(TA, 100) (A,40) + (B,60)	A: installing speakers B: programming commands
T2, T4. T8, T29	TB: Actuating home lights	Cost(TB, 100) (C,40) + (D, 60)	C: Installing actuators D: programming commands
T3, T6, T9, T35, T40	TC: Actuating home doors and windows	Cost(TC, 120) (E,60) + (F, 60)	E: doors/windows actuator F: programming commands
T17, T18, T19	TD: Emailing	Cost(TD, 40)	
T20, T28, T36	TE: Phoning	Cost(TE, 50)	
T22, T26, T38	TF: SMS sending	Cost(TF, 40)	
T34, T39	TG: RFID authentication	Cost (TG, 90)	
T33, T25, T30	TH: Control on cameras	Cost(TH,160) (J,80)+(K,60)+(G,20)	J: installing commanded cameras K: programming commanded behavior G: giving control to other party
T31	Control on Speakers	Cost(T31, 60) (A,40) +(G,20)	
T32	Control on electricity	Cost(T32, 120) (H, 40)+(I,60)+(G,20)	H: actuator on electricity switches I: programming commands
T7	Actuating ventilator	Cost(T7, 100) (L,60) + (M,40)	L: installing commanded ventilator M: programming commands
T23	Positioning and directing system	Cost(T23, 140)	
T27, T11	TI: Sending audio message	Cost(TI,60)	
T37, T24	TJ: taking and sending snapshots to PDA	Cost(Tj, 110) (J,80)+(N,30)	N: sending snapshots
T14	Media server login	Cost(T14, 15)	
T15	Loading and showing video	Cost(T15, 15)	
T42	Showing medical record of patient on TV	Cost(T42, 100) (DB, 60) + (DBM1, 20)+(TV,20)	DB: DBMS TV: communicating information to TV DBM1: data manipulation
T12	show guest info on patient TV screen	Cost (T12, 100) (DB,60)+(DBM2,20) (TV,20)	DBM2: guest data manipulation
T16:	Printing and sending via post a letter	Cost(T16,20)	
T41	show TV video for the moment before falling down to police on TV	Cost(T41,100) (J,80), (TV,20)	
T13	show guest info on patient PC	Cost (T13, 100) (DB,60)+(DBM2,20) (PC,20)	PC: communicating info via patient PC.
T21	Call caregiver by public speakers	Cost(T21, 15)	
T43	Switching on TV	Cost(T43, 20)	

Figure 6.7: A partial description of the resources needed for Smart Home tasks development and their estimated costs.

visitors do not take pictures when it is not allowed. Moreover, some pieces of art should not be touched by visitors, and the system is also expected to enforce such a rule. In both cases, the system has to make the visitor aware of the rule when it is applied. It has also to act

to prevent potential violation of these rules and to communicate and provide information to the rule enforcement staff if a violation has been made. Some other general rules, such as being out of the museum by the closing time, has to be supported by the system.

- **Informing visitors about pieces of art**: the system is expected to deliver information about pieces of art to visitors when they tend to be interested in that. The museum contains several information terminals that the visitor may use to get information. The system may inform the visitor about that and guide him to the nearest terminal. There will be also a number of assistance staff that can be asked for information. The system may communicate and arrange a meeting between a free staff and the visitor. Also, the system may deliver appropriate information using the visitor's PDA. Each of these choices of delivering information has to be applied in the right context.

- **Communicating with rules enforcement staff**: when a violation of a museum rule is made by a visitor, the system has to inform the rules enforcement staff and facilitate his interaction with that visitor. The system has to send relevant information about the visitor to the rule enforcement staff such as his profile and his current location. Moreover, the system has to provide a proof that the visitor has violated the rules. For example, it can be a picture or a video showing the visitor picturing or touching a piece of art when this is not allowed. The picture can be printed via dedicated printer the staff can pick it from. Alternatively, the system may send a digital picture to staff's PDA.

- **Communicating with assistance staff**: the system is not supposed to completely replace assistance staff but to assist their work towards more comfort for them and visitors at the same time. In certain contexts, visitor would need a direct help from assistance staff and the system has to facilitate this task. The system is required to notify staff in a way that is context dependent. For example, there could be different ways for notification such as SMS, voice call, and museum speakers. Each of these ways is applicable in a certain context.

- **Advertising museum special events**: the museum adminstration organizes regularly special events as part of the museum cultural activities. The museum includes a theater and may allow for different kinds of cultural events such as concerts, movies, and plays. The museum-guide system is supposed to advertise such events to visitors who are potentially interested. Moreover, the system is required to assist the

reservation of visitors who are willing to attend one event. This reservation includes the specification of seats to them and their payment. The system has to support payment in two methods: online payment, and cash payment at a dedicated desk.

- **Obtaining visitors feedback**: besides the traditional guest book and the dedicated post box that the museums traditionally use, the adminstration of the museum is interested in getting visitors feedback through the system as a new mean. The system is required to get visitors feedback about the services provided at the museum and their opinion about the exhibited pieces of art. The system has to provide visitors with comfortable way of providing their feedback and encourage them to give it.

The system is required to autonomously decide the requirements to meet and the way to meet them taking into account the context. For example, delivering information to a visitor can be done through a terminal, his PDA, or an assistance staff. The information terminals may be used in a context like "there is a free terminal close to the visitor and he is able to use it". The visitor's PDA may be used in a context like "the information of the piece of art is not complicated and the visitor has the ability and the knowledge to use PDAs and new technology". Alternatively, the information can be delivered to a visitor through an assistance staff in a context like "the visitor is not able to use PDA and not familiar with terminals and the visitor is classified as an important visitor". The system has to always monitor the context and decide upon what alternative to follow and, consequently, what functionalities to execute.

6.2.1 Contextual Goal Model of Museum Guide

We have organized a seminar and invited four requirements analysts with a good expertise in goal modeling and mobile information systems scenarios that the museum-guide system belongs to. We have explained our RE framework to them and answered their related questions. We have then invited an expert in mobile application from the Laboratory of Mobile Application (LaMA[4]) at University of Trento to describe the museum-guide system scenario to requirements analysts. Then, we asked the requirements analysts to use our framework for modeling the museum-guide system requirements. Together with the domain expert, we have answered the questions the analysts have raised during the session. In this section, we outline the contextual

[4]http://lama.disi.unitn.it/

goal model developed during the session, while the results of the reasoning techniques and the feedback gotten form participated requirements analysts will be reported in Section. 6.3.

In Fig. 6.8, we show a contextual goal model for the museum-guide system developed during the session we organized. The model is annotated with contexts at some variation points. The description of these contexts and a set of logical relations between them are reported in Fig. 6.9. The main goal of the system is "visitor is assisted to make a useful and rules-compliant visit". The three system actors that collaboratively reach this root goal are:

- **Visitor guide system actor**: this actor is the main actor of the museum-guide system. It takes the initiative to assist visitors, to enforce the museum rules, promote special events organized at museum, and obtain visitors feedbacks. For example, this actor enforces the rule that prevents taking pictures of some pieces of art where picturing is not allowed. To this end, the actor issues a reminder, that concerns the rule, to a visitor when he enters a room that contains non-pictured pieces of art. The actor blocks the camera of the PDA given to the visitor by the museum when he is inside such rooms. When a visitor uses another device to take a picture of a piece of art, this system actor will take a set of actions. It will take several snapshots of the visitor taking a picture and send them to the rules enforcement staff system actor. This later system actor will help a rule enforcement staff to come and handle with the visitor who violated the rules. Similar scenario will be repeated concerning visitors who touched pieces of art when that is not allowed.

- **Rules enforcement staff system actor**: this system actor is responsible of communicating and facilitating the job of rules enforcement staff in handling with visitors who violated some of the museum rules. It enables the staff of getting an evidence about the violation of rules that this actor got already from the visitor guide system actor described above. This evidence is in form of several snapshots of a visitor taking a picture of a piece of art, or touching it, when it is not allowed. This snapshot may be in form of printed images or digital images. After that, this actor will give the necessary information about the visitor who violated the rules such as his profile, his current location, and his picture.

- **Assistance staff system actor**: this actor is responsible of communicating with and facilitating the work of assistance staff in order to help visitors. For example, the assistance staff may need to explain to

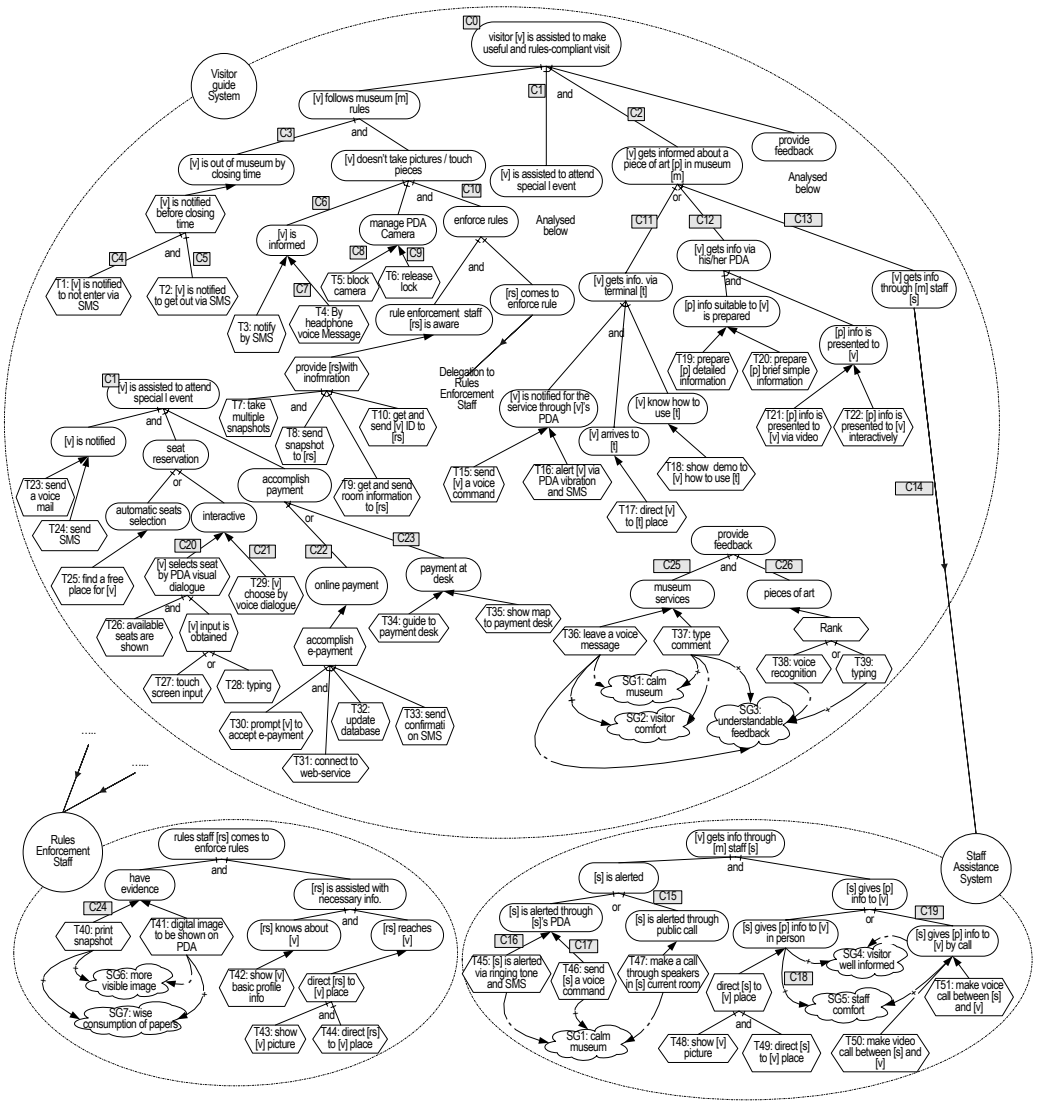


Figure 6.8: The contextual goal model for the Museum-guide.

