Modelling and Analyzing Location-based Requirements: Goal-oriented Approach

Raian Ali, Fabiano Dalpiaz, Paolo Giorgini
University of Trento - DISI, 38100, Povo, Trento, Italy.
{raian.ali, fabiano.dalpiaz, paolo.giorgini}@disi.unitn.it

Abstract. In emerging computing paradigms, such as mobile information systems, there is a strong relation between the system requirements and location. The system may operate in various locations and location can be a main factor in deciding the requirements to meet and the way to meet them. In this paper, we propose a goal-oriented requirements engineering approach to model and analyze location-based requirements. We propose a set of modelling constructs to represent location, show how to capture location-based requirements via location-based goal model, and present a set of analysis and reasoning techniques about the formalized models.

Keywords: Location-based Requirements, Goal Modelling.

1 Introduction

The advances in size, power, and ubiquity of computing, sensors, and communication technology made it possible to develop mobile or nomadic information systems [13]. In such systems, variability of location and system behaviour is a central issue. The system may have to act differently in different locations to reach its objectives. Capturing and analyzing the relation between the variability of system and the variability of location is essential for systems expected to operate in and reflect varying locations and keep users objectives satisfied.

Considering requirements for mobile information systems, location may have a main role in deciding what requirements to meet, how, and how well to meet them. Location, for us, is an organization rather than a geographical area. Location can be seen differently by different actors depending on their goals. In [1, 12], we defined location as: "*the set of available actors and resources that can be employed to achieve an actor's goals*". Our definition considers the content of locations (actors, resources, and the relations between them), and emphasizes that location is not absolute as it is seen from the perspective of an actor that has objectives.

Alternatives are the cornerstone for adaptability in general that includes adaptability to varying location in mobile information systems. A system with one alternative cannot be adaptive. Moreover, supporting different alternatives without selection criteria, such as location, could be unjustified. We aim to weave together the alternatives the system needs to support and the location where each one is applicable.

Software variability is a term commonly used to represent software provided with different behaviors, whose variants can be produced guaranteeing low costs, short time, and high quality [6]. Feature modeling is a well known modeling technique exploited by product line engineering to tailor a product from a family of possible products [7]. Location-based software is expected to select *autonomously* among the different alternatives depending on the location. Lapouchnian et al. [8] propose techniques to design autonomic software based on an extended goal modeling framework, but the relation with the surrounding location is not captured. A variant of this approach is proposed in [9], where the emphasis is on the variability modeling under the requirements engineering perspective, and the classification of the intentional variability while decomposing a goal. In our work, we focus on the variability of location, i.e. the unintentional variability, which influences the applicability and the efficiency of each goal satisfaction alternative.

In the area of context modeling, the relation between context and its use is often missed (e.g. [2], [3] and [4]). There is a complementary relation between the variable behaviors of actors (both human and software systems) and context. When the relation between context and its use is omitted, questions like "*how do we decide the relevant context?*", "*why do we need context?*" and "*how does context influence the software behavior?*" may not be answered. Modeling context information is not a standalone activity, context has to be elicited in conjunction with the analysis we do for discovering the alternative software behaviors. Salifu et al. [5] investigate the use of problem frames to handle variability in context-aware software and its relation to context. In our work, we use goal analysis to elicit the requirements without assuming that requirements are already recognized, we also try to find suitable abstractions to model location and a process to elicit the location model in conjunction with the goal analysis.

In this paper, we propose a goal-oriented requirements engineering approach to model and analyze the location-based requirements. We extend Tropos [10,11] goal model to capture the relation between requirements and location. Goal model is a well established technique to explain the rationale behind software requirements, and our aim is to enrich this rationale by considering location. We propose the location-based goal model to represent the relation between the intentional variability, captured by goal model, and the unintentional variability of location. We also propose a set of concepts to model location, define a process to elicit location model via goal analysis, and define several automated reasoning to check properties on the proposed models.

The paper is structured as follows: in Section 2, we present a motivating example. In Section 3, we discuss how we extend Tropos goal model to capture both the intentional and location variability. We propose a set of modeling constructs to represent location in Section 4. We show how we can elicit the location model using goal-oriented analysis in Section 5. The formalization of our proposed location-based goal model and several kinds of analysis on goals and locations are described in Section 6. In Section 7, we conclude and introduce our future work.

