An UML Profile for the Modelling of mobile Business Processes and Workflows

Michael Decker
Institute AIFB, University of Karlsruhe (TH)
Kaiserstr. 89
76 128 Karlsruhe, Germany
decker@aifb.uni-karlsruhe.de

ABSTRACT

Thanks to the progress in the field of mobile computing hardware (e.g. PDAs, smartphones, notebooks) and wireless data communication standards (e.g. UMTS, WLAN) in the recent years it is possible to provide access to information systems to mobile employees while they are working in the field or are on journeys. Further there are several technologies available to determine a mobile computer's location, e.g. the satellite-based Global Positioning System (GPS). In this article we look at mobile technologies from a process-centric viewpoint: we provide an extension to UML activity diagramms that enables the modeller to express statements concerning the locations where individual activities must or mustn't be performed. These statements are called *location constraints*. We discuss several classes of location constraints, e.g. static or dynamic location constraints and show that location constraints can also be used for UML usecase diagrams.

Location constraints are motivated by several considerations: Since mobile computers have only a small display and restricted means for data input (e.g. no full keyboard) the user will appreciate it if only relevant data is provided by a mobile information system. But location constraints help also to mitigate specific security issues that are associated with the employment of mobile technologies: e.g. devices could get lost or stolen, so it is of advantage if there are location constraints that forbid the access to confidential data at locations where it is not necessary or plausible to access that data.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Workflow management

Keywords

Mobile Business Processes/Workflows, Modelling, UML, Activity Diagrams, Usecase Diagrams, Location-based Services

MobiMedia'09, September 7-9, 2009, London, U.K.

1. INTRODUCTION

A business process (or just "process") is a set of partially ordered activities that have to be executed to reach a particular goal of a company or organization [4]. An often used example to explain this concept is the handling of an order that requires that several activities are performed by several actors, e.g. check consistency of order, check availability of requested item, collect items for order, dispatch shipment, write invoice and check for for receipt of payment. Some activities in a process might be optional, e.g. we might have an additional activity "approve order" for cases where the value of the order exceeds a certain value.

If a business process is automatically executed by a special information system we talk about a workflow. The special attribute of a mobile business process or mobile workflow is that some activities are performed with mobile computers like PDAs, notebooks or smartphones.

The main contribution of our paper is the introduction of an UML profile that extends activity diagrams so that it is possible to express requirements concerning the locations where individual activites can be performed or are not allowed to be performed. This also includes the presentation of a simple location model. It is not only possible to define location constraints in advance but also to model *dynamic* constraints whose concrete spatial extents can only be determined during the runtime of the process. To the best of our knowledge the concept of location-constraint for processes is novel; also we are not aware of other works concerning UML profiles for activity diagrams to model mobile-specific constraints for the execution of a process.

It is important to be able to model location constraints when working with mobile processes since the mobility of the actors is the most prominent characteristic of these processes. Location constraints can be employed to enforce that activities which require the access to confidential information (e.g. customer data) are only performed at locations that are deemed as safe enough and where it is really plausible to perform that activity. For example, an organization could use location constraints to ensure that person-related data can only be accessed with mobile computers when the respective user is on the company's premises; further, there could be a rule that forbids access to documents describing technical details when mobile users stay in a country where espionage has to be feared. But location constraints also provide benefits beyond the scope of security issues: if a mobile information system knows which activities should be performed at which locations it can assist the user by showing only relevant data on the small display of the mobile

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

computer.

The remainder of our article is structured as follows: in section 2 some preliminaries are presented that are necessary for the understanding of the rest of the paper, namely a short introduction to UML that concentrates on activity diagrams (subsection and the introduction of a location model (subsection 2.2) 2.1). The main contribution of our paper can be found in section 3, where the concept of location constraints for mobile processes is introduced; there are several types of location constraints for which also the new graphical elements for UML activity diagrams are introduced. Some examples how the novel UML profile can be applied are demonstrated in section 4. The basic idea of location constraints can also be applied for UML usecase diagrams which is sketched in section 5. Some works that are related to our work are discussed in section 6 before we conclude by giving a summary and an outlook to further work in the final section 7.

2. PRELIMINARIES

2.1 UML Activity Diagrams

The Unified Modeling Language (UML) offers thirteen types of diagrams to describe various aspects during a software development process [14, 13]. It is termed *unified* because it is the result of an integration of several individually developed graphical notations from the domain of software engineering. UML offers *structure diagrams* to model a static view on the system to be developed, e.g. class diagrams to depict the relations between the data structures of a information system or deployment diagrams to show how different software component are distributed on several physical computers. The other class of diagrams are *behavior diagrams* to express dynamic aspects; *Activity Diagrams* are one of these behavior diagrams. The purpose of this diagram type is to model the relationships between individual activities of a process that have to be performed to reach a particular goal.