a visitor about a piece of art in, or a service provided by, the museum. This system actor will notify the assistance staff by varying ways. The adoptability of each way can be context-dependent. For example, sending a voice command to a staff is adoptable when the staff is not calling and he is putting his headphones on. While sending a ringing tone and SMS, as a way of notifying a staff, is adoptable when the staff is not calling. This actor is also responsible of facilitating the communication between staff and visitors. This communication can be done either on

Context	Description	Relations with the rest
C0	The visitor has to be inside the museum area including parking places and the public square in front of museum. Moreover, the visitor should be registered to and have accepted the autonomous assistance by the museum-guide mobile information system.	
C1	There is a special event at the same day that could be of interest of the visitor, and still there are places.	
C2	Visitor is inside the gallery building and he is interested in getting explanation of a piece of art.	C2→¬C3 C2→¬C4
C3	Museum closing time is approaching.	C3→ ¬C25 C3→ ¬C26
C4	Visitor is still outside the museum and on the way to enter the museum.	C4→ ¬ (C5 v C25 v C26 v C6 v C10 v C8 v C9)
C5	Visitor is still inside the museum, and far away from the exit.	C25→C5 C26→C5
C6	Visitor hasn't been informed before and gallery contains non-pictured/ non-touchable pieces of arts and visitor has just entered the gallery.	
C7	Headphones are plugged-in and user puts them on, and does not listen to other audio contents or calling.	
C8	There is non-pictured piece of art close to the visitor.	C8→¬C9
C9	There is no non-pictured piece of art close to the visitor.	
C10	A picture for a non-pictured piece of art has been taken by the visitor or a piece of art has been touched where not allowed.	
C11	There is a free terminal close to the visitor and he/she is able to use it.	
C12	The information of the piece of art is not so complicated and the visitor has the ability and the knowledge to use PDAs and new technology.	C12→C20
C13	The visitor is not able to use PDA and not familiar with terminals and that the visitor is classified as important visitor.	C13→ ¬C20
C14	There is a staff that is free and talks a language common to the visitor and knows enough about the considered piece of art comparing to the visitor knowledge.	
C15	The room, where the visitor is, does not include audio art contents.	
C16	Staff assistance is not calling.	C17→C16 C19→C16
C17	Staff assistance is not calling and is putting the headphone on.	
C18	Visitor is close to assistance staff.	
C19	Staff is not using PDA for calling and information is relatively simple.	
C20	Visitor has good experience using PDA and new technology.	
C21	It is not noisy around the visitor.	
C22	Visitor has provided his card info and accepted e-payment in the registration process and the event accepts payment using the kind of card provided.	C22→ ¬ C23
C23	Visitor has not provided his card info and accepted e-payment in the registration process, or the event does not accept payment using the kind of card provided.	
C24	The PDA screen is small or visitor cannot recognize pictures on PDA screens.	
C25	Visitor is still inside the museum and has used one of the museum services and far away from the exit	
C26	Visitor is still inside the museum and has been showing much interest in a piece of art and far away from the exit	

Figure 6.9: The descriptions of contexts specified at Fig. 6.8 and a set of logical relations between them.

person or remotely in a context dependent way. To facilitate meeting on person, this system actor will send to staff the picture of the visitor and direct him to the visitor's location. To facilitate the remote communication, this actor may establish a video or voice call between staff and visitor.

Besides capturing the variants to satisfy the main goal of the system, the contextual goal model captures the quality of these variants from the perspective of certain quality measures (softgoals). It also captures the influence of context on the assessment of such measures through the notion of contextual contribution that we have proposed. In other words, the quality of each variant is not always absolute but may vary according to the context. For example, the assistant staff may be informed and guided to meet a visitor on person or to give information to him remotely. The alternative of giving information on person (G_{45}) has a good quality from the perspective of a quality measure such as "staff comfort" in certain contexts like "staff and visitor are close to each other". The system can choose between the variants, that are applicable in a certain context, based on their qualities. Consequently, the system has to instantiate the values of the contextual contributions based on the monitored context and then rank the alternatives of goal model based on the contributions each alternative gives to the set of softgoals as we have explained in Chapter 4.

Concerning the context analysis, it was necessary to make such analysis for certain contexts while in some other cases the definition of context was straightforward. For example, the context C_{22}="assistance staff is not calling and is not putting the headphone on" is a straightforward context that is a conjunction of its two facts. Some other contexts can not be verified in straightforward way and may need further analysis to get ways to verify them. For example, the context C_2="visitor is inside the gallery building and he is interested in getting explanation about a piece of art" is not monitorable by itself but it is an abstraction of other visible facts. An analysis is needed to discover and agree on the set of facts that gives a truth value to such high level context. Fig. 6.10 reports the analysis we have came up with to identify the variant ways to judge if C_2 holds.

The analysis shown in Fig. 6.10 refines hierarchically the context C_2 towards a formula of facts that specifies it. In other words, the analysis is done in top-down way from non-visible descriptions of the world (statements) into visible ones (facts). For example, the context "visitor is interested in a piece of art" is a statement that can be decomposed to more specific statements such as "he is behaviorally interested" and "he is historically interested". The first substatement can get support from visible facts such as "visitor usually asks for information about pieces of art of the same profile as the piece of art under discussion" and "visitor has attended the opening of the gallery that contains the piece of art". The second substatment can be supported by means of some other visible facts such as "visitor is looking at the piece of art for a long time" and "visitor had a look recently so often at the piece of art". We remark here that the truth of these facts does not give a com-

plete evidence to the refined statements. However, when we transform the context hierarchy into a boolean formula we assume this complete evidence. We may need to enrich our context analysis model with probabilistic and/or weighted "Support" relation between a world predicate formula and a refined statement and we leave this enrichment for a future work.

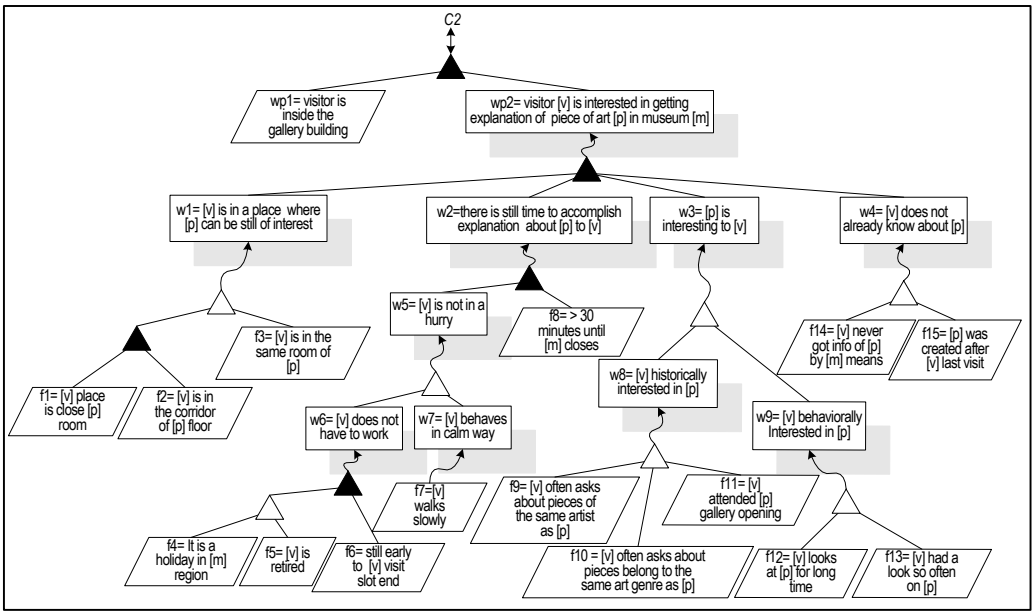


Figure 6.10: A Context analysis example from Museum-guide.

The context analysis helped us to systematically identify the data that the system has to collect of its environment. The leafs of the context analysis hierarchy are facts that are world predicates verifiable by an actor (the museum-guide system). The computation of the truth values of these predicates requires the system to collect data of its environment. For example, and taking the leaf facts of the statement w_3="piece of art is interesting to a visitor" of Fig. 6.10, we could elicit the data conceptual model shown in Fig. 6.11. A fact like "visitor looks at a piece of art for a long time" is computable based on the classes "Visitor", "Piece_of_Art" and "Looks_at". Using the terms of database systems, we may look at this fact as a *view* over the three mentioned classes.

Conflicts analysis, that we have explained in the last chapter, needs the specification of parallelism and sequence operators between the tasks of goal model. The specification of these operators at the higher levels of goal model hierarchy would lead to a fewer number of specifications in comparison to this

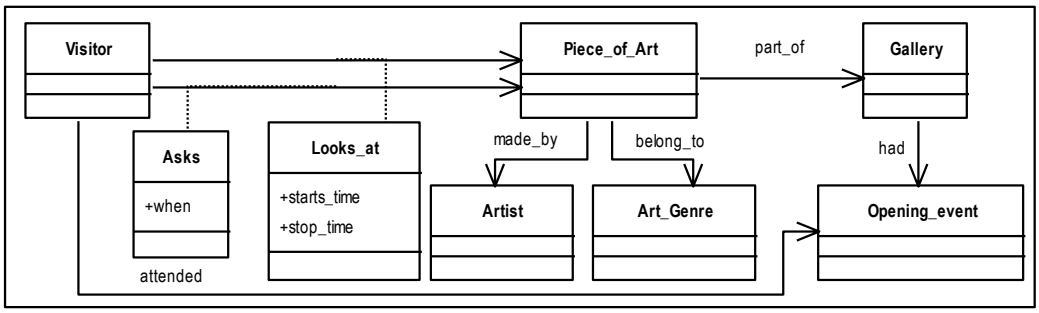


Figure 6.11: Data needed to judge the truth of w_3 of Fig. 6.10

number when we directly specify these operators at the level of leaf tasks. Moreover, it could be also hard to specify these operators directly at the level of leaf tasks without knowing the top level goals that are reached by those tasks. In Fig. 6.12, we show the contextual goal model of Fig. 6.8 enriched by the parallelism and sequence operators. The propagation of these operators at the higher level of the goal model hierarchy will decide if two leaf tasks execute in parallel or in sequence. Such information is essential to decide if there could be a conflict of the kinds we consider in this thesis.

Here we give an example to explain the meaning and the usefulness of specifying parallelism and sequence operators at goal models. The parallelism operator between the two goals G_1 and G_2 means that the two goals are independent. In other words, this operator means that reaching any of them does not depend on reaching the other. It also means that this operator applies between any node of G_1 hierarchy and any node of G_2 hierarchy as well. However, it is not always possible to specify this between each two goals/tasks in one AND-decomposition. In this case, we will need to go into the lower level and specify these operators. In the worst case, we would arrive until the leaf tasks level to be able of specifying them. Saying that G_1 and G_2 can be reached in parallel and they are independent means that this apply on any leaf task from the hierarchy of the G_1 with any leaf task from the hierarchy of G_2 and we avoid the enumeration of those pairs of tasks to specify the parallel/sequence operators.