2 Motivating example

Let us consider an example of a mobile information system (hereafter MobIS) [13] that may support different actors (e.g. clients, technicians) in different shopping malls and interact with them through their PDAs. To help a client, the MobIS has firstly to establish a connection to a mall network accessible by the clients. To allow the client to specify a product, the MobIS can adopt different modalities like identification by reading RFID tags or by reading barcodes. Reading RFID tags is adopted if the client's PDA can read RFID and the mall products have such tags. In this case, the MobIS will show a demo which can be interactive when the client has a good expertise in using PDAs and the PDA has a touch screen, while the demo can be a video-like one in the other case.

Product specification can be done interactively by enabling the client to type some of the product parameters or by proposing to the client a list of products to select between. The proposed list can be based on the client interests or current position in the mall. It can be also based on the mall characteristics like listing the products discounted in or specialized to the mall where the client is. To get information, the MobIS might query the mall database, browse the mall web site after formulating the query, or try to coordinate a meeting with one mall technician. The MobIS has also to work on behalf of the technicians for several objectives and one of them is to answer the questions of clients. To answer a client question, the MobIS has firstly to alert the technician and then present the requested information. The alert can be done through sending SMS, vibrating ringtone, or giving a voice command. To follow each of these alternatives, a specific location condition has to be satisfied, and furthermore each of these ways influences differently some quality measures like the efficiency of the alert, and the privacy of the technician. In the case that more than one alternative can be adopted, the MobIS will consider these measures and adopt the alternative that better satisfies the preferred measures.

3 GORE for Location-based Software

Goal-oriented requirements engineering (GORE), mainly adopted by Tropos [10,11] and KAOS [14], models the software requirements as a set of high-level goals, which are refined iteratively into sub-goals or executable tasks. Besides goals, non-functional requirements (softgoals, in Tropos) are those requirements whose satisfaction is not determined by clear-cut criteria. Softgoals play a central role to select between alternatives, representing quality measurements to compare between the different refinements of high-level goals.

Goal analysis can be effectively exploited to develop high-variability software, especially during the requirements engineering and architectural design. The basic idea is to support more than one alternative for satisfying a goal, and then choose one of them autonomously according to specific criteria. The criteria we consider here is *location*. The software may need adopt different goal satisfaction alternatives in different locations. Softgoals still play a main role for selecting the best applicable alternative, i.e. in some location, more than one alternative may be adoptable and the system may apply the one that contributes better to the preferred softgoals.

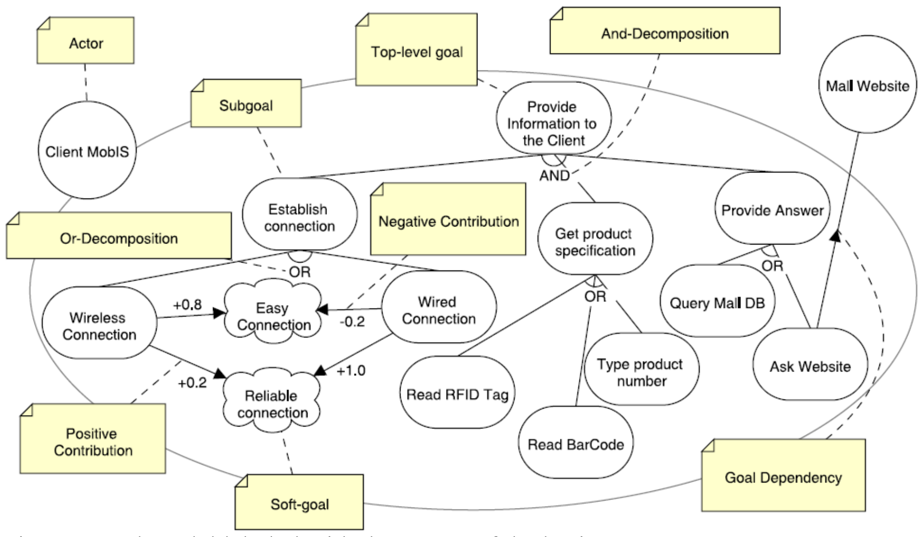


Fig.1. A goal model labeled with the names of the basic concepts.

