UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

Location-based Software Modeling and Analysis: Tropos-based Approach

Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini

April 2008

Technical Report # DISI-08-019

Location-based Software Modeling and Analysis: Tropos-based Approach

Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini

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

Abstract. The continuous growth of interest in mobile applications makes the concept of location essential to design and develop software systems. Location-based software is supposed to be able to monitor the location and choose accordingly the most appropriate behavior. In this paper, we propose a novel conceptual framework to model and analyze location-based software. We mainly focus on the social facets of locations adopting concepts such as social actor, resource, and location-based behavior. Our approach is based on Tropos methodology and allows the analyst to elicit and model software requirements according to the different locations where the software will operate. We propose an extension of Tropos modeling and adapt its process to suit well with the development of location-based software. The proposed framework also includes automated analysis techniques to reason about the relation between location and location-based behavior.

1 Introduction

Advances in computing, sensing and communication technology have recently led to the growth of interest in software mobility. Mobility emphasizes several concerns (space, time, personality, society, environment, and so on) often not considered by traditional desktop systems [1, 2]. Besides computing ubiquity, the 21st century computing [3] is expected to have a core "mental" part: computing systems act on behalf of humans executing tasks without prompting them for and receiving their explicit requests, i.e. computing will realize the concept of agency. Advances in technology do not necessarily imply the easiness of exploiting it, rather more challenges are introduced. Software systems can be given more responsibility, and they can now actively support several decision making processes. Appropriate software development methods and models need to be developed, or adapted, to cope with the new achievable innovative requirements.

Location-based software is characterized by its ability to reason about the surrounding location, which includes the user itself, and adapt autonomously a behavior that complies with the location settings. Consequently, we need to model and analyze the variable locations that users can be part of, and define how location influences software. To adopt one behavior, the software needs to reason on what exists and what can be done, basing its choice on user preferences, cost, time, priority, and so on.

In the area of context modeling, the relation between context and its use is not clearly considered (e.g. [4], [5] and [6]). We believe in the tight complementary relation between variable behavior (both human and software ones) and context. When the relation between context and its use is omitted, we cannot answer questions like " *how do we decide the relevant context?*", "*why do we need context?*" and "*how does context influence variable behavior?*". Modeling context information is not a standalone activity, rather context has be elicited in conjunction with the analysis we do for discovering the alternative software behaviors. Salifu et al. [7] investigate the use of problem descriptions to represent and analyze variability in context-aware software; the work recognizes the link between software requirements and context information as a basic step to design a context aware system.

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 [8]. Feature modeling is a well known modeling technique exploited by product line engineering to derive a tailored product from a family of possible products [9]. A mobile software is expected to select *autonomously* among the different alternatives it supports depending on the location settings. Lapouchnian et al. [10] propose techniques to design autonomic software based on an extended goal modeling framework, but the relation with the surrounding location is not focused on. A variant of this approach is proposed by the same authors in [11], where the emphasis is on variability modeling under the requirements engineering perspective, with a focus on the classification of variability concerns. One of those concerns explicitly refers to location, but it does not define location and how it can be integrated into other models of software variability.

Goal models, mainly adopted by KAOS [12] and Tropos [13, 14] methodologies, represent a paradigm shift from object orientation. While goal-oriented analysis is more natural for the early stages of requirement analysis, object-oriented analysis fits well to the later stages [15]. Goal models begin from a high level goal and start a top-down analysis to discover the more specific sub-goals and tasks for satisfying that goal. Goal models allow for different alternatives to satisfy a goal, but do not specify in which cases each alternative can be adopted. Alternative behaviors and location variability are complementary. Supporting two alternative behaviors without specifying when to follow each of them rises the question *"why do we support two alternatives and not just one?"*. Conversely, considering location variability without supporting alternative behaviors rises the question *"what can we do if location changes?"*.

In this paper, we propose location-based Tropos as a variant of Tropos conceptual modeling framework [13, 14], for developing location-based software. We deal with the social structure level of location, discuss how to model it and how it influences the behavior adaptation of location-based software, i.e. the integration of location with software behavior. We introduce three automated analysis techniques on the proposed models to check software against location and vice versa.