The other information that we need to capture to enable the reasoning about conflicts concerns the impact of the leaf tasks of the contextual goal model on the objects in the system environment. This information is essential to judge if a conflict occurs. Two tasks executing in parallel are conflicting if they change the same object into two different states or each one of them needs and exclusive possession of one object. For example, both

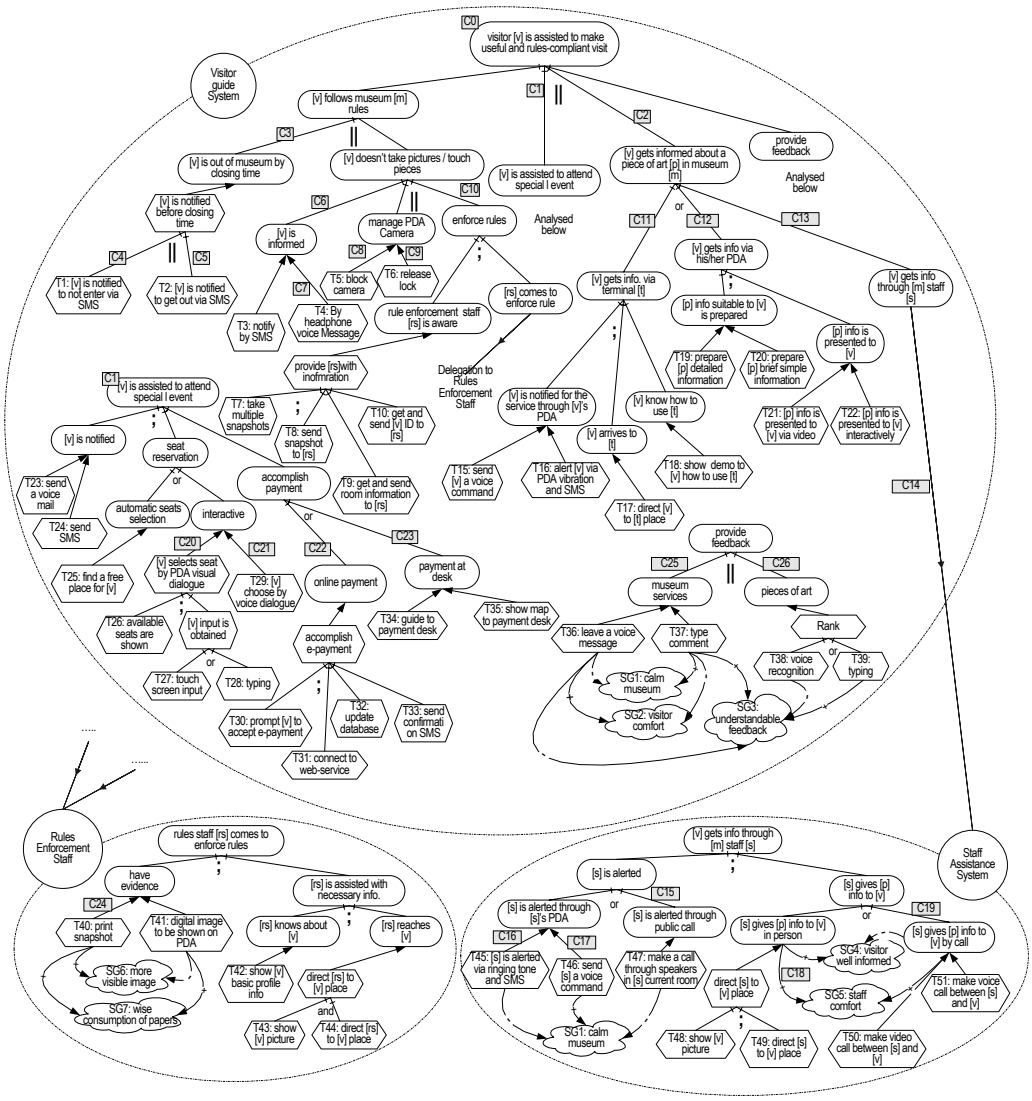


Figure 6.12: The contextual goal model of Museum-guide annotated with parallel and sequence operators.

tasks T_{33}="choose a seat by voice dialogue with visitor" and T_{55}= "make a voice call between visitor and staff" need an exclusive possession on the microphone of a visitor's PDA. If these two tasks execute in parallel, a conflict would occur. In Fig. 6.13, we show the impact of the leaf tasks of the contextual goal model shown in Fig. 6.8 on the objects in the museum environment. Differently from the smart home system described on the previous section, where tasks had impact on a living environment objects such as windows

and lights, the impact of tasks in the museum guide scenario is mainly on communication devices that are the visitors' and staff PDAs.

Environment Object	Tasks Vs. State changes /Exclusive possession over objects
Visitor PDA screen	T17.ex, T18.ex, T21.ex, T22.ex, T26.ex, T27.ex, T28.ex, T30.ex, T34.ex, T35.ex, T37.ex, T39.ex, T50.ex
Visitor PDA camera	T5.block, T6.unblocked
Visitor PDA audio	T4.ex, T15.ex, T22.ex, T29.ex, T34.ex, T36.ex, T38.ex, T49.ex, T51.ex
Visitor PDA microphone	T29.ex, T36.ex, T38.ex, T50.ex, T51.ex
Visitor PDA Vibration	T16.on,
Visitor PDA keyboard	T28.ex, T37.ex, T39.ex
Snapshot camera	T7.ex
Rules staff PDA screen	T8.ex, T9.ex, T10.ex, T41.ex, T42.ex, T43.ex, T44.ex
Assistance Staff PDA screen	T45.ex, T48.ex, T49.ex, T50.ex
Assistance Staff PDA audio	T45.ex, T46.ex, T49.ex, T50.ex, T51.ex
Assistance Staff PDA microphone	T50.ex, T51.ex

Figure 6.13: The influence of Museum-guide tasks on museum environment objects.

The reasoning about derivation of contextual goal model variants that lead to a system developed with minimum costs needs two information. These two information are the logical relations between contexts and the resources needed for developing each leaf fact. In Fig. 6.9, we have already showed a set of logical relations between contexts that are specified at museum-guide goal model. These relations are essential to decide the core groups of variants. In other words, to exclude those variants (i) that are unadoptable due to context inconsistency and (ii) those that can be replaceable by other variants in all contexts due to entailments between contexts as we have explained in Chapter 4. The second information concerns the resources needed for the development of each leaf task in the contextual goals model. As we mentioned earlier, tasks may share the same development resources. This influences the total costs of developing a set of tasks. In Fig. 6.14, we show estimation of the resources and costs related to each leaf task of the contextual goal model shown in Fig. 6.8.

6.3 Evaluation Results

To evaluate our modeling and reasoning RE framework, we have organized two lab sessions involving a group of requirements analysts that applied our framework on two systems scenarios. Each lab session concerned one of the systems described in this chapter: the smart home system and the museum-guide system. The requirements analysts, who participated in the sessions, have already a good expertise in goal modeling and they are familiar with emerging computing paradigms scenarios such as ubiquitous computing and

The tasks groups based on cost equivalence	Group name	Cost	Resource Description
T1, T2, T3, T9, T10, T16, T24, T33, T45	Ta: send notification via SMS	Cost (Ta, 40) (A,30) +(B,10)	A: communication Server B: programming notification via SMS
T4, T15, T23, T46	Tb: send notification via voice message	Cost(Tb, 40) (A,30)+(C,10)	C: programming notification via voice
T57	Tc: send public speakers message	Cost(Tc, 60) (A,30)+(C,10)+(D,20)	D: installing speakers for notifications
T42, T43, T48,	Td: showing visitor info on PDA extracted from the developed database	Cost (Td, 120) (DB,60)+(E,60)	DB: having DBMS E: manipulating visitors' data.
T25, T26, T32	Te: processing Database of seats	Cost(Te, 90) (DB,60)+(F,30)	F: manipulating seats information
T37, T39	Tf: getting via typing and storing comments in database	Cost(Tf, 80) (DB,60)+(G,10)+(HH,10)	G: getting input via typing programming HH: manipulating comments storage
T36	Tff: getting via voice and storing comments in database	Cost(Tf, 80) (DB,60)+(GG,10)+(HH,10)	GG: getting input via voice programming
T28, T30	Tg: on PDA local HCI.	Cost(tg, 10) (G,10)	G: getting user input via typing.
T17, T34, T44, T49	Th: directing interactively	Cost(Th, 120) (H,45)+(I,75)	I: installing positioning equipments and getting a person position H: interactive communication programming
T21	Ti: piece info shown via video	Cost(Ti, 20) (J,20)	J: programming a video
T22	Tj: piece information are shown interactively	(Tj, 140) (H,45)+(II,75)+(J,20)	II: manipulating piece information for interactive display.
T19	Tk: detailed info about piece of art are prepared and stored.	Cost(Tk, 120) (DB,60)+ (JJ,30)+(K,30)	JJ: gathering and storing basic info. K: gathering and storing advanced info.
T20	Tl: brief info about piece of art	Cost(Tl, 90) (DB,60)+ (JJ,30)	
T5, T6	Tm: Block/unblock Camera	Cost(Tm, 10)	
T7	Tn: snapshots	Cost(Tn, 30)	
T29, T38	Tp: Voice recognition	Cost(Tp, 80)	
T50	Tq:Video call	Cost (Tq, 70) (L,40)+(M,30)	L: connecting and voice call M: video calling
T51	Tr: Voice call	Cost(Tr, 40) (L,40)	
T31	Ts: Connect to Web Service	Cost(Ts, 20)	
T18	Tt: Demo about terminals	Cost(Tt, 15)	
T35	Tu: Map to payment desk	Cost(Tu, 10)	
T8, T41	Tv: send snapshot	Cost(Tv, 40)	

Figure 6.14: A partial description of the resources needed for Museum-guide tasks development and their estimated costs.

mobile information systems. We have explained to them our framework and answered their questions about it. In each session, we have asked the analysts to model requirements using our contextual goal model. During the development of the models, a domain expert was present to answer the analysts' questions related to the systems being modeled.

We classify the obtained results into three categories. The first one concerns the quality of our framework from the perspective of practitioners. This evaluation is based on our observations, and interviews with participated analysts, that we have done in both of the two lab sessions. The second category concerns the results we got when applying our reasoning on the contextual goal models developed by the analysts. This evaluation is to estimate how important these techniques are in practice. The third one concerns the performance of our developed tool that implements our proposed reasoning mechanism. This evaluation is to estimate the scalability of our tool with respect to the size of processed models.

6.3.1 Analysts feedback and observations

During each of our organized lab sessions, we have observed the attitude of analysts when using our framework and documented the questions they asked frequently besides the common difficulties they used to face. After each session, we have interviewed the participated requirements analysts to get their feedback about our framework and to confirm our observations that we made during the sessions. The observations we made, and the feedback we got, were helpful to evaluate our framework from the perspective of practitioners acceptance and efficacy of use. In this regard, the main obtained results are the following:

- **The proposed extension to goal model is easy to understand**: the main extension to the goal model, that we have proposed, concerns the identification of variation points at goal model and the modeling constructs for analyzing contexts. The analysts easily understood this extension and applied it in the right way. Moreover, the additional constructs, that are needed to enable the reasoning about contextual goal models such as conflict and costs, were also easy to understand and the analysts have used them in a straightforward way.

- **The framework provides a useful systematic way to analyze contexts**: this systematic way is of high importance especially when a group of analysts specify, and/or need to agree on the specification of, one context. It allows for a step by step refinement of high level

contexts that are not monitorable/verifiable per se. The goal of this refinement is the discovery of visible facts that an actor, possibly the system, may capture of its environment to verify if the refined context holds. Besides the systematic refinement of contexts, context analysis constructs showed very useful way for the communication between analysts. For example, when an analyst had to give the specification of one context in terms of a formula of facts, it was hard to explain why the set of facts is relevant and why they must be composed in that given way to judge if the analyzed context holds. Doing the hierarchial context analysis, his rationale when analyzing context became clear to all other participants.