Fig.1 shows a part of Tropos goal model for the MobIS scenario presented in Section 2. Actors (*Client MobIS* and *Mall Website*) have a set of top-level goals (*Provide Information to the Client*), which are iteratively decomposed into subgoals by and-decomposition (all subgoals should be achieved to fulfill the top goal) and or-decomposition (at least one subgoal should be achieved to fulfill the top goal). In Fig.1, the top-level goal is and-decomposed into *Establish Connection*, *Get product specification*, and *Provide Answer*; the goal *Provide answer* is or-decomposed into *Query Mall DB* and *Ask Website*. Softgoals are goals for whose satisfaction there is no clear cut criteria (*Easy Connection* is an example of such concept), and they are contributed either positively (0, +1] or negatively [-1, 0) by goals: *Wireless Connection* contributes positively (+0.8) to *Easy Connection*, while *Wired Connection* contributes negatively (-0.2) to *Easy Connection*). Goal dependencies represent situations where an actor cannot fulfill a goal by itself, but depends on another actor to fulfill it: the actor *Client MobIS* depends on the actor *Mall Website* for the achievement of the goal *Ask Website*.

In this work, we use Tropos goal models to represent the variable alternatives an actor goal can be satisfied through, and then we specify when (or where) to follow each alternative. We define five types of variation points on Tropos goal model, on which location description (properties) can be attached to help for deciding the adoptable alternative to goal satisfaction in each different location:

1. *Location-based Or-decomposition:* Or-decomposition is the basic variability construct. It allows for alternatives to reach a goal. Each of these alternative may be adoptable in certain locations. For example, the goal *Establish connection* (in Fig.1) can be achieved using *Wireless Connection* only if the mall has a wireless network and the client can access it.
2. *Location-based contribution to softgoals*: the value of the contributions to softgoals can vary from one location to another. For example, the contribution from the goal *Wireless Connection* to softgoal *Reliable Connection* changes depending on the level of the received signal. When the signal coming from the WiFi access point is high, the contribution will be positive (+0.8, for instance), while when the client is far from the WiFi access point and the signal level is poor, the contribution will be negative (-0.5).
3. *Location-based dependency*: in certain locations, an actor might not be able to satisfy a goal using its own strategies. In such cases, the actor might delegate this goal to another actor that can satisfy it. For example, the MobIS can satisfy the goal *Provide Answer* by fulfilling *QueryMall DB*; while if the database is offline and the mall website exists and has a mobile devices version, the MobIS can delegate the goal to another actor (*Mall Website*) browsing that website.
4. *Location-based root goal activation*: an actor, when location changes, might find necessary or possible triggering (or stopping) the satisfaction of a root goal. For example, if the client in interested in a product and behave in way indicates that, MobIS may activate the root goal .
5. *Location-based and-decomposition*: a sub-goal might (or might not) be needed in a certain location, that is some sub-goals are not always mandatory to fulfill the top-level goal in an And-decomposition. For example, to satisfy the goal *Wired Connection*, the MobIS has first to show a descriptive demo to client only if the client is using the system for the first time. This subgoal is not needed if the client is already familiar with the system.

4 Location Conceptual Modelling

Using Tropos concepts, we define the location from the perspective of an actor as: *the set of available actors and resources that can be employed to achieve an actor goals*. We project the location as a set of actors and resources. Actors are described by attributes, the current activity they are involved in, and a set of capabilities they have. Resources are described by attributes, and the current states they are in. Other relations, without a specific semantic, can be found among actors and resources. The metamodel of our location modeling concepts is depicted in Fig. 2. Now we explain our proposed concepts to model location:

- **Actor**: represents an intentional entity (a human or a system) that have strategic goals within an organization. E.g. *Client*, *Technician*, *Web Site*.

- **Resource**: represents an unintentional entity, either physical or informational. E.g. *PDA*, *Product*, *Client profile*.