The paper is structured as follows: Section 2 discusses location variability and a variety of conceptual modeling challenges introduced by it, and classifies the main features the location-based software in particular has to support. In Section 3 we study Tropos conceptual modeling framework for location-based software development. In Section 4 we introduce location-based Tropos, proposing modifications on Tropos at both modeling and process levels. In Section 5 we show several kinds of analysis on the new models, and in Section 6 we draw conclusions and present future work.

2 Location Variability and Location-based Software

One main concern of software mobility is the ability to perceive the location where the user is, and then tailor a location-based bahavior to achieve user objectives. Location-based software has not only to perceive the technical details of computing environment (communication protocols, network roaming, data interoperability, and so on), but also the social environment the user is part of. The technical level will certainly be the base to handle the low level aspects of software interoperability, related to the machine level. On the other hand, the social level will be the base for tailoring human-oriented behaviors to achieve user goals. In this work, we focus on modeling the social variability of location and how it can influence software behavior.

Let us consider a passenger with the goal of buying a ticket in railway station. Each specific railway station enables different ways to buy a ticket (e.g., a passenger can buy a ticket through terminals, e-pay, offices, or through passenger assistance clerks when passenger needs help). Each of these different ways requires specific location properties. For example, buying through terminals requires that a free terminal exists, supports one language the passenger knows, and supports one of the passenger's credit cards.

In order to satisfy user's needs and goals, location-based software is supposed to be able to select one appropriate behavior according to the location. The behavior has to be compliant with the current state of the location, considering the availability of resources and the existence of other users. Location may be characterized by different dimensions, such as the degree of expertise each user has (in using resources, and communicating with other users), the availability of resources, and the rules that have to be used to coordinate the use of resources, or regulate the interactions between users. In this vision, the conceptual modeling of software system needs to deal with a variety of challenges, such as:

1. *Modeling constructs*: finding an appropriate set of modeling concepts that can capture the relevant features of location.
2. *Relevancy*: to build a location model, we need a systematic way to decide what has to be modeled, i.e. what is relevant in a location to the target software. E.g. when we model a railway station location, do we need to include passenger's current position, or expertise in using PDAs, in the model? How do we decide that?

3. *Location rules*: location, as a system, will impose rules for the interaction among people and for the use of resources. Rules have to be integrated with the location model and modeled using location constructs. E.g. a railway station might impose the rule that only passengers who are foreigners or over a certain age can ask for assistance, and passenger assistant must help even if this implies stoping less priority activity the assistant is involved in.
4. *Location-based behavior*: for satisfying a user's objective, the current location state allows certain set of behaviors. Modeling the relation between location state and possible behaviors is essential for location-based software. E.g. buying a ticket through e-payment can be done only if the railway station has a network and passenger is allowed to access it, and the passenger's PDA is able to connect to it.
5. *Hierarchial behavior construction*: modeling in a way to avoid "*one location, one behavior*" enumeration, exploiting commonality of both locations and behaviors fragments, and enabling hierarchial construction of location-based behavior. E.g. getting passenger position is a shared software function that is used to guide passengers in all stations where there are terminals or offices.
6. *Behavior evaluation*: based on some payoff functions, each behavior in each location has to be evaluated. We need to model the criteria for evaluating alternative behaviors in variable locations. E.g. when a railway station provides both terminals and e-pay, the software has to decide which one to adopt, and consequently which tasks to do. We need modeling constructs for representing the criteria on which such kind of decisions can be taken.

Location-based software is supposed to support mainly five features (hierarchically represented using feature model in Fig. 1):

- *Location identification*: representing what exists, where the mobile user is, according to a pre-defined location model, i.e. instantiating the location model. E.g. software will receive railway station description and instantiate a railway station model, that reflects the current station, from a template railway station model.
- *Location-based behavior selection*: having an objective, and knowing current location, the software will reason and select a possible, and even recommended, behavior through which the user can achieve his/her objectives. Behaviors include operational tasks that are done by software, and non-operational ones the software assists, or simply asks, user to do. The behavior might include, but not limited to, one of the next three tasks.
- *Location-based information processing*:
  1. *Information request*: software enables users to request location-based information explicitly, e.g. enabling passengers to ask for the train schedule in the current railway station. Other information requests are implicitly made when location changes, e.g. when train is not in the time, certain information has to be presented to passenger.
  2. *Relevant information extraction*: filtering what is relevant, and composing useful information. E.g. when a train is late, but it is not the passenger's train, the warning has not to be shown. Also, when a passenger

asks how to buy a ticket, and has only cash money that are not accepted by railway station terminals, the location-based software will exclude terminal from the possible ways of buying a ticket.