- **The relation between goals and context is strong in certain systems**: in the systems we have dealt with during the lab sessions, that are the smart home and the museum guide systems, the analysts could recognize the relation between goals and context and the importance of this relation. The analysts confirmed to us that goal models is a right abstraction level for considering certain contexts especially the contexts that influence human's decisions. The reason is that the context influences such human's decisions before the software itself and goal model can represent an actor intentions and rationale. Capturing this influence is preliminary as it allows software to reflect human adaptation to context and, consequently, derive useful functionalities to execute. From the other hand, some other contexts, such as the bandwidth of the network and the indexing mechanisms supported by a DBMS, should be defined at later stages of the development, possibly the design, and goal models may not be the right abstraction level to capture them. Moreover, our proposed conceptual model is mainly suitable for the requirements of a kind of systems that operates in and reflects varying contexts. Other systems, that are inherently invariable when operating in different contexts, will not need our proposed modeling and reasoning. For example, a system to register students in a faculty often follows a uniform process that is not a subject to contexts changes.

- **Context analysis is needed for certain contexts**: during the sessions, the analysts did not need to use context analysis for simple contexts and it was possible to specify them at the level of facts directly. In such cases, there was an immediate agreement between the analysts on how such simple contexts can be verified. For example, in the smart home system and concerning the contexts φ_{20}="it is not

night time" and φ_9="the patient is outside for long time and it is night time", there was an immediate agreement between the analysts that these two contexts are directly based on verifiable facts and there is no need to analyze them further. Other contexts, included a straightforward parts and parts that needed further analysis. For example, in the smart home system, the context φ_3="the patient is feeling bored and long time passed without an entertainment activity and patient is at home" included a straightforward part that is "long time passed without attending an entertainment activity and patient is at home" and a part that needs analysis "patient is feeling bored". The analysis is needed to identify and agree on facts, verifiable pieces of information, that give evidence to such a context.

- **Context analysis could be a subject to viewpoints**: for some complex contexts, there have been disagreements (viewpoints) between the analysts concerning the correct refinement of a context. For example, in the museum guide system and concerning the context C_2="visitor is inside the gallery building and he is intersted in getting explanation about a piece of art", there were viewpoints about the way the statement "the piece of art is interesting to the visitor" can be supported by facts. Some analysts stated that new pieces of art should be interesting to a visitor regardless his profile. Other analysts stated that a piece of art is interesting to a visitor if he has shown interest in similar pieces of art previously regardless if the one under discussion is new or old one. In some other cases, it was debatable if a certain context is a statement or a fact. For example, in the museum guide system and concerning the context C_{11}="there is a free terminal close to the visitor and he is able to use it", some analysts considered being close to a terminal as a fact while others considered it as a statement. The reason for the latest opinion is that considering something close or far from a person depends on different factors such as age and physical ability of that person. Therefore, and as a result of the above two observations, new policies to manage such situations are still required. One of our future work directions concerns the detection of mismatches between different analysts' viewpoints, the assessment of the severity of this mismatch, and the resolution policies.

- **Context analysis could be more expressive**: in other words, other modeling constructs could be needed for more expressiveness of our proposed context analysis. One of the possible extensions to our context analysis could be the temporal relations between contexts. For exam-

ple, in the smart home system and concerning the statement "patient did not succeed to sleep", the analysts wanted to decompose this into two statements "patient recently tried to sleep", "patient is now not sleeping" and model that the first statement should have occurred before the second. Our modeling constructs do not explicitly capture this last information. Obviously, this information is important for the system at runtime to decide the truth of a context and require us more work to model and reason about it. Moreover, a weighted "support" relation is needed to decide the extent to which a statement is true when the supporting world predicates formula is true. For example, the fact "walking slowly" gives high evidence to the statement "customer is not in a hurry" while the fact "is still early to the closing time" gives lower evidence to it.

- **The specification of user prioritization is limited**: we have proposed the specification of user priorities as ranks given to softgoals. Sofgoals are used as quality measures of the goal model variants. In a certain context there could be more than one applicable variant and we rank them based on their contributions to softgoals. The variant that better contributes to the highly ranked softgoals is preferred. While this way avoids us handling with the variants themselves, that may be hard task when we have a large number of variants, it is somehow limited. For example, two variants that have no contributions to softgoals will be ranked equally. Another criteria to rank variants can be the history of the system with each variant. In other words, the priorities of users are not always specifiable at one step. Rather, we may need the system to learn over the time and decide the variants the user may give high priority. To this end, new concepts could be needed and our model may need further enrichment to capture them. For instance, we may need concepts to capture what the system has to monitor to judge the users attitude against each variant. Such information may allow the system to evaluate the priorities of variants continually during the actual operation of the system.

- **The size of input, provided manually, to enable reasoning may become big**: the proposed reasoning for variants derivation in its two kinds has reasonably small size input provided by analysts. Variants derivation for a given context and user priorities requires a manual ranking of softgoals. This ranking involves manual specification of the priority of each softgoal once. The variant derivation for system developed with minimum costs requires the analyst to provide the de-

velopment costs of each task once. From the other side, the input required for reasoning about contextual goal model consistency may become big. We presumed that the logical relations between contexts (facts and statements) are manually provided. We also presumed that the parallelism and sequence operators between goal model nodes are provided by the analyst. These operators may not be specifiable at each AND-Decomposition immediately as it was the case in the systems we dealt with. In some cases, we may need to go more steps down in sub-hierarchies of an AND-decomposed nodes in order to be able of specifying these relations. This obviously will increase the number of specifications needed to be provided by the analysts. Consequently, more techniques and automated support are needed to help analysts and reduce the size of input they need to provide.

- **There are more kinds of conflicts to consider**: we have considered a fairly simple kind of conflicts. It concerns the conflicting changes on the state, and the exclusive possession of, shared objects between two executable processes (tasks). Our objective was not to study and develop through detection of all kinds of possible conflicts and problematic interactions between tasks that are manifested on context. The objective was to explain the importance of discovering the goals behind the conflicts, the context in which a conflict happens, the alternatives the system may adopt to avoid a conflict, and the quality of those alternatives. The analysts agreed on the importance of this reasoning as a way to explain conflict and provide useful information for better resolution decisions. Obviously, many other kinds of conflicts may exist such as the cyclic stimulation between two variants caused by the changes in the context that each variant leads to.

- **There are other kinds of costs to consider**: we have considered the costs of resources needed for developing each task in the goal model. We have used this information to derive the less expensive set of tasks implementing a set of variants that allow the system to operate in all considered contexts. However, there could be different kinds of costs to consider when we do this activity. The cost of a task does not only mean the cost of developing it but may also include the costs of the operation of this task. For example, suppose we have two tasks for delivering confirmation about a visitor's reservation for a special event in the museum. The first one is by sending e-copy and reservation number via bluetooth, WiFi, or SMS to his PDA. The other is by printing the reservation document on a dedicated printer. The first

option is more expensive as the programming needed is more complex. From the other hand, the costs of printing over the time will make the second task more expensive. The analysts have made this observation and we would need modeling and reasoning about costs that is more thorough than the one proposed in this thesis.

- **The contextual contribution is, sometimes, hard to put as a binary decision**: context may have different influences on requirements. It may influence the set of requirements relevant to a system, the set of possible ways to reach them, and the quality of each of such ways. We have adopted goal model to represent requirements and variants ways to reach them and the qualities of each of such ways. The concept of softgoals can be used to represent the quality of each way of reaching goals. The influence of context on the quality of each way to meet goals was captured through the concept of contextual contribution to softgoals. In other words, the quality of each way to meet goals is not absolute but context dependent. However, we considered the influence of context on contributions to softgoals as binary, i.e., positive or negative. This could be hard assumption in some cases. For example, the distance between a person using his PDA to connect to WiFi and the connection access point decides how reliable the connection is. Such distance does not influence the reliability of a connection in binary way. In other words, the quality measure "connection reliability" will be assessed more positively/negatively based on the distance between the visitor's PDA and the connection access point. We would need ways to model and reason about more detailed relation between context and contributions to softgoals in the contextual goal models.

6.3.2 Reasoning results

Each of the groups, who participated in our organized lab sessions, has delivered a contextual goal model for the system studied at that session. We got a contextual goal model for the smart home system and another one for the museum-guide system. We have formalized these two contextual goal models and run our automated reasoning support tool and got the results of the reasoning proposed in this thesis. In this regard, Fig. 6.15 reports different categories of information. We report the time needed to construct and formalize the contextual goal model, the size of goal model, the results of applying the reasoning mechanisms on it. We report this information for both of Smart home and Museum guide systems in the same figure.

Factor	Description	Smart Home System	Museum guide System
TD	time to develop the graphical model (in hours)	14	16
TF	time to formalize the model & fix inconsistencies (in hours)	6	7.5
NA	Number of actors	5	3
NG	Number of goals	35	41
NT	Number of tasks	50	51
NSG	Number of softgoals	5	7
NVP	Number of variation points	25	26
V	Number of variants	25560	324000
IV	Number of variants with inconsistent contexts	11556	322612
I	Number of iterations needed to fix/accept all context inconsistencies	27	40
NCV	Number of non-core variants	1908	192
CGV	Number of core groups of variants	192	84
Results of Reasoning about minimum development costs			
TC	Total cost of developing all tasks each one separately	3595	2845
SC	The shared costs between all tasks	1820	2045
AAC	The costs of developing all alternatives of goal model	1775	800
MC	The cost of developing the set of tasks that lead to a system operating in all considered contexts.	1185	525
Results of reasoning about conflicts			
NC	Number of conflicts	29	121
VwC	Number of variants with conflicts	13789	1224
CCGV	Number of conflictual core groups of variants	184	36

Figure 6.15: A report of reasoning results on both Smart home and Museum-guide systems.

- **Development time**: as shown in Fig. 6.15, we divide the time needed to construct the contextual goal model into two parts. The first one concerns the development of the conceptual graphical model (TD). This includes modeling the goal model, the variation points, context analysis, development costs, parallelism and sequence operators, tasks influence on the system environment objects, and the logical relations between contexts. The second one concerns the time needed for the formalization of the contextual goal model (TF). We formalize it using Datalog, and formalize the context analysis hierarchies as boolean formulae (formulae of facts). We still do not have an automated translator of the graphical models into formal representation. Moreover, the second time includes the time needed to decide about fixing or accepting context inconsistency as our developed automated support tool interacts with analysts and this requires time to take the right decision as we have explained in Chapter 5.

- **Size of goal model**: the size of goal model is reported in Fig. 6.15 in terms of the number of nodes (goals (NG), softgols (NSG), tasks (NT), and actors (NA)). Moreover, the number of variation points (NVP) and the number of goal model variants (V) are main factors in deciding the complexity of goal model. For example, a goal model with only AND-Decomposition does not allow for variants to goal satisfaction. Actually, such a goal model has only one variant. This makes the reasoning we propose for variants derivation (for given context and user priorities, and for minimum developments cost system) meaningless. It also limits the importance of reasoning about conflicts and context consistency since they will concern only a single variant. We also report the number of variants that are non-core (NCV), i.e. those replaceable by others in all considered contexts, and the number of core groups of variants (CGV). These measures are indicators to the size of input the reasoning about costs and conflict will deal with.

- **Reasoning about context inconsistency**: our automated support tool is designed to interact with analysts when detecting a context inconsistency. When an inconsistency in a context of a variant is detected, the tool asks the analysts to decide about accepting it or modifying the model and repeating the consistency check. When the inconsistency of a goal model variant context is accepted by analyst, the tool accepts the inconsistency of contexts of the other goal model variants that include the discussed one. This is for reducing the number of interactions with analyst as we have explained in Chapter 5. The number of iterations (I) where the tool interacts with analyst is also reported.

- **Reasoning about costs**: the other category of information that we report in Fig. 6.15 concerns the results of our developed reasoning about costs. We report four information in this regard. The first one concerns the sum of development costs of each task separately without considering the shared resources (TC). The second concerns the sum of shared resources between all tasks (SC). Having these two information enables us to estimate the ratio of shared resource of the total costs of developing the tasks separately. The third information concerns the costs, considering the shared resources, of the development of all tasks (AAC), i.e., implementing all goal model variants. The forth information concern the cost of developing a set of tasks that allows for a system operable in all considered contexts with minimum cost (MC). However, our reasoning finds this set of tasks as well and the figure reports only the cost of developing the tasks belonging to this set.