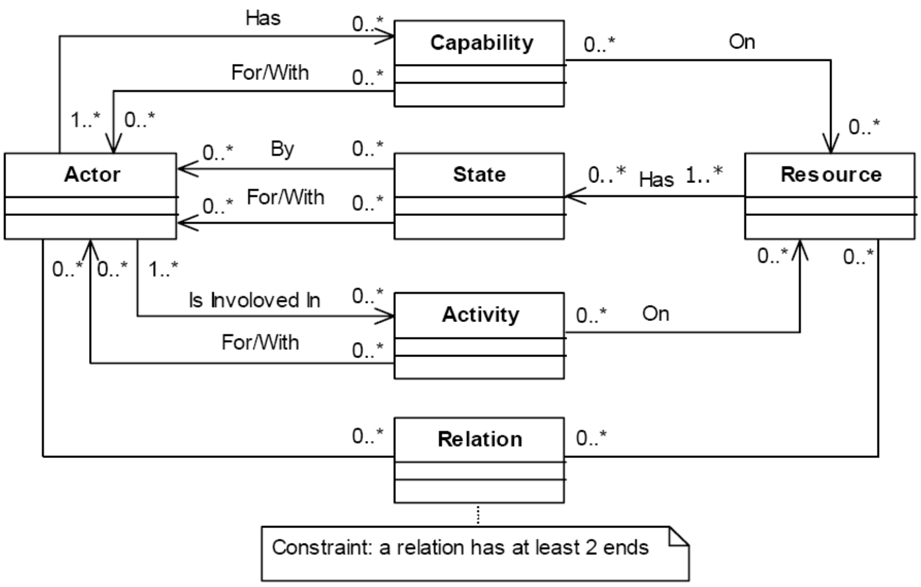
- **Capability**: represents a certain level of expertise of an actor in doing a task. Capability can be unary like *talking, walking, negotiating*. Capability can involve a resource (*Client can use PDAs that have keyboard*), or another actor (*Technician is able to communicate with Clients who are over a certain age*). Capability can be n-ary relation that involves more than one resource or actor. E.g. a location where there is a female technician (the capable actor) that can explain well (capability) about products of the type diet food (resource) to clients who have little scientific knowledge about diet (another actor with capability).

- **Activity**: represents the current work that is being done by an actor. Similar to capability, activity can be unary like *is working, taking a rest, leaving* .It can involve another resource (*is organizing products*) or actor (*is discussing with client*), or can also be n-ary relation that involves more than one resource or actor, e.g. *technician (actor) is interviewing (activity) an expert client (actor) about one product (resource)*.

- **State**: represents the last-known or current status of a resource. A state can be: unary like *printer is idle, is printing, or is under maintenance*. State can also involve a relevant actor, e.g. *the mall touch-screen terminal (the resource) is being used for more than 30 minutes (state) by a client (actor)*, or can also be n-ary relation that involves more than one actor, e.g. *the PDA (the resource) is being used for a voice call(state) with a client(actor) by the technician (actor)*. Another example can be: *the mall touch-screen terminal (the resource) is being used (state) by an unknown profile client to get information about pocket devices (Resource)*.

- **Relation**: is a construct to capture categories of relations that can exist between resources and/or actors without specifying special semantics as we have classified above, *e.g., the client (actor) bought, during the last year, a product of the electronics category (resource)*.

- **Attribute**: is a (key, value) entity that can describe a property of a resource, actor, skill, state, activity, or a relation. We did not depict attributes in the metamodel and the examples above for purpose of simplicity.



- Fig.2. The metamodel for the proposed location modeling constructs

5 Eliciting Location Model

In this section, we show how to capture location properties and model in conjunction with goal analysis. We specify the relation goal model and location at the variation points of goal model. That is to say, the variation points in goal models are enriched with location properties. Doing this we define the location where each goal satisfaction alternative is applicable. Assigning the location properties to the goal model variation points will allow for different automated reasoning on location-based goal model as we are going to show in the next section.