3. *Information delivery*: communicating information to the user in a right way. E.g. notifying the passenger assistant has not to be done by voice message when the assistant is using his/her PDA for a phone call. Also, a demo about using terminals should be interactive, only when passengers have good expertise in using PDAs.

– *Act on behalf of user*: location-based software will represent the user when interacting with other location actors, both in requesting and answering requests, and in using resources available in location. E.g. when passenger asks for help, the help request will be prepared and sent on behalf of passenger, including the information needed by passenger assistant to decide how to accomplish the help.
– *Personalization*: software will behave differently with different users. Software considers user personality as one location mobility dimension. E.g. when both wireless and wired connections are available in a railway station, and the passenger prefers reliable connection, the software will lead passenger to wired connection terminal, and when passenger wants more fast connection, the software will configure wireless one.

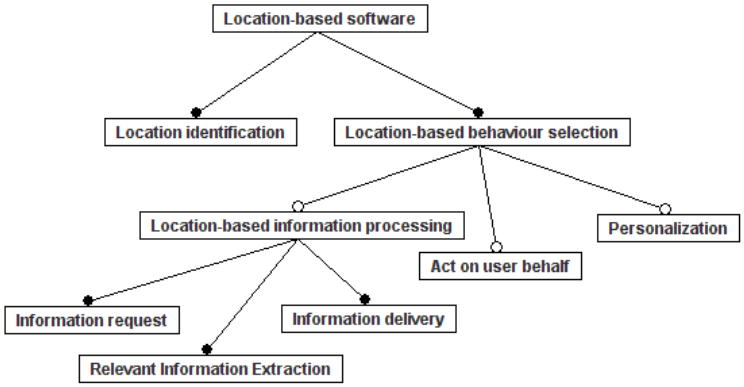

Fig. 1. Feature model for location-based Software

Our goal is to provide a framework that enables "*one software, multiple locations*" development instead of "*one location, one software*" one. The relation between variability of location and variability of software is complementary, and both of them justifies each other. Modeling a software with multiple behaviors without modeling the different locations where each behavior can be adopted, will make the provision of behaviors redundant, since we can simply adopt *a priori* behavior and support only it in the developed system. On the other hand, modeling a variable location, without modeling different behaviors, will make the location model unjustified, since location can not have any effect.

"*What to start modeling first?*" is another crucial question for location-based software modeling. If we take an objective, and start modeling all the alternative behaviors to satisfy it without considering specific location, we will end with a huge number of behaviors. On the other hand, if we start by modeling the location, without knowing how to decide what is relevant and what is not, we will end with a huge model, and still we might miss relevant location assets.

In our approach, we suppose that a location-based software will operate on one class of locations that has a large degree of commonality, e.g. airports, railway stations, libraries, museums, and so on. The domain expert knows how human objectives can be achieved in collaboration with location resources and other actors. The software engineer will analyze how software can take its position within this process, taking some responsibilities a location assistant usually provides. In the next sections, we propose a process by which we start with a rough location model, representing the main actors and resources, and then we analyze the location-based alternative behaviors and refine, in conjunction with this analysis, the initial location model.

3 Tropos for Location-based Software

Our approach is based on Tropos methodology [13, 14], which offers an agent-oriented conceptual framework for modeling both the social environment and the system-to-be. Tropos starts its software development life cycle with the *early requirements* phase. In this phase, the organization (location at the social level) is modeled as a set of actors that strategically depend on each other for satisfying their objectives, then the rationale of satisfying each actor own objectives is modeled. If we take the railway station scenario, the strategic dependency between railway station actors with respect to the goal *Ticket is Issued* will be as shown in Fig. 2. Tropos early requirement fits well to project the social structure of the location at a higher level as a set of actors and resources. Taking into consideration a variable location, this phase will not be sufficiently enough and we will need to adapt it to deal with points such as:

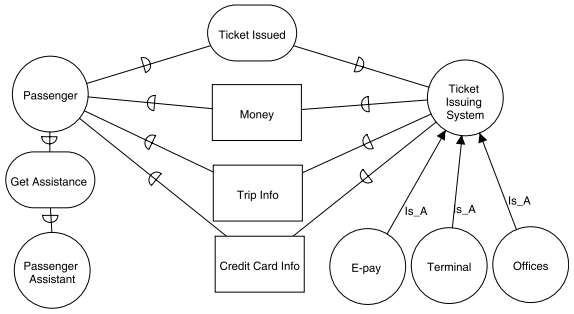

Fig. 2. A strategic dependency model for the railway station scenario.