- **Reasoning about conflicts**: the last category of information reported in Fig. 6.15 concerns the results of reasoning about conflicts. We report the number of conflicts that may happens between the tasks of a contextual goal model (NC). We report this number since the same conflict may occur in multiple goal model variants and it, therefore, gives an indicator about how many conflicts we need to fix if we wanted a conflict-free goal model. The other information we give concerns the number of variants with conflicts (VwC). This number indicates how many variants are affected by the set of conflicting tasks. The last information is about the number of conflictual core groups of variants (CCGV). In other words, the number of cases in which the resolution can not be done by adopting a conflict-free alternative and, therefore, a resolution strategy has to be provided.

6.3.3 Performance analysis

We have developed a CASE support tool, called RE-Context, to implement our proposed reasoning about contextual goal model. In order to assess the performance of RE-Context, we installed it on a machine with two CPUs AMD Athlon(tm) 64 X2 Dual Core Processor 5000+ and 4 GB of RAM. Fig. 6.16 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 plus 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 graph on the right-hand side depicts the results

shown in the table: the "x"-axis represents logarithmically the number of variants, the "y"-axis represents logarithmically the computation time needed. The collected data shows that the time needed for computation is growing exponentially with the increase of the problem size. Anyhow, given that the reasoning is performed at design-time, the tool scales quite well in the test cases (it takes less than 16 minutes with 648.000 variants).

The time to compute the conflictual core groups of variant is negligible in comparison to the time needed to compute the core groups of variants themselves. Also, we do not present the results for deriving variants under given context and user prioritization over softgoals, as the computational cost is negligible. Moreover, the algorithm we have developed to derive variants for a system developed with minimum-cost led to a negligible time as well. The naive algorithm that takes the cartesian products of core groups of variants and find the combination with minimum costs was highly expensive. Our minimum-costs variant derivation algorithm was developed to exploit the nature of the problem we deal with and reduce the complexity of the naive approach.

To get large goal model sizes, we adopted an approach similar to the one adopted in [110] and cloned the original goal model that we have developed for the smart home system. RE-Context needed 40 minutes for a goal model of 100,000 variants. As shown in Fig. 6.16, the derivation of goal model variants and the inconsistency check scale quite well, whereas the identification of core groups of variants has scaling 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 crucial. Since our proposed reasoning is done at design time, time is not a critical problem for medium size goal models. However, we still need optimization of the algorithms used to minimize the complexity when dealing with very large scale goal models.

To deal with scalability problems for very large goal models, we might benefit of two technique. The first is the iterative check the model during construction. We can reason about consistency and conflicts while constructing the goal model instead of treating the entire final goal model at once. 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 a simple cartesian product.

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.

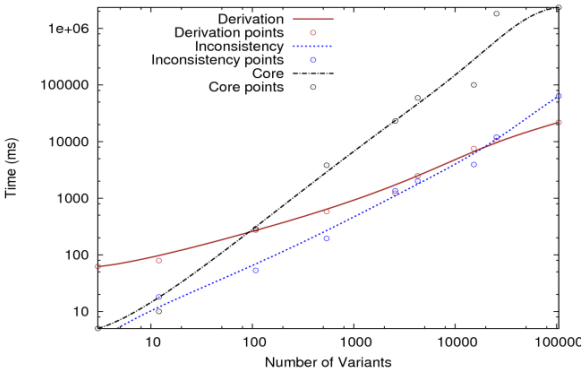


Figure 6.16: Tabular and Graphical representation of the performance of RE-Context.

6.4 Chapter Summary

In this chapter, we have applied our proposed modeling and reasoning framework on two systems intended to operate in and reflect varying context. The first one concerns a Smart Home for patients with dementia problems and the second concerns a Museum-guide system. We have organized a lab session for each of these two systems inviting a group of requirements analysts to apply our framework. The sessions enabled us to evaluate our proposed modeling and reasoning framework. We have reported the feedback we got from the analysts and the observation we made during the sessions. This included the limitations of our approach and the areas that need further research. Moreover, we have applied our reasoning mechanisms on the model developed during the sessions and reported the results. We have also showed and discussed the performance of our developed automated support tool to test its scalability for different goal model sizes.

Chapter 7

Conclusions and Future work

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 a human's living environment to transparently meet his needs [111]. A core element of these emerging paradigms is a varying context. Context plays a major role in requirements. It can influence the set of requirements relevant to a system, what alternatives the system can adopt to meet these requirements, and how good each alternative is.

Most RE research ignores the variable nature of software systems context and an RE approach tailored to systems living in and reflecting varying contexts is crucially needed. This thesis aimed to develop a RE modeling and reasoning framework that reduces the gap between requirements and context. It uses goal model for requirements analysis and provides constructs to capture the relationships between goal models and context. In this way, we answer the *When/Where* of requirements besides the *Why* that traditional goal modeling answers.

7.1 Summary of The Thesis

In this thesis, we have proposed a RE framework for modeling and analyzing requirements of systems living in and reflecting varying context. We adopted Tropos goal model as an intentional ontology that captures the variant ways to satisfy stakeholders' goals and the qualities of each of such ways. Context has influence on stakeholders' goals and choices to meet such goals. Capturing this influence is essential for a final software that meets user requirements in multiple contexts. Software has to reflect humans' adaptation to context, when reaching their goals, in order to derive suitable functionalities to execute.

We suggested to weave between goal model variants and the context in which the system may operate. To this end, we defined a set of variation points at Tropos goal models. At these points, context may intervene to take a decision about the variant to adopt. Specifying context at the variation points avoids us the enumeration of the whole set of goal model variants and the specification of context for each variant separately. Obviously, the specification of context for enumerated variants is time consuming, hard, and error-prone process and the use of variation points is a way to make such process more efficient.

Context plays a major role in the derivation of applicable goal model variants. It may be a factor in activating a variant, i.e., certain contexts could make necessary to meet a set of goals and execute a set of tasks. From the other hand, certain contexts may have to hold to make a variant adoptable. In other words, adopting a certain way to reach the activated goals and execute the activated tasks may require a certain context to hold. The goodness of each variant, from the perspective of a quality measure (softgoal), is not always absolute. It, however, can be context-dependent. In certain contexts, a variant may be of good quality and in others may not be.

Context has been classified into three types: *Activation*, *Required*, and *Quality* contexts. Each of these types is represented at different set of goal model variation points. The activation context of a goal model variant is the conjunction of contexts specified at the variation points of the types (i) Root goals and (ii) AND-Decomposition. The required context of a goal model variant is the conjunction of contexts at the variation points of the types (i) OR-Decomposition, (ii) Means-end, and (iii) Actors dependency. The quality contexts are those specified at the contributions link between the nodes of a variant and softgoals.

The categorization of context into these categories (Activation, Required, and Quality) allows us, amongst other things, to query the model and answer various questions such as: in a certain context, are there any requirements to meet? Is there any possible way to meet the activated requirements? what is the quality of each of such ways. In other words, it allows for a systematic derivation of the goal model variants in multiple contexts.

Context is analyzed through a hierarchial analysis using a set of modeling constructs that we have proposed. This analysis allows for a systematic identification of the facts in the environment to monitor and the way these facts are composed to judge if an analyzed context holds. The context analysis is analogous to goal analysis. While goal analysis allows us to systematically identify alternative sets of tasks that the system may execute to reach the analyzed goals; context analysis allows us to systematically identify alternative sets of facts to monitor in order to judge if a context holds.

A set of mechanisms for reasoning about contextual goal models have been developed. The reasoning mechanisms proposed fall into two categories. The first one is to detect errors the model may contain, i.e., the consistency of the model. The errors we have been interested in are those concerning the consistency of contexts specified for goal model variants and the conflicts between goal model executable processes (tasks) manifested via conflicting changes on context. The second category of reasoning mechanisms concerns the systematic derivation of goal model variants. The derivation we have been interested in is that for deriving the variants fitting to given context and user priorities at runtime, and that for deriving, at design time, the variants leading to a system developed with minimum costs and operable in all analyzed contexts.

We have developed a prototype CASE tool, called RE-Context, that establishes the reasoning algorithms we have developed. We have formalized contextual goal models using Datalog to generate the variants to goal satisfaction. Datalog has been also used to perform several other utilities reasoning such as generating the variants with conflicts between their tasks and generating the variants developed with minimum costs. We have transformed context hierarchy into a boolean formula and used SAT-Solver to reason about the consistency of contexts. We have developed techniques that exploits the nature of our model and gave guidelines to reduce the complexity and enhance the scalability when reasoning about very large models.

We have applied our modeling and reasoning RE framework on two systems intended to operate in and reflect varying contexts. The first system is a Smart Home for enhancing the life of people with dementia. The second is a Museum-guide for assisting visitors of a museum and ensuring the visitors' behaviors against the museum rules. The framework was evaluated based on these two systems and the evaluation results have been discussed .

7.2 Generality of The Approach

Although we have proposed a RE framework that is goal-oriented, we remark that most of the concepts and mechanisms included in this thesis are, potentially, re-usable in and capable of being integrated with other RE modeling approaches. In principle, models that allow for variants could be contextualizable and our proposed framework has a chance of being adapted and re-used for them. Moreover, the context analysis proposed in this thesis is not necessarily restricted to contexts specified at goal models. Rather, it is usable to analyze contexts regardless the variability model that we need to contextualize. In this section, we briefly discuss a version of our modeling framework applied on a variability model that is Feature Model [100, 99].

Feature models capture the commonality and variability of a family of (software) products. The main building block of feature models is *feature*. A feature represents a system characteristic at certain level of abstraction from the perspective of certain stakeholder. For example, feature model may include features understandable by non experts, or those understandable by designers, programmers, and so on. A feature model captures the mandatory and the optional features that a final product may include. It represents a space of variant configurations of features towards establishing the root feature. Each variant represents a possible software product variation.

7.2.1 Running example

Let us consider a software development company that wants to develop a software for visitors in shopping centers. The software is going to communicate with visitors through their PDAs to assist them when they need help, to ensure the adherence of a shop rules, and to facilitate the obtaining of their feedback about the quality of services/products offered at a shopping center. The company wants to build the software in way that makes it systematically customizable to different users, PDAs, shopping centers and so on. For this reason, the design has to consider multiple variants of the system-to-be, multiple contexts in which the system may operate, and the relation between variants and context.

The functionality of giving announcement to visitors concerning parking rules, such as parking in inappropriate places or parking when closing time is approaching, is needed for those shopping centers that have dedicated parking places. Alarming visitors about parking in an appropriate place is critical when there are, usually, few free places for staff/handicaps to park in. Otherwise this functionality is not so critical. Notifying visitors about closing time approaching is needed if the parking place closes at certain time. Such notification is not needed if there is no closing time restriction for the parking place dedicated to the shopping center and the parking is opened all the time.

The system aims to enable visitors for searching and getting explanations about products in a shopping center. Such features are the core of the system and should be applicable for any shopping center. Searching a product can be done by typing the code of the product if the products of the shopping center are categorized and coded. Otherwise, searching the product can be done interactively via clicking, if the used PDA has a touch screen, or via voice recognition when visitor is able to train the system on his voice. Giving explanation about searched products can be done via video presentation or via images and texts.

Obtaining visitors' feedback is a feature that some shopping centers may ask for. Obtaining feedbacks can be done via different interaction methods. A visitor can leave a voice message using his PDA as a recorder. A visitor can be shown an option box to rank a service he used or a product he purchased. Alternatively, the visitor may type his feedback as a text in case his PDA has comfortable keyboard.