Fig. 3 shows a partial goal model for two system actors (Client MobIS, and Technician MobIS) of the scenarios described in Section 2. The model shows the alternatives to satisfy root goals of each actor, and how each alternative can contribute differently to the softgoals. This model still lacks the specification of the location where each alternative can be adopted. From one actor perspective, the location model includes those actors and resources that influence the goal satisfaction rationale. Goal analysis is performed for each Tropos actor to discover the different alternatives that can be followed to achieve its high level goals and the contributions from those alternatives to softgoals. To weave this analysis with location, we have to decide the set of variation points where the decision between alternatives is location-dependent. At the location-dependent variation points, we define location properties, i.e. description of the location. Each location property, at each variation point, may accumulate a fragment to the location model. In Algorithm 1, we summarize the the elicitation of the location properties and model in conjunction with the goal analysis.

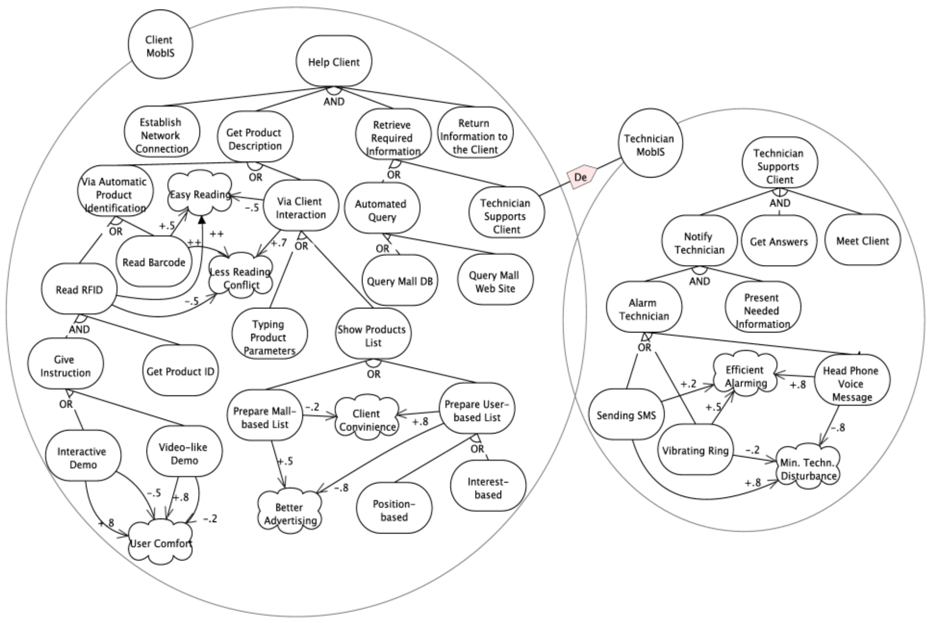


Fig.3. A partial goal model for two actors of the mall MobIS scenario

The algorithm examines all the actors in the goal model (*gm*) passed as a parameter to the algorithm (line 2), and all variation points of their goal analysis are investigated (line 3). We define if the decision at each variation point is location dependent (line 4). If this occurs, location properties at such variation points will be defined and the concepts underlying them (actor, capability, activity, state, resource, relations, and attribute) will be extracted and added to the location model *locModel* (lines 5-8). In details, the location property is defined first (line 5), then it is analyzed to extract the basic location concepts included in it (line 6), the analysis will also extract the relations between the extracted concepts (line 7), and the location model is updated by adding the new elicited concepts and relations (line 8).

```
Algorithm 1 LocationModelElicitation(gm : GoalModel)
1: locModel = null
2: for all Actor In gm do
3:    for all VariationPoint vp  In Rationale(a) do
4:       if IsLocationDependent(vp) then
5:            locProp = DefineLocationProperty(vp)
6:            conceptSet = ExtractConcepts(locProp)
7:            relationsSet = ExtractRelations(locProp)
8:            locModel = locModel ∪ conceptSet ∪ relationsSet
9:        end if
10:   end for
11: end for
```

In Table 1, we show how we defined the location properties on some variation points of Fig. 3 and how we could extract the concepts and relations from each property and construct, accumulatively, the location model. We only consider examples that help to examine the five types of variation points and the different location modeling constructs we identified. The plus sign in the table refers to the newly discovered concepts, while the indention refers to the association between concepts.