UML also specifies several layers of meta-models and thus explicitly supports that end users create custom extensions for UML diagrams when the standard diagrams don't meet their requirements. Such an extension is called "profile". In this paper we describe a profile for UML activity diagrams to gain the capability to model mobile-specific aspects of mobile processes. There are other graphical languages to model processes, e.g. the Business Process Modelling Notation (BPMN) or petri nets; however, these languages don't provide dedicated extension mechanisms.

2.2 Location Model

For the sake of simplicity in the article at hand only two dimensions are considered. Especially for indoor scenarios with multi-storey buildings it would be necessary to have three dimensions. However, it would be easy to extend the model introduced in this subsection to cover a further dimension. Further, it is assumed that the mobile user's position can be determined with perfect accuracy.

The location model for the UML profiles provides three ways how location restrictions can be expressed:

- A constraint can be defined by refering to a location, which is a polygon with non-crossing lines within the reference space ("universe"). These locations are modelled as instances of one particular location class. The idea of modelling locations as instances of a particular class or type is borrowed from the Geographic Markup Language (GML) [12], where objects with spatial extents are defined as *features* that belong to a particular *feature type*.

- It is also possible to describe a spatial region of circular shape by defining a center point and a radius. This way to describe a location restriction is especially useful for the case when the current location of a user is known (e.g. by using GPS) and it should be demanded that certain activities are performed within a certain distance.

- In our location model it is possible to attach security labels to locations, e.g. a building could have a clearance of "Secret", while the strongroom within that building has even a clearance of "Top Secret". We can restrict the locations where a activity can be performed by defining a minimum security level that has to be met by the location where the user has to stay when performing that activity.

The location model is depicted as UML class diagram in figure 1: Instances of *AbstractLocationDescription* are employed to make statements about the locations where activities of a process are allowed to be performed or not. This class is an abstract class, i.e. it is not possible to instantiate that class directly, rather one has to create an instance of one of its three subclasses, namely *LocationInstance*, *SecurityLabelInstance* or *CircularArea*.

As depicted in the location model each *SecurityLabelInstance* belongs to exactly one *SecurityLabelClasses*. The security labels are inspired by the classifications used in multilevel security models for mandatory access control like the Bell-LaPadula-model [3]. Each *SecurityLabelClasses* represents a certain thematic category, e.g. a particular product category or research field, e.g. product A or B or semiconductor technology. The label instances that belong to a label class describe how secure or trustworthy a particular location is with regard to that thematic category. For each security class there have to be at least two labels; however, the number of individual labels for each security class might differ. As example we assume that there is a country within the reference space where many companies are established that operate on the market for product A, so espionage has to be feared; therefore the location instance representing that country is assigned to the security label "productA:low".

For *LocationInstances* there are two conditions: Two locations that belong to the same class mustn't overlap spatially. Further, for a given location class the union of all its locations have to completely cover the reference space. These two conditions guarantee that for each point in the universe for a given location class there is exactly one location that contains that point. For example, if we have a location class "City" then instances would be "London" or "Berlin"; another location class could be "Country" with instances like "U.K." and "Germany".

3. LOCATION CONSTRAINTS

3.1 Static Constraints

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

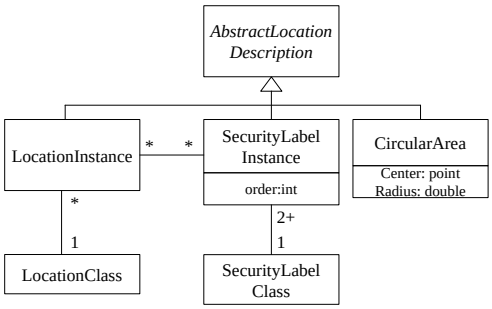

Figure 1: Location Model

A location constraint makes a statement about where the user with his mobile computer is allowed to stay when an activity is performed [6, 5]. It would be possible to formulate constraints concerning other context parameters (e.g. device capabilities, time, network quality) but since the user's location is the most prominent context in mobile computing we consider only location constraints, so we use the term "constraint" as short form for "location constraint".

There are *positive* and *negative* location constraints: A positive constraint states the location(s) where the respective activity is allowed to be performed while a negative constraint makes a statement about locations where it isn't allowed to perform the activity. This attribute is called the *mode* of a constraint.

An orthogonal classification approach is to distinguish static and dynamic constraints. A *static constraint* is a constraint that counts for each instance of the process explained by the activity diagram. This means that such constraints have to be defined during the design phase of the respective process. In contrast to this *dynamic constraints* are derived during the runtime of a process; it is not possible to define them in advance and they belong to a process instance rather than the process schema. That's why they could also be called *runtime constraints* or *instance constraints*. Since they are not simply assigned at design time of a process model there are several methods how dynamic constraints can be defined, which are discussed in the following subsection.