1. Tropos modeling supposes the existence of all modeled actors (terminal, e-pay, offices, passenger assistant), and this assumption will not hold when we consider a variable location, i.e. location structure is not static.
2. Tropos modeling has to consider actors and resources profiles to deal with several location modeling difficulties:
   (a) Tropos modeling is not able to differentiate between availability levels of actors and resources. In such modeling, we can not differentiate between two terminals, while a terminal that is close to the passenger is more available than a terminal which is far away.
   (b) Dependencies between actors are not required or achievable in every location, and we can not specify this using the rigid form of describing actors and resources in Tropos modeling. *Credit card info* can be required when *Ticket Issuing System* enables payment through credit cards.
   (c) When more than one actor is available to satisfy one objective, there is no way to differentiate between them, and consequently choosing the best. If we consider *Terminal* and *E-Pay* as two *Ticket Issuing Systems* without considering their profiles and matching it with passenger profile, these two issuing ticket systems can not be differentiated.
3. Tropos proceeds, in the next step of early requirements phase, to analyze the rationale of *Issuing Ticket System* to satisfy *Ticket is Issued* goal, and that is not what we need. *Ticket Issuing System* already exists, and we do not need to develop a software for it, rather for *Passenger* to deal with this already functioning system.

In Tropos *late requirements* phase, the system-to-be is introduced as a new actor that takes some responsibilities, already identified in the first phase, and provides an automated solution. The rational of the system-to-be actor is represented by a goal model, starting with a high level goal and finding alternative sets of behaviors that lead to the satisfaction of that goal. Considering location-based software, the rationale of the system-to-be actor is to find suitable behavior for each possible location. In our railway station scenario, the developed location-based software will be for passengers, and passenger assistants as mobile actors. It will work as an automated location expert that operates on a user's computing device, and knows both its user and location social structure.

In a way different from Tropos late requirements, the system-to-be actor is not necessarily assigned an objective that is recognized in the first phase, and is mainly developed to assist users in the already functioning system that is modeled in the first phase. In our example, two system-to-be actors need to be introduced, one for *passenger* and another for *passengers assistant*. The rationale of these two location-based software actors is partially shown in Fig. 3. On this goal-oriented rationale model, that represents well the alternative behaviors of location-based software, we can also highlight several remarks:

1. The system-to-be has particular nature in two points:
   (a) It is naturally decentralized, a location-based software will be assigned for each mobile actor, and might deal with another location-based soft-