Variation point and Location Properties	**Extracted Location Model Fragments**
The *Or-decomposition* of the goal "Via Automatic Product Identification". The first alternative "Read RFID" can be adopted in a mall where the PDAs have RFID readers, and the products have RFID tags. While "Reading Barcode" can be adopted when the PDAs have barcode readers and the products have barcodes.	**+*Resource*: PDA +*Attribute*: Automatic Reading tool(ART) +*Resource*: Product +*Attribute*: Automatic identification tool (AIT)**
The *contribution* from the goal "Interactive Demo" to the softgoal "User comfort". The contributions is positive (+.8) when the client experience in using PDAs is good, and the mall's PDAs have touch screen. The contribution is negative (-.5) in the opposite case.	***Resource*: PDA +*Attribute*: Screen type +*Actor*: Client +*Capability*: Using +*Attribute:* Level *Resource:* PDA**
The *Or-Decomposition* of the goal "Alarm Technician". Alarming through "Send SMS" can be adopted when the technician's PDA is being used for whatever reason, through vibrating ringtone when the PDA is not being used, or through "Headphone Voice Message" when the PDA is not being used and has headphones and these headphones are currently at the technician's ears.	***Resource*: PDA +*State*: Current Use +*Resource*: HeadPhone +*State*: Is In +*Relation*: Has *Resource*: PDA *Resource*: Headphone**
The *dependency* for the goal "Technician Supports Client". The delegation can be accomplished when the client and the technician speaks at least one language in common, the technician can explain the requested information about the specific product, and the technician is close to the client.	***Actor*: Client +*Attribute*: Language +*Attribute*: Position +*Actor*: Technician +*Attribute*: Language +*Attribute*: Position +*Capability*: Explaining *Resource*: Product *Actor*: Client**
The activation of the goal "Notify Technician". The goal will be triggered when the technician is not working, or when the technician is explaining a low profit product to a new client and the new request is for a high profit product issued by an old client who buys always at this period of the year.	**Actor: Technician +Activity: State of Work +Activity: Explaining Resource: Product +Attribute: Profit Actor: Client +Attribute: Reg_date +Relation: Buy Actor: Client Resource: Product +Attribute: Buying_Date**
The *And-Decomposition* of the goal "Read RFID". The subgoal "Give Instructions" is necessary to satisfy the goal "Read RFID" when the client is not familiar with the Client MobIS.	***Actor*: Client +*Capability*: Interacting +*Attribute*: Level +*Actor*: Client MobIS**

Table 1. Eliciting location by goal analysis

6 Formal Analysis for Location-based Requirements

In order to formalize the location properties evaluated against the location model, preliminary step is to define formally the location model, that is, the relevant entities that characterize the considered location. Table 2 presents a partial view of the location, using the Datalog¬ syntax [15]. Datalog (and its variant Datalog¬) can be used as an automated reasoning tool that evaluates the truth value of derived predicates (the intentional database – IDB) against a certain data set (extensional database – EDB). In our approach, we express the current location instance (e.g., the actors and resources that actually exist) as extensional information (*facts*); the location model and location properties are both expressed in the IDB, for they are derived information (*rules*).

Client(A) :-Class(A), Language(A,L), Lang(L), 1 = #count {X,Y : Position(A,X,Y), #int(X), #int(Y)}, 1 = #count {D : RegistrationDate(A,D), *#int(D)}.*
Product(A) :-Class(A), 1 = #count {P : HasProfit(A,P), Profit(P)}, 1 = #count {T : Identification(A,T), ART(T)}.
PDA(A) :-Class(A), 1 = #count {S : Screen(A,S), ScreenType(S)}, 1 = #count {T : Identification(A,T), ART(T)}.
Technician(A) :-Class(A), Language(A,L), Lang(L), 1 = #count {X,Y Position(A,X,Y), #int(X), #int(Y)}, 1 = #count {S : Salary(A,S), #int(S)}, 1 = *#count {St : HasStatus(A,St), State(St)}.*
OldClient(A) :-Client(A), RegistrationDate(A,D), D≤20071231.