7.2.2 Contextual feature model

Following principles similar to those explained in this thesis, we may enrich feature models with the context dimension. Enriching feature models with an explicit notion of context allows us to systematically derive product variants for multiple contexts. However, context is only one criteria to reduce the space of variants that a feature model represents. It reduces the space of variants towards more systematic decision about what features to support. Other criteria, such as costs, may help us to systematically reduce the space alternative even more. Moreover, context may allow us to answer several important questions such as: what are the possible variants of a product in a given context? what are the features that are mandatory, not needed, or optional in a given context?.

The model shown in Fig. 7.1 represents a classical feature model where context is not represented explicitly. The model reflects the PDA shop assistant system described in the previous section. It represents the possible configurations of the system in terms of mandatory, optional, alternative features and their possible composition to establish the root feature "PDA shop assistant".

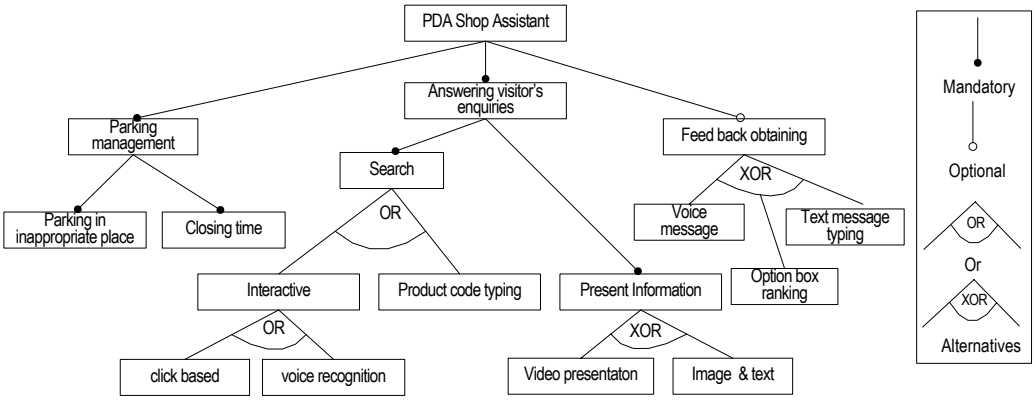


Figure 7.1: Shops assistant Feature Model.

The feature model shown in Fig. 7.1 does not contain an explicit notion of the context. This could make it, especially for systems intended to operate in varying contexts, less expressive and make the process of product derivation less systematic. For example, the feature "parking management" may be useless in a context like "the shop does not have a dedicated parking place". The feature "alarm about parking in inappropriate place" may become optional in a context like "there are often a plenty of places to park in". The feature "alarm about closing time" become useless if the context "the parking place does not close" holds. Moreover, context may influence the adoptability of each sub-feature in OR and XOR decomposition. The feature "click-based" is adoptable if the context "user's PDA has a touch screen". Figure 7.2 shows a contextual feature model that captures the relation between feature model variants and context. We give the description of the contexts annotated on it in Table. 7.1.

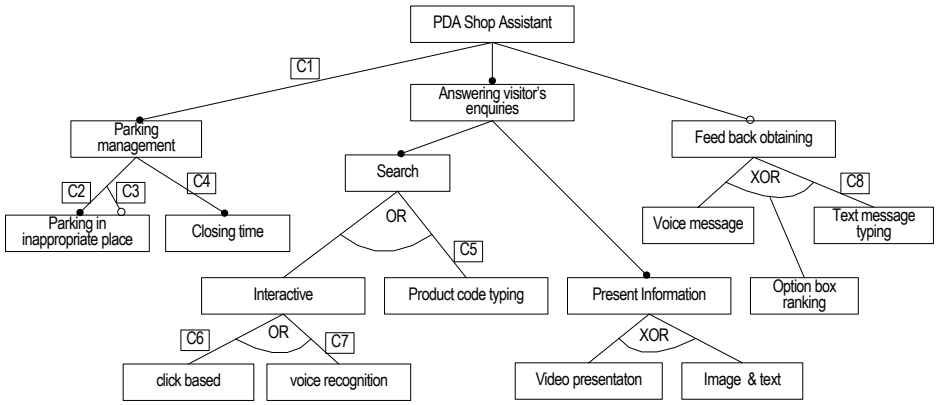


Figure 7.2: Contextual Feature Model example.

Our proposed context analysis can be similarly used to analyze contexts specified at a contextual feature model. The constructs we have proposed are not restricted to goal model as we have already mentioned. The reasoning for context consistency, the derivation of variants for a given context, and the minimum cost variants derivation are reusable in an almost straightforward way. The reasoning for conflicts needs the enrichment of feature model with parallelism and sequence operators and the effect of leaf features on the objects in the system environment. The derivation of variants that reflect the stakeholder priorities may be doable by enriching feature models with more modeling constructs such as softgoals. However, although this is theoretically feasible, further research is needed to establish it.

	Description
C_1	the shop has a dedicated parking place
C_2	the number of people who use the parking place is often high and staff/handicaps may not easily find parking place
C_3	there is often free parking places
C_4	the parking place closes at certain hours
C_5	the products in the shop are categorized and given codes
C_6	the PDA used has touch screen
C_7	user has good expertise in new technology and knows how to train system on his voice and shop is, usually, not noisy
C_8	PDA has comfortable keyboard

Table 7.1: Fig. 7.2 contexts descriptions.

7.3 Towards a Unified Framework for Contextual Requirements

In emerging computing paradigms, such as pervasive, ubiquitous, and ambient computing, context is a main factor in determining what requirements to meet, what options are possible to meet them, and how good each option is. This impact of context is on both the users and the system. From the other hand, the system execution may involve interaction with context. This interaction may cause changes in the context in order to meet requirements. We have studied such mutual influence between context and requirements at the goal level as an early and essential step toward system operating in and reflecting varying contexts. However, recent works have also addressed relatively similar problem at different abstraction levels using different RE approaches. Here we resume our proposed contextual goal model besides two other works that treated the role of context in requirements:

- **Contextual goal models**: the goal-based analysis elicits different alternatives to satisfy a goal, but it does not explicitly specify which alternative should be used for a particular case or context. Supporting alternatives without specifying when to follow each of them raises the question "why does the system support several alternatives". On the other hand, the consideration of different contexts that the software has to adapt to without supporting alternatives leads to the question "what can the system do if the context changes?". Our proposed contextual goal model aims to reduce the gap between the variability in requirements and that in the context. Context is a main factor in de-

ciding the requirements to satisfy, how and how well to satisfy them. Context influences the requirements early at the goal level, as it influences human intentions and choices that are a main source of software requirements.

- **Contextual feature models**: features are characteristics of the system, and feature model represents the variability of these characteristics for configuring a family of software products. Context influences the set of features to be included in a software product variant. Considering context at the design time can model a feature as mandatory or optional, whilst at the runtime context needs to be considered when switching to an alternative feature. As we have shown in the last section, feature models can be enriched with context towards more systematic derivation of software variants. A recent work by Hartmann et al. [101] studies the relation between context and features to support the engineering of software supply chains. This work is in line with our work on the relation between context and variability at the goal level and our last section discussion.

- **Monitoring and switching problems in context**: Salifu et al. [97] apply Problem Frames approach to analyze different specifications that can satisfy the core requirements, under different contexts. The relationship between contexts, requirements, and the specification (machine) are represented by a problem description. Alternative problem descriptions corresponding to different contexts are elicited to identify variant problems. Variant problems are variations of the original problem adapted for a particular context. The specifications to the variant problem are then composed into a context-aware system. A change in context that violates the requirement triggers a switching action to an alternative specification for restoring the satisfaction of requirements. Here there is a clear distinction between the system and the world perceived as its context. The way that the system modifies the context is clearly described.

7.3.1 Integrated model for contextual requirements

Each of the resumed RE approaches covers different aspects of the relationship between requirements and contexts. Goal models capture stakeholder needs and intentions [10] at a time when variability of features in a product line-to-be has not been conceptualized. Relating goals to solution-oriented features leads to a requirement traceability problem [102].

Problem Frames approach makes explicit the distinction between the Requirements (R), the World (W), and the Specification (S). They are related by the entailment relation $W, S \vdash R$. Problems Frames approach captures such a structural relation of a problem more explicitly than both goal models and feature models [112]. However, the Problem Frames approach has the notion of a "variant problem", but it does not natively support a hierarchy of variability as goal and feature modeling approaches do.

Besides its role of giving a rationale to features in the solution space and constraining, at the intentional level, the variability in problem frames, goal modeling can also represent quality requirements as softgoals that cannot have a clear-cut satisfaction criterion. The different requirements alternatives may contribute differently to reaching these softgoals. Moreover, user preferences over alternatives might be expressed by prioritizing the quality measures, i.e. softgoals [62, 92]. In Fig 7.3, we summarize the contribution that each of these approaches can provide to the others, and their relations with context.

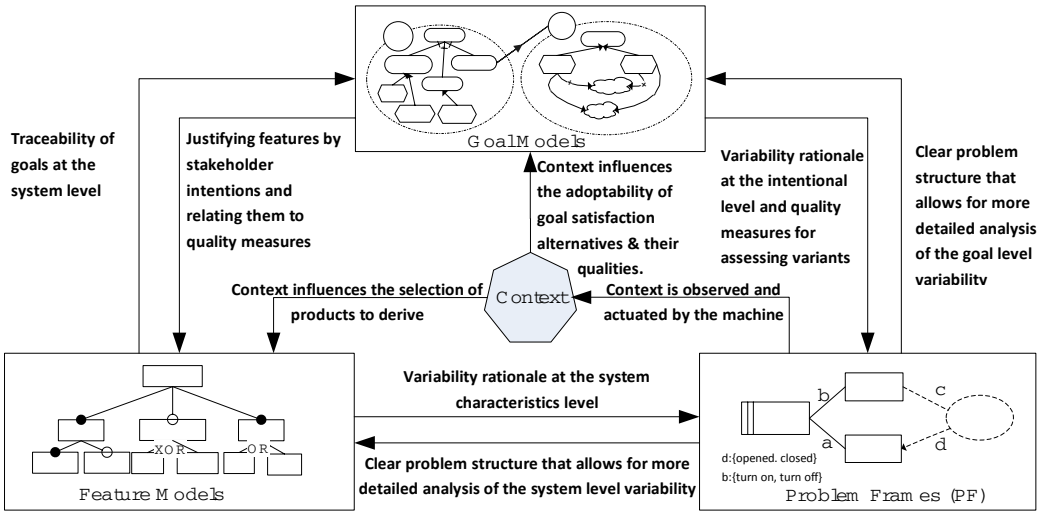


Figure 7.3: Contextual requirements: unified framework

7.3.2 Benefitting from the integration: an example

The integration of the three mentioned approaches has the potential for better expression of and reasoning about the requirements of systems living in and reflecting varying contexts. For instance such problems as conflicts between the system requirements on sharing the context objects can be detected

and resolved early on. To illustrate this, in Fig 7.4, we sketch an example of a "smart home" an automated adaptable living environment that supports patients with dementia as we explained in Chapter 4.

In this sketch the system might need to communicate with the caregiver and patients' relatives (see goal model in Fig.7.4a). Since such communication can be required for different goals that are not alternatives, it may happen at the same time and for different intentions (e.g., to manage the patient's anxiety, and to arrange a social meeting). One way to establish the communication is by making a phone call (shown in Fig.7.4b). If phone is to be used for all communications, this may cause a conflict on this shared resource. Such a conflict can be easily detected when problem frames are used to depict the interaction between the system and its environment (Fig.7.4c).