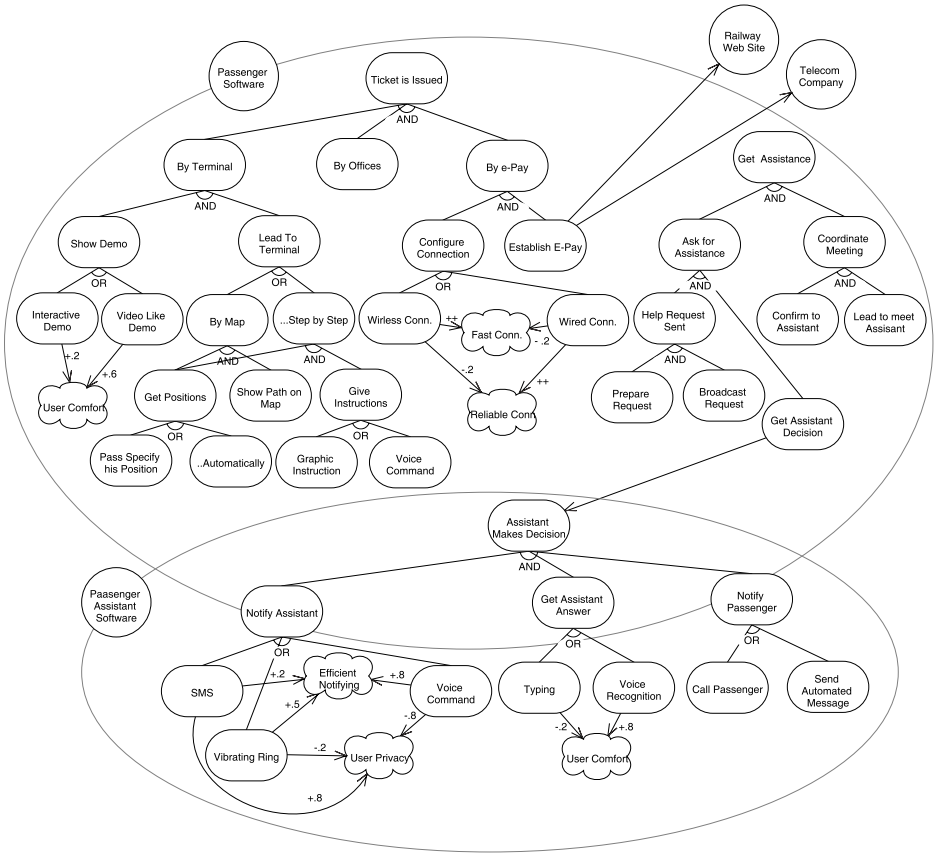

Fig. 3. System-to-be actors goal model for the railway station scenario.

ware assigned to other actors. In our example, we need two location-based software actors, one for *Passengers*, and another for *Passenger Assistants*.

(b) The responsibilities given to the system-to-be actors fall into the categories we have listed in Section 2, and the rationale analysis concerns how to assist mobile users in already functioning system. For example, passengers' location-based software will choose the way that fits to them and to railway station when they need to buy tickets, and it will interact with passengers assistants on behalf of passengers to ask for help.

2. Tropos goal analysis supports different alternatives to satisfy the high level goals. What we need is a kind of location-based goal analysis, that adds location properties to each alternative specifying where it can be adopted. For example, in Tropos goal analysis shown in Fig.3, we do not specify where each of the possible alternatives for having a ticket can be adopted.

3. The contribution to softgoals can be location-based, and not always static. The relation between contribution and location is omitted in the current Tropos goal model. For example, the goal *Wireless Connection* contributes better to the softgoal *Reliable Connection* when passenger is close to wireless network access points, than it does when user is far from it.
4. The autonomous selection amongst alternatives, when more than one are available, needs to be specified based on some criteria. For example, in a railway station where offices are opened, terminals are available, and passenger has the ability to adopt each of these alternatives, we need to specify the decision to be taken. An initial work on this problem has been done in [16], where the decision making is based on preferences that are expressed using softgoals.

4 Location-based Tropos

In the previous section we have addressed the potential and the limitation of Tropos with regards to location-based software development. Early requirements conceptualization, that concerns modeling location, is not sufficiently enough to model variable location and needs mainly to consider actors and resources profiles. We have shown the system-to-be, introduced in the late requirements, as a set of location-based software actors that assist mobile actors to satisfy their needs in location. We have also addressed the gap between Tropos goal-oriented rationale and location, since we need mainly to associate goal satisfaction alternatives with the location where they can be adopted.

When the analyst builds the goal model shown in Fig. 3, a specific assumption about the location, where each of the alternatives can be adopted, could be thought about but was not explicitly represented in the model. Here we discuss five variability points on Tropos goal model that might need location properties to take location-based decision:

1. *Location-based Or-decomposition*: Or-decomposition is the basic variability construct; in current Tropos the choice of a specific Or-alternative is left to actor intention, without considering location properties that can inhibit some alternatives. E.g. the alternative *By Terminal* can be adopted when the terminal is free, has one language in common with passenger, and supports the cash money -in both type (coins, papers) and currency- or one credit card the passenger has. The alternative *E-Pay* can be adopted when there is a wireless network in railway station and passenger's PDA supports WiFi, or when there is wired network with cable-based connection terminals and passenger's PDA has cable connection capability.
2. *Location-based contribution to soft-goals*: the value of contributions to softgoals can vary from one location to another. E.g. the goal *Interactive Demo* contributes positively to softgoal *User Comfort* when user has good expertise in using PDAs, and the used PDA has a touch screen, while the contribution is negative in the opposite case. Also, the goal *Wireless Connection* contribution to softgoal *Reliable Connection* depends on the distance between passenger and WiFi access point to which passenger is connected.