Table 2. Datalog¬ formalization representing the location entities.

In the location model formalization of Table 2, a variable A represents Client as it is a class that has at least one language, has one position (expressed as couple representing 2D coordinates), and has one registration date. We exploit the #count aggregate predicate offered by DLV [16] (the Datalog implementation we have chosen) to check cardinalities; for instance, #count {RegistrationDate(A,D), #int(D)} returns the number of RegistrationDate relations between A and D, where D is an integer. A *Product* is characterized by having exactly one profit level (whose admitted values are low, medium, or high), and one identification device (either RFID tag or barcode). A *PDA* is a class that has exactly one screen (whose screen type can be either touch screen or not touch screen) and one product identification device (either RFID or barcode reader). A *Technician* knows one or more languages, has exactly one position, one salary, and one status (busy or idle). An *OldClient* is a client who is registered before the beginning of 2008.

We explain now the analysis of the location properties, which evaluate the EDB with the aid of the location model expressed in Table 2. Location properties are presented in Table 3. L1 and L2 are valid whenever a product can be identified by the PDA that the client is using; L1 considers the products identifiable by RFID, L2 those identifiable by barcode. These properties check that the product and the PDA share the identification system type. L3 and L4 are true when a client has a PDA and a good (not good) expertise in using that PDA model. Properties L5-L7 are preconditions for notifying a technician in different ways: L5 is true when the technician has a PDA and is currently using it; L6 holds if the technician is not using the PDA; L7 is valid if the technician's PDA is idle and the technician has headphones in his ears. Property L8 holds when the technician has a language in common with the client, they are close to each other, and the technician can provide information about the considered product. L9 is valid in two cases: (a) the technician is idle, or (b) the technician is demonstrating a low-profit product to a new client, and there is an old client who could be interested in a high-profit product. L10 is true when the client is not familiar with the MobIS interface.

(L1-L2) The product can be identified by the client via RFID (barcode)
L1(C,P) :-Client(C), Product(P), HasPDA(C,Pd), Identification(Pd,rfid), Identification(P,rfid). L2(C,P) :-Client(C), Product(P), HasPDA(C,Pd), Identification(Pd,barcode), Identification(P,barcode).
(L3-L4) The client has a PDA and is good (not good) in using that model.
L3(C) :-HasPDA(C,Pd), Model(Pd,M), Using(C,M,good), PDA(Pd). L4(C) :-HasPDA(C,Pd), Model(Pd,M), Using(C,M,X), PDA(Pd), X!=good.
(L5-L7) The technician's PDA is busy (L5), idle(L6), idle and the technician has headphones in her ears (L7)
L5(T) :-Technician(T), HasPDA(T,Pd), PDAState(Pd,busy). L6(T) :-Technician(T), HasPDA(T,Pd), PDAState(Pd,idle). L7(T) :-Technician(T), HasPDA(T,Pd), PDAState(Pd,idle), HasHeadphones(T,H), InEars(T,H).
(L8) The technician has a language in common with the client, can provide information about the selected product, and is near to the client.
L8(T,C,P) :-Technician(T), Client(C), Product(P), Language(T,L), Language(C,L), Near(T,C), CanProvideInfo(T,P).
(L9) The technician is (a) idle or it is (b) busy explaining a low-profit product to a new client, and there is an old client interested in a high-profit product.
L9(T,P,C) :-Technician(T), HasStatus(T,idle), Product(P), Client(C). L9(T,P,C1) :-Technician(T), HasStatus(T,busy), Product(P), Explains(T,C,P2), not OldClient(C), HasProfit(P2,low), P!=P2, HasProfit(P,high), OldClient(C1), C1!=C.
(L10) The client is not familiar with the MobIS interface.
L10(C) :-not FamiliarWithMobIS(C), Client(C).

Table 3. Datalog¬ formalization of some location properties referring to Table 2.

We propose various types of analysis to examine software variability against location, and vice versa[1]. A preliminary step consists of evaluating the validity of the location properties at the variation points of the goal model against location.