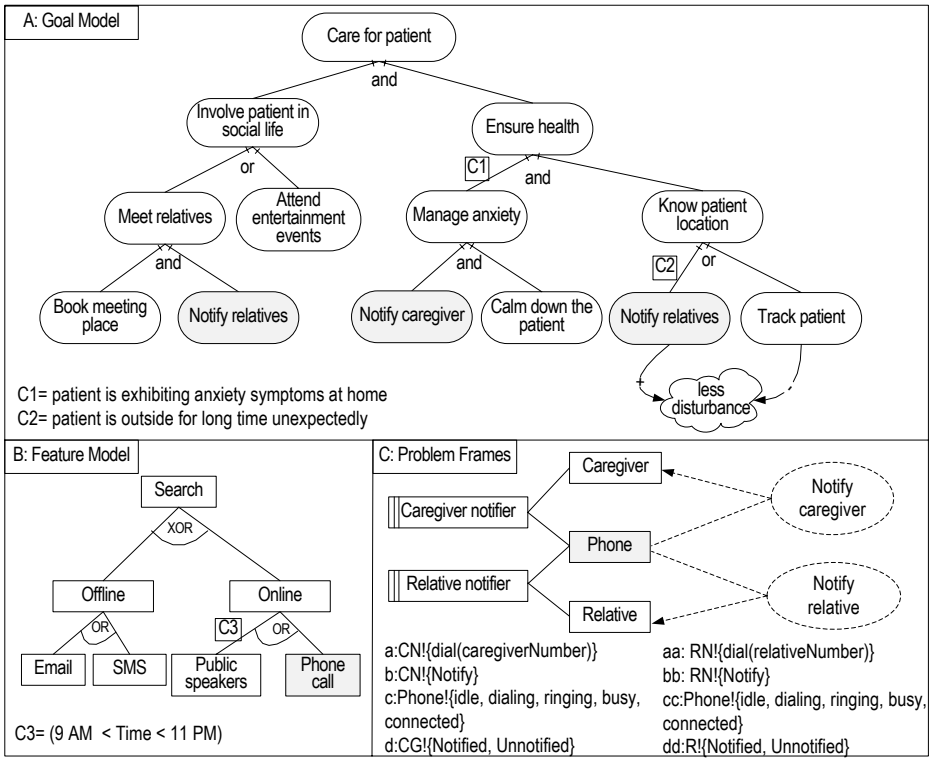


Figure 7.4: Modeling requirements via the unified framework.

Based on this example, we show how each of the three discussed approaches contributes to detection and resolution of such a conflict.

- *Problem frames* have a clear distinction between the physical environment elements (e.g., phone) and the way the system interacts with

them. This clear distinction helps the detection of potential conflicts on a shared element (i.e. exclusive use of the phone). Worth noting is that in order to ascertain that sharing of a resource does lead to a conflict, we need to model the behavior of the shared resource.

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

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

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

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

7.4 Future Work

For the future work, we are interested in addressing the limitations of our approach mentioned earlier (see Chapter 6) and other problems such as:

- **Thorough reasoning about harmful requirements interplay manifested on context:** the mutual influence between context and system requirements necessitates an analysis to discover cases in which this influence is problematic. In this thesis, we addressed only one kind of such problems that concerned the object sharing conflicts. Other problems may arise because of such a mutual influence. One of the problems is the cyclic activation of requirements as a consequence of the changes on context the meeting of these requirements leads to. Another problems could be the denial of adoptability of some alternatives as a consequence of context changes the other alternatives lead to.

  Example 19. in a smart home, the system may have to keep two goals satisfied: entertaining habitant and refreshing the air inside home. If the humidity level inside home is above a certain level, the smart home may open the windows to circulate air. This action will, at daytime, increase the light level inside. If, at that moment, the smart home system is showing an entertaining program on TV, decreasing the light level is desirable and the system would close the windows. The smart home will, in a cyclic way, close and open the windows trying to meet the two goals which is obviously problematic. Detecting such harmful interplay, the goals behind them, the other alternatives the system has, the reconciliation between quality and operability, need an analysis wider than the one we have developed in this thesis.

- **Reasoning about context and monitoring requirements:** the system at runtime should collect different environmental data that are needed to verify facts and judge if certain contexts hold. Upon monitoring context the system needs to adapt to it by adopting a suitable requirements variant. This raises new category of requirements that can be called *Monitoring Requirements*: what data the system has to capture of its environment and the way these data are logically composed to describe high level contexts. Monitoring requirements themselves need an analysis that is as complex as the one needed for the functional and non-functional requirements. In this thesis, we have provided a reasoning about the consistency of context specification. An example of other important reasoning mechanisms to develop is that of finding the set of information to monitor (monitoring alternative) with less costs (in time and monitoring equipments) that leads to verify a given context.

  Example 20. in a museum-guide mobile information system, the system may give explanation about a piece of art if the visitor is inside

the room of the piece and is interested in that piece. As a part of the process of showing information, the system may readjust the screen settings into night mode if the level of light is low. Consequently, the functionality of readjusting the screen into the night mode is accumulatively preconditioned by the context $C = C_1 \wedge C_2 \wedge C_3$ where C_1= "visitor is in the piece of art room", C_2= "he is interested in the piece of art", C_3= "the light level at visitor's location is low". If in one museum, the light level in the pieces of arts rooms is low, to conserve them or for decorating reasons, then $C_1 \rightarrow C_3$ and C can be reduced into $C_1 \wedge C_2$. Therefore, there is no need to install light sensors inside the rooms. Suppose that another functionality is preconditioned by $C' = C_1 \vee C3$, then under the same assumption $C_1 \rightarrow C_3$ we can reduce C' into C_3.

- **Managing viewpoints of context:** besides the potential inconsistency between different stakeholders specifications of requirements, that is well studied in the literature (e.g., [113]), the context specification itself might be debatable. We need to manage multiple perspectives (viewpoints) of context since different stakeholders might specify context differently or even in contradictory ways. Categorizing, detecting, and managing, such differences in viewpoints are necessary to have well specified requirements.

  Example 21. considering a context like "it is good weather outside" as a precondition of opening the windows to circulate air inside home, two stakeholder may have different specification for it. One stakeholder may say "weather is good if the temperature is above 15 degrees and it is not windy", another may say "weather is good if it is sunny and not windy". Considering another context like "tourist is interested in attending a cultural event", one stakeholder may say "he is interested if the event conveys new information to the tourist", another may say "if the event presents something related to the tourist's culture". Managing these viewpoints in context specification and inventing policies to reconcile between them is a challenge to be addressed in our future research.

- **Lifelong adaptation to context**: the system has to monitor context at runtime and adapt to it. However, it is desired that the system has a degree of autonomy to evolve over the time and decide how to enhance the way it satisfies user's requirements since not all decisions can be fully specified by designers at design time. In other words, the system after operating in one environment for a period of time, has to know

what requirements and what alternatives fit better to this particular environment.

Example 22. considering a system for managing promotion of products inside shopping malls, for some people/cultures the gender of sales staff is a factor in the success of the promotion in person. We may design a system that has the ability to continually adapt itself to the context it operates in. Deciding the gender of sales staff to select for promoting a certain product for certain visitor's profile in a certain society is a decision hard to be taken at one step. Moreover, the same society may see changes in traditions and attitude over time. This motivates us to enable the system to support a lifelong contextualization ability.

- **Context and security requirements:** most of security requirements approaches (such as Secure Tropos [114]) deal with security requirements that are context-independent. In some cases, context can influence security requirements and we would need to do research in context-dependent security requirements.

  Example 23. taking the museum-guide system that we studied in this thesis, in an emergency situation (such as fire), a visitor will allow a rescue team to know his location and other data needed to guide him to a safe area, while in a normal situation a visitor would have more restricted security concerns.

- **Improving the automated support**: as we mentioned earlier, the automated support developed in this thesis needs optimization. Moreover, other activities that are still manually done needs further automation. For example, in this thesis we presumed that the relations between world predicates formulae i.e., contradictions and implications, are manually given by the analysts. These relations are important to check the consistency of, and the entailments between, the contexts of goal model variants. Moreover, such relations are used as assumptions to optimize the monitoring requirements as we have described in a previous future work point. Defining these relations for small size systems could be doable manually. For larger systems, this manual specification could be error-prone and time consuming. We aim at facilitating this task by automatic discovery of potential relations between contexts and deriving new relations based on the manually specified ones. In other words, we aim to minimize the effort of analysts and ensure the correctness of contexts relations themselves.

Bibliography

[1] A. Finkelstein and A. Savigni. A framework for requirements engineering for context-aware services. In *Proceedings of the 1st International Workshop From Software Requirements to Architectures (STRAW 01)*, 2001.

[2] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. *Future of Software Engineering, 2007. FOSE'07*, pages 259–268, 2007.

[3] P. Oreizy, N. Medvidovic, and R. N. Taylor. Runtime software adaptation: Framework, approaches, and styles. In *Companion of the 30th international Conference on Software Engineering (ICSE Companion '08)*, pages 899–910, 2008.

[4] A. Schmidt. Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2):191–199, 2000.

[5] S.B. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers. Easy: Efficient semantic service discovery in pervasive computing environments with qos and context support. *The Journal of Systems & Software*, 81(5):785–808, 2008.

[6] M. Jackson. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.

[7] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(1):1–30, 1997.

[8] I. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, pages 71–80. IEEE Computer Society, 2008.

[9] A. Van Lamsweerde et al. Goal-oriented requirements engineering: A guided tour. *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering RE 01*, page 249, 2001.

[10] 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.

[11] E.S.K. Yu. Modelling strategic relationships for process reengineering. *Ph.D. Thesis, University of Toronto*, 1995.

[12] 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.

[13] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: The tropos project. *Information Systems*, 27(6):365–389, 2002.

[14] A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

[15] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, 1999.

[16] 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., 1995.

[17] 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, 2008.

[18] K. Yue. What does it mean to say that a specification is complete? In *Proceedings of the Fourth International Workshop on Software Specification and Design IWSSD-4*, 1987.

[19] A. Van Lamsweerde. Requirements engineering in the year 00: a research perspective. *Proceedings of the 22nd international conference on Software engineering (ICSE 00)*, pages 5–19, 2000.

[20] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and JC Burgelman. Scenarios for ambient intelligence in 2010. *The IST Advisory Group (ISTAG).*

[21] PL Emiliani and C. Stephanidis. Universal access to ambient intelligence environments: Opportunities and challenges for people with disabilities. *IBM Journal of Research and Development*, 44(3):605, 2005.

[22] D. Saha and A. Mukherjee. Pervasive computing: a paradigm for the 21st century. *Computer*, 36(3):25–31, 2003.

[23] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, page 11, 2001.

[24] G.D. Abowd and E.D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):29–58, 2000.

[25] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, 1993.

[26] G.D. Abowd, M. Ebling, G. Hung, H. Lei, and H. Gellersen. Context-aware computing. *IEEE Pervasive Computing*, 1(3):22–23, 2002.

[27] P. Dourish. Seeking a foundation for context-aware computing. *Human-Computer Interaction*, 16(2):229–241, 2001.

[28] J. Hong, E. Suh, and S.J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009.

[29] P. Brezillon. Context in Artificial Intelligence: I. A survey of the literature. *Computers and artificial intelligence*, 18:321–340, 1999.

[30] B. Schilit, N. Adams, R. Want, et al. Context-aware computing applications. *Proceedings of the workshop on mobile computing systems and applications*, pages 85–90, 1994.

[31] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. *IEEE network*, 8(5):22–32, 1994.

[32] A.K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[33] A.K. Dey, G.D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, 2001.

[34] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth, TR2000-381, 2000.

[35] S.S. Yau, Y. Wang, and F. Karim. Development of situation-aware application software for ubiquitous computing environments. *In Proceedings of the 26th Annual International Computer Software and Applications Conference. COMPSAC 2002*, pages 233–238, 2002.

[36] S.S. Yau and F. Karim. Reconfigurable context-sensitive middleware for ADS applications in mobile ad hoc network environments. *Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS 2001)*, pages 319–326, 2001.

[37] 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.