3. *Location-based dependency*: in some locations, an actor might be unable to satisfy a goal using its own alternatives. In such case, the actor might delegate this goal to another actor that is able to satisfy it. E.g. delegation of the goal *Establish E-Pay* to the actor *Railway Website* can be done when that web site enables e-payment using one credit card in common with user's credit cards, and has a mobile device version.
4. *Location-based goal activation*: an actor, and depending on location settings, might find necessary or possible triggering (or stopping) the desire of satisfying a goal. E.g. the goal *Assistant Makes Decision* is activated when the assistant is not doing any particular activity, has one language in common with requesting passenger, and close to that passenger.
5. *Location-based And-decomposition*: a sub-goal might (or might not) be needed in certain location, that is some sub-goals are not always mandatory to fulfill the top-level goal in And-decomposition. E.g. The goal *Show Demo* has to be satisfied when the passenger is not familiar with using terminals.

The goal analysis of location-based Tropos associates location properties to each location-based variability point. In addition, this analysis helps to refine the initial location model represented in the first phase. If we consider the location properties in the above examples, we can identify how the location model of Fig. 2 can be refined. The resulted location model of the railway station scenario, with respect to the location properties above, is shown in Fig. 4. This model adds mainly actors and resources profiles, and also introduces new resources and actors that can influence tailoring location-based behavior.

There are two top-level classes in the location model: actors and resources (Res in the figure). The actor Passenger is characterized by some attributes: spoken Languages (we put it as attribute to simplify the diagram), Position in the railway station, and Expertise in using PDAs. The passenger might have three relevant resources: PDA, Credit Card, and Cash Money. The resource PDA is characterized by an attribute Screen_Type, defining if the PDA has a touch screen or not, and it has a Can_Connect association to the Network it can connect to. A network can be specialized into Cable_NT and Wireless. Cable_NT stands for wired networks, and it is composed of a set of network terminals (NT_Terminals), characterized by a Status that can be free, busy, under maintenance, out of service, and so on. Wireless network is composed of several wireless access points (Access_Point); an Access_Point has the attributes Position and Coverage_Range, used together to compute if a customer is covered by an access point signal. The actor Assistant has a Current_Activity he/she is performing, a Position in the railway station, and spoken Languages. The assistant's relevant resources include only the assistant's used PDA. The actor Railway Website has the attributes E_Pay_Supported, to indicate if e-payments are supported, and Has_Mob_Device_Version, set to true when the website can be browsed by PDAs. The Credit_Card resource class represents the types of credit cards passenger might use, and terminals and railway station website might support. The actor Terminal might support multiple Languages, be in a variable Status, and support Credit Card or Cash Money payment.