In the first row of the table shown in figure 2 the graphical notation for both positive and negative constraints can be found: To attach a static constraint to an activity in a UML diagram our profile demands that a dotted arrow pointing to a parallelogram is drawn. The parallelogram represents the location instance and is labled with the identifier of the location instance. To distinguish between positive and negative constraints a circle holding the symbol for "equals" resp. "not equals" is drawn on the dotted line.

3.2 Dynamic Constraints

There are three basic ways to obtain dynamic constraints: (1) The constraint can be set by a mobile or stationary user manually; (2) the constraint can be set automatically based on data stored in a backend system; (3) the constraint can be derived automatically based on the mobile user's current location (e.g. determined by a GPS receiver integrated in his mobile computer). For this a rule has to be defined.

UML annotations to express the first two possibilities are

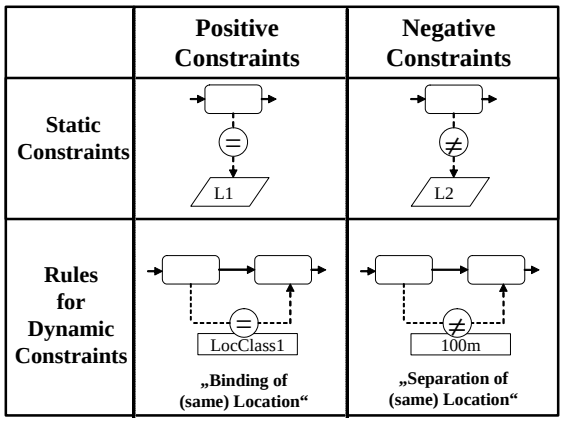

Figure 2: Different classes of location constraints: Positive vs. negative and static vs. dynamic constraints

depicted in figure 3: The activity on the left side (a) has a manual constraint; this is represented as a *matchstick man* connected with a dotted line to the activity. On the dotted line there is a circle that show the symbol for the mode of the constraint. In the box attached to the circle class names are listed to restricted the location instances that can be chosen by the user as constraint. The notation for dynamic constraints based on backend data is similar (b), but instead of the man the *component symbol* from UML deployment diagrams is drawn at the end of the line. Like these examples indicate it is possible to specify several location classes to make a statement which locations can be assigned as constraints.

The third case for the definition of dynamic constraints is the most interesting one: in this case the user's current position when performing a so called *source activity* is determined using a locating system, e.g. the *Global Positioning System (GPS)* or triangulation of signal runtimes between the mobile computer and several base stations. Since a discussion of such locating technologies is outside the scope of this paper the reader is referred to [11]. Based on the current location of the user location constraints for one or more *target activites* are determined during the runtime of the process. To describe how these dynamic constraints have to be derived based on the current location of the mobile user there are so called *location rules* in the process diagram. Such a rule is depicted as dotted line that points from the source activity which triggers the creation of the location constraint to the target activity. It is possible that for a rule a target activity is also the source activity. In the second row of the table depicted in figure 2 examples for rules for the generation of positive and negative constraints can be found. The case of a positive constraint generated by a rule can also be called *binding of location* since this demands that the target activities are executed at the *same* location; the case of a negative constraint is called *separation of location* and says that it is not allowed to perform the target activity at the *same* location where the source activity was just performed. What is considered as the "same" location for both binding and separation of location is defined by the annotation in the box attachted to the circle with the equal

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

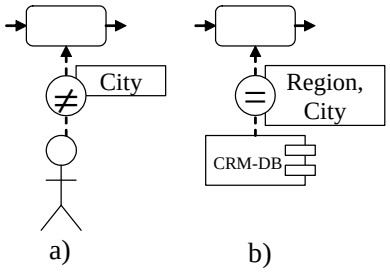

Figure 3: Annotations for dynamic constraints defined manually by user (a) or obtained from a back-end system (b)

resp. the not equal sign. In this box the modeller can either specify a location class or a numeric value in meters that constitutes the radius. If a location class is stated (like "LocClass1" for the positive constraint rule in figure 2) this means that the generated location constraint for the target activities will point to that location instance of the stated class that covers the current location of the user. Since it is demanded that all the location instances that belong to the same location class don't overlap each other and cover the reference space exhaustively there is always a unique location instance for each location of the mobile user. The rule that generates a negative constraint in figure 2 is annotated with the second possibility to define the granularity of what constitutes the same location, namely a radius. This means that the target activities cannot be performed at a point that is less than 100 meters away from the point where the source activity that triggered the rule was executed.

3.3 Shortcuts

In figure 4 some shortcut notations are given for static constraints (a-c) and for dynamic constraint rules (d).

The UML fragment in figure 4a shows two activities that are grouped together by a rectangular shape; this is called "swimlane" in UML parlance. If we assign a location constraint to this