[38] J. Krogstie. Requirements engineering for mobile information systems. *In Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ'01)*, 2001.

[39] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. *In Proceedings of the First International Conference on Pervasive Computing*, page 180, 2002.

[40] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 89–101, 1999.

[41] A. Schmidt, M. Beigl, and H.W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[42] W.N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University, 1995.

[43] A. Zimmermann, A. Lorenz, and R. Oppermann. An operational definition of context. *In the Proceedings of the Sixth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 07)*, LNCS 4635:558–572, 2007.

[44] M. Rosemann, J.C. Recker, C. Flender, and P.D. Ansell. Understanding context-awareness in business process design. *In Proceedings of the 17th Australasian Conference on Information Systems*, 2006.

[45] M. Rosemann, J.C. Recker, and C. Flender. Contextualisation of business processes. *International Journal of Business Process Integration and Management*, 3(1):47–60, 2008.

[46] T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In *In the Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management, (UbiComp 2004)*, 2004.

[47] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[48] K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. *In Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 04)*, page 77, 2004.

[49] T.A. Halpin. *Information modeling and relational databases*. Morgan Kaufmann Publishers, 2001.

[50] H.W. Xiao, Q.Z. Da, G. Tao, and K.P. Hung. Ontology based context modeling and reasoning using owl. In *In the Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW 04)*, pages 18–22. IEEE Computer Society, 2004.

[51] T. Gu, X.H. Wang, H.K. Pung, and D.Q. Zhang. An ontology-based context model in intelligent environments. *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004.

[52] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, et al. OWL web ontology language reference. *W3C recommendation*, 10:2006–01, 2004.

[53] H. Chen, T. Finin, and A. Joshi. Using OWL in a pervasive computing broker. *In the Proceeding of the 3rd Workshop on Ontologies in Agent Systems*, page 9, 2003.

[54] C. Simons. CMP: a UML context modeling profile for mobile distributed systems. *Hawaii International Conference on System Sciences*, 40(10):4848, 2007.

[55] Object management group. OCL 2.0 Specification, ptc/2005-06-06.

[56] S.S. Yau and J. Liu. Hierarchical situation modeling and reasoning for pervasive computing. *Proceedings of 3rd Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS)*, pages 5–10, 2006.

[57] R. Darimont and A. van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. *SIGSOFT Softw. Eng. Notes*, 21(6):179–190, 1996.

[58] A.I. Anton. Goal-based requirements analysis. *Proceedings of the Second International Conference on Requirements Engineering (RE 96)*, pages 136–144, 1996.

[59] V. Plihon, J. Ralyté, A. Benjamen, N.A.M. Maiden, A. Sutcliffe, E. Dubois, and P. Heymans. A reuse-oriented approach for the construction of scenario based methods. *In the Proceedings of the International Conference on Software Process (ICSP 98)*, pages 14–17, 1998.

[60] C. Rolland, C. Souveyet, and C.B. Achour. Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering*, 24(12):1055–1071, 1998.

[61] K. Pohl and P. Haumer. Modelling contextual information about scenarios. *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ*, 97:187–204, 1997.

[62] B. Hui, S. Liaskos, and J. Mylopoulos. Requirements analysis for customizable software: A goals-skills-preferences framework. In *Proceedings of the 11th IEEE International Conference on Requirements Engineering*, pages 117–126. IEEE Computer Society, 2003.

[63] A.I. Anton. *Goal identification and refinement in the specification of software-based information systems*. PhD thesis, Georgia Institute of Technology Atlanta, GA, USA, 1997.

[64] A. Van Lamsweerde, R. Darimont, and P. Massonet. Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, page 194, 1995.

[65] C. Rolland, C. Souveyet, and C.B. Achour. Guiding goal modeling using scenarios. *IEEE Transaction on Software Engineering*, 24(12):1055, 1998.

[66] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.

[67] L.L. Constantine and L.A.D. Lockwood. *Software for use*. Addison-Wesley Reading, MA, 1999.

[68] L. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos. On goal-based variability acquisition and analysis. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 76–85, 2006.

[69] A. Lapouchnian. Goal-oriented requirements engineering: An overview of the current research. Technical report, Toronto University, 2005.

[70] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. *Proceedings of the 13th IEEE International Requirements Engineering Conference (RE'05)*, 5:167–176, 2005.

[71] P. Giorgini, F. Massacci, J. Mtlopoulos, and N. Zannone. Modeling social and individual trust in requirements engineering methodologies. *In Proceedings of iTrust 05*, pages 161–176, 2005.

[72] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Detecting conflicts of interest. *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, 6:315–318, 2006.

[73] Y. Asnar and P. Giorgini. Ensuring dependability in socio-technical system by risk analysis. *Proceedings of the 6th European Dependable Computing Conference*, 2006.

[74] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone. From trust to dependability through risk analysis. *Proceedings of the 2nd International Conference on AReS.*, 2007.

[75] Y. Asnar, P. Giorgini, F. Massacci, A. Saidane, R. Bonato, V. Meduri, and C. Riccucci. Secure and dependable patterns in organizations: An empirical approach. *Proceedings of the 15th IEEE International Requirements Engineering Conference (RE 07)*, pages 287–292, 2007.

[76] V. Bryl, F. Massacci, J. Mylopoulos, and N. Zannone. Designing security requirements models through planning. *In the proceeding of the 18th Conference on Advanced Information Systems Engineering (CAiSE 06)*, 4001:33, 2006.

[77] V. Bryl, P. Giorgini, and J. Mylopoulos. Designing cooperative is: Exploring and evaluating alternatives. *In Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS 06)*, 4275:533–550, 2006.

[78] V. Bryl and P. Giorgini. Self-configuring socio-technical systems: Redesign at runtime. *In Proceedings of International Conference on Self-Organization and Autonomous Systems in Computing and Communications (SOAS 06)*, 2006.

[79] A. Siena. Engineering normative requirements. *Proceedings of the First International Conference on Research Challenges in Information Science, RCIS*, pages 439–444, 2007.

[80] A. Siena, N. Maiden, J. Lockerbie, K. Karlsen, A. Perini, and A. Susi. Exploring the effectiveness of normative i* modelling: Results from a case study on food chain traceability. *In the Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE 08)*, 2008.

[81] A. Siena, J. Mylopoulos, A. Perini, and A. Susi. Designing law-compliant software requirements. *In the proceedings of the 28th International Conference on Conceptual Modeling (ER 09)*, 2009.

[82] D.C. Nguyen, A. Perini, and P. Tonella. A goal-oriented software testing methodology. *In the proceedings of the 8th International Workshop on Agent-Oriented Software Engineering, AOSE 07*, LNCS 4951:58–72, 2008.

[83] C.D. Nguyen, A. Perini, and P. Tonella. Automated continuous testing of multi-agent systems. *The fifth European Workshop on Multi-Agent Systems*, 2007.

[84] C.D. Nguyen, A. Perini, and P. Tonella. Ontology-based test generation for multiagent systems. *In the Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS 08)*, pages 1315–1320, 2008.

[85] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. *In Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE)*, 2010.

[86] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Reasoning about agents and protocols via goals and commitments. *In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS).*, 2010.

[87] M.P. Singh. Agent communication languages: Rethinking the principles. *IEEE computer*, 31(12):40–47, 1998.

[88] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High variability design for software agents: Extending tropos. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4):16, 2007.

[89] M. Morandini, L. Penserini, and A. Perini. Towards goal-oriented development of self-adaptive systems. *In the proceedings of the 2008 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 08)*, pages 9–16, 2008.

[90] M. Morandini, L. Penserini, and A. Perini. Modelling self-adaptivity: A goal-oriented approach. *In the Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2008. short paper.

[91] M. Morandini, F. Migeon, M. Gleizes, C. Maurel, L. Penserini, and A. Perini. A goal-oriented approach for modelling self-organising mas. *In the Proceedings of the 10th International Workshop on Engineering Societies in the Agents' World (ESAW 2009)*, 5881, November 2009.

[92] Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. An architecture for requirements-driven self-reconfiguration. In *In the Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE 09)*, pages 246–260, 2009.

[93] F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Software self-reconfiguration: a BDI-based approach. *In the Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2:1159–1160, 2009.

[94] S. Liaskos, L. Jiang, A. Lapouchnian, Y. Wang, Y. Yu, J.C.S. do Prado Leite, and J. Mylopoulos. Exploring the Dimensions of Variability: a Requirements Engineering Perspective. *In the proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 07)*, page 17, 2007.

[95] S. Liaskos, S. McIlraith, and J. Mylopoulos. Representing and reasoning with preference requirements using goals. Technical report, Dept. of Computer Science, University of Toronto, 2006. ftp://ftp.cs.toronto.edu/pub/reports/csrg/542.

[96] M. Jackson. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.

[97] M. Salifu, Y. Yu, and B. Nuseibeh. Specifying monitoring and switching problems in context. In *In the Proceedings of the 15th International Conference on Requirements Engineering (RE 07)*, pages 211–220, 2007.

[98] M. Salifu, B. Nuseibeh, L. Rapanotti, and T. Tun. Using problem descriptions to represent variability for context-aware applications. *In the proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 07)*, 2007.

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

[100] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh. Form: A feature-oriented reuse method with domain-specific reference architectures. *Ann. Softw. Eng.*, 5:143–168, 1998.

[101] H. Hartmann and T. Trew. Using feature diagrams with context variability to model multiple product lines for software supply chains. *In the Proceedings of the 2008 12th International Software Product Line Conference (SPLC 08)*, pages 12–21, 2008.

[102] Y. Yu, J.C.S. do Prado Leite, A. Lapouchnian, and J. Mylopoulos. Configuring features with stakeholder goals. In *In the Proceedings of the 2008 ACM symposium on Applied computing*, pages 645–649. ACM New York, NY, USA, 2008.

[103] S. António, J. Araújo, and C. Silva. Adapting the i* framework for software product lines. *ER '09: Proceedings of the ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling - Challenging Perspectives*, pages 286–295, 2009.

[104] X. Shen, B. Tan, and C.X. Zhai. Context-sensitive information retrieval using implicit feedback. *In the Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, page 50, 2005.

[105] S. Campadello, L. Compagna, D. Gidoin, S. Holtmanns, V. Meduri, J.-C. Pazzaglia, M. Seguran, and R. Thomas. Serenity deliverable a7.d1.1. scenario selection and definition. Technical report, 2006.

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

[107] A. Nhlabatsi, R. Laney, and B. Nuseibeh. Feature interaction: the security threat from within software systems. *Progress in Informatics*, (5):75–89, 2008.

[108] A. Lapouchnian, Y. Yu, and J. Mylopoulos. Requirements-driven design and configuration management of business processes. In *Proceedings of the 5th International Conference on Business Process Management (BPM 2007)*, volume LNCS Vol. 4714, pages 246–261. Springer-Verlag, 2007.

[109] Y. Yu, A. Lapouchnian, S. Liaskos, J. Mylopoulos, and J.C.S.P. Leite. From goals to high-variability software design. In *In the Proceedings of the 17th International Symposium on Methodologies for Intelligent Systems (ISMIS'08)*, page 1. Springer, 2008.

[110] Y. Wang, S.A. McIlraith, Y. Yu, and J. Mylopoulos. An automated approach to monitoring and diagnosing requirements. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 293–302. ACM New York, NY, USA, 2007.

[111] W. Mark. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.

[112] A. Classen, P. Heymans, R. Laney, B. Nuseibeh, and T.T. Tun. On the structure of problem variability: From feature diagrams to problem frames. *In the proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 07)*, page 109, 2007.

[113] B. Nuseibeh, J. Kramer, and A. Finkelstein. Expressing the relationships between multiple views in requirements specification. In *Proceedings of the 15th international conference on Software Engineering*, pages 187–196. IEEE Computer Society Press Los Alamitos, CA, USA, 1993.

[114] H. Mouratidis and P. Giorgini. Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.