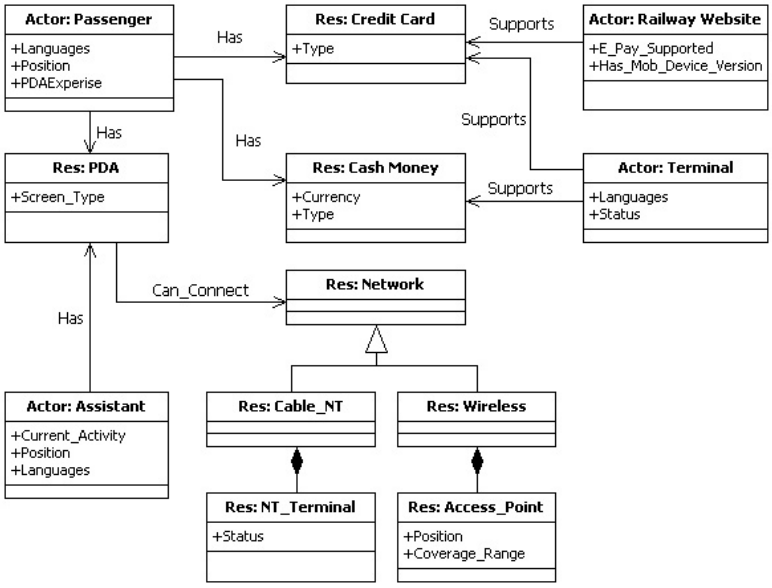

Fig. 4. Location model for the railway station scenario

We describe now our proposed *location-based Tropos process* that leads to the production of our proposed models. We start by **(i)** modeling the social structure of a location class, before introducing the system to-be, using a strategic dependency diagram. In this step, we identify roughly the main location actors and the strategic dependencies between them. Then **(ii)**, this diagram is examined to determine a set of mobile actors, i.e. actors who need location-based software to assist them in the considered class of locations. The next step is to **(iii)** assign a system-to-be actor to each mobile actor, and to model the rationale of these system-to-be actors, using goal analysis. While doing the goal analysis, system analyst **(iv)** decides those location-based variability points, and specifies the location properties at each of them to help selecting between alternatives. Location properties refine the location model, that consists initially of the actors and resources recognized in the first step. System analyst **(v)** will extract new location model constructs (actors or resources properties and relations, new resources or actors) that each location property at each location-based variability point might contain, and keep updating the location model.

By following our proposed location-based Tropos process, we will have three models: the first is the classical Tropos strategic dependencies model, the second represents the location-based rationale of the system-to-be actors (Fig. 3 associated with location properties at the location-based variability points), and the third is the elicited location model (the model of Fig. 4). The metamodel of our proposed extension of Tropos modeling is shown in Fig.5.

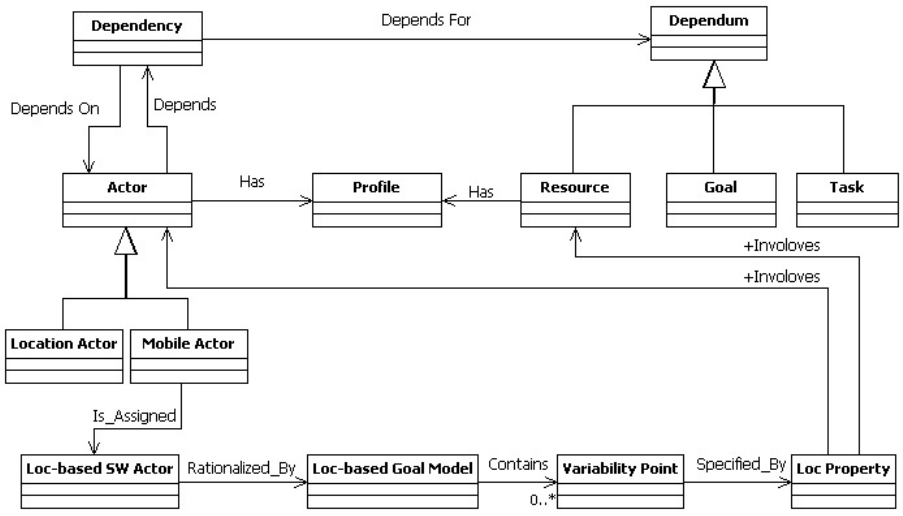

Fig. 5. Metamodel shows the proposed extension of Tropos

5 Reasoning on Location-based Models

We propose various types of analysis for examining location-based software against a specific location, and vice versa. A preliminary step consists of evaluating the validity of location properties at the variability points of the goal model on the current location instance. This step can be done automatically using an automated solver after formalizing the location and location-based goal models. In [17], we used EER diagram to represent location, and we formalized it besides the location properties using Datalog¬ [18]. We used DLV solver [19] to do the reasoning. Here we discuss several kinds of automated analysis on our proposed models:

– *Location-based goal satisfiability*: this kind of analysis is aimed to verify if a goal is achievable through one alternative in the current location instance. The analysis can be performed using the goal reasoning algorithm proposed by Giorgini et al. [20] on the goal model restricted by the evaluation of the location properties. A strategy for evaluating satisfiability follows a top-down approach: starting from a top-level goal, we should check that all (at least one) sub-goals in and- (or-) decompositions can be achieved, or that the top-level goal can be achieved via a *makes* (+1.0) contribution from an achievable goal. For example, in a railway station where there is no positioning system, offices are closed because of vacation, there is a kind of network compatible with passenger's PDA connectivity, and the railway company website supports one of passenger's credit card for e-pay, the algorithm will mark the root goal *"Ticket is Issued"* as a satisfiable goal. The algorithm finds

the alternative *E-Pay* satisfiable, because of the satisfiability of its two And-decomposition subgoals. The alternative *By Terminal* can not be satisfied due to the absence of positioning system, and therefore the unsatisfiability of its and-decomposition subgoal *Lead to Terminal* that can not be satisfied by any of its alternatives in its turn. The alternative *By Offices* can not be adopted, because it requires a location property *Offices are working*, to be satisfied.