- *Location-based goal satisfiability*: this analysis is aimed at verifying if a goal is achievable via one alternative in a given location. It searches the goal model and evaluates the location properties at variation points to find the possible alternatives to satisfy a given goal.
- *Location properties satisfiability*: this analysis checks if a given location is compliant with the software goals. It is exploited to identify what is missing in a particular location where some top-level goals have been identified as unsatisfiable by location-based goal satisfiability analysis. Defining the location properties that deny the goal satisfaction will help to define the changes that need to be done in a location to make goals always satisfiable.
- *Preferences analysis*: this analysis requires the specification of preferences over alternatives. Preferences can be specified using softgoals as studied in [17], [18]. Users specify their preferred softgoals instead of the actual goal satisfaction alternatives. The adopted alterative will be the one that satisfies better the preferred softgoals and applicable in a given location. We need this analysis in two cases:

    - When a certain location allows for several alternatives to satisfy a goal: the selection will be based on the contribution (possibly location-based) each alternative has to the softgoals.
    - When a certain location does not allow for any alternative to satisfy a goal: in this case, the results location properties satisfiability analysis leads to some proposals for possible changes on location. The adopted changes are those which make applicable the alternative that better satisfies the preferred softgoals.

7 Conclusion and Future Work

In this paper, we have proposed a goal-oriented approach for capturing and analyzing location-based requirements. We have proposed a set of constructs for the conceptual modeling of location. We have proposed a set of variation points at goal models where location may intervene about the applicable alternatives. We have shown how location models can be elicited through in conjunction with goal analysis. We have also introduced automated analysis techniques on the location-based goal models. In our future work, we will look for a process for eliciting location-based requirements from scenarios and validate the location-based requirements model against user expectations in varying locations. We aim also to develop an architecture to support location-based software at runtime, as location changes and its dynamism has a strong impact on the runtime behavior of location-based software.

[1] For more detailed discussion, please see [12].

References

1. Ali, R., Dalpiaz, F., Giorgini, P.: Location-based variability for mobile information systems. In Bellahsene, Z., Léonard, M., eds.: CAiSE. Volume 5074 of Lecture Notes in Computer Science., Springer (2008) 575–578
2. Yau, S., Liu, J.: Hierarchical situation modeling and reasoning for pervasive computing. Proceedings of 3rd Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS) (2006) 5–10
3. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. PerCom (2004) 77–86
4. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Washington, DC, USA, IEEE Computer Society (2004) 18–22
5. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In: the 15th International Conference on Requirements Engineering (RE'07) (2007) 211–220
6. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
7. Kang, K., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: Form: A feature-oriented reuse method with domain-specific reference architectures. Annals of Software Engineering 5 (1998) 143–168
8. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research (2006)
9. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On goal-based variability acquisition and analysis. Proc. 14th IEEE International Requirements Engineering Conference, Minneapolis, USA, Sep (2006) 11–15
10. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. Thesis, University of Toronto (1995)
11. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3) (2004) 203–236
12. Ali, R., Dalpiaz, F., Giorgini, P.: Modeling and analyzing variability for mobile information systems. In Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L., eds.: ICCSA (2). Volume 5073 of Lecture Notes in Computer Science., Springer (2008) 291–306
13. Krogstie, J., Lyytinen, K., Opdahl, A., Pernici, B., Siau, K., Smolander, K.: Research areas and challenges for mobile information systems. International Journal of Mobile Communications 2(3) (2004) 220–234
14. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Selected Papers of the Sixth International Workshop on Software Specification and Design table of contents (1993) 3–50
15. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. ACM Transactions on Database Systems (TODS) 22(3) (1997) 364–418
16. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Transactions on Computational Logic (TOCL) 7(3) (2006) 499–562
17. Liaskos, S., McIlraith, S., Mylopoulos, J.: Representing and reasoning with preference requirements using goals. Technical report, Dept. of Computer Science, University of Toronto (2006) ftp://ftp.cs.toronto.edu/pub/reports/csrg/542.
18. Hui, B., Liaskos, S., Mylopoulos, J.: Requirements analysis for customizable software goals-skills-preferences framework. In: RE, IEEE Computer Society (2003) 117–126.