– *Location properties satisfiability*: this analysis checks if the current location structure 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. When a goal can not be satisfied, the analysis will identify the denying conditions and suggest ways for solving the problem. For example, in a railway station while passengers have PDAs with only wired connectivity feature, while railway station does not provide cable-based connection terminals, the previous analysis will mark *Configure Connection* as unsatisfiable goal. The reason is that location properties on each of the two connection modalities, wireless and wired, are not satisfied. Location properties satisfiability will reason on what is needed to satisfy the *Configure Connection* goal, i.e. what is needed to satisfy location properties on its alternative behaviors.

– *Preferences analysis*: this type of analysis requires the specification of preferences over alternatives. As shown in [16], preferences can be specified using soft-goals. This analysis is useful in cases like:
  - When some locations allow for several alternatives to satisfy a goal: the selection will be based on the contributions (possibly location-based) to preferred softgoals. For example, in a railway station where both *Wireless Connection* and *Wired Connection* can be satisfied, location-based software will adopt the one preferred by its users. User preferences can be specified over softgoals: when user gives more importance to *Reliable Connection* than *Fast Connection*, the *Wired Connection* alternative will be adopted, while *Wireless Connection* is adopted when user cares *Fast Connection* more than *Reliable Connection*.
  - When certain location does not allow for any alternative to satisfy a goal: the *location properties satisfiability* might provide several proposals about the needed location modifications. The adopted modifications are those leading to better satisfying preferences expressed over soft-goals. For example, in one railway station where *Configure Connection* can not be satisfied due to the absence of wireless network, or cable based terminals, the railway adminstration has to decide between establishing wireless or wired network. When railway station adminstration cares more *Reliable Connection*, a wired network terminals has to be spread over the station, while wireless access points will be installed when *Fast Connection* is more preferred.

6 Conclusions and Future Work

In this paper, we have shown the particularity and importance of modeling location variability for location-based software, and addressed some challenges conceptual modeling faces with this regards. We classified several tasks location based software in particular has to do. To develop location-based software we relied on Tropos methodology, and have shown its potential and limitation for developing such software. We have suggested to modify the conceptualization and the process of Tropos to fit well with location-based software development. By formalizing our proposed models, several kinds of automated analysis, on the relation between location-based behavior and location, become possible. We have shown three kinds of this automated analysis on our proposed location-based models. In this work, we have modeled the social level of a location class as a set of profiled actors and resources; our future work will be towards refining this modeling by finding a set of common concepts that can construct more specifically actors and resources profiles and relations. Consequently, we will also need to find a formal language that is expressive enough for representing the location-based models, and practical for the needed automated analysis.

References

1. 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
2. Pernici, B.: Mobile information systems: infrastructure and design for adaptivity and flexibility. Springer (2006)
3. Weiser, M.: The computer for the twenty-first century. Scientific American **265**(3) (1991) 94–104
4. 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
5. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. PerCom (2004) 77–86
6. 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
7. Salifu, M., Nuseibeh, B., Rapanotti, L., Tun, T.: Using problem descriptions to represent variability for context-aware applications. First International Workshop on Variability Modelling of Software-intensive Systems (2007)
8. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
9. 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
10. 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)

11. 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
12. 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
13. 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
14. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. Thesis, University of Toronto (1995)
15. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. Commun. ACM **42**(1) (1999) 31–37
16. 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.
17. Ali, R., Dalpiaz, F., Giorgini, P.: Location-based variability for mobile information systems. Technical Report DISI-08-008, DISI, University of Trento, http://eprints.biblio.unitn.it/archive/00001351/
18. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. ACM Transactions on Database Systems (TODS) **22**(3) (1997) 364–418
19. 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
20. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. Conceptual Modeling-ER 2002: 21st International Conference on Conceptual Modeling, Tampere, Finland, October 2002: Proceedings (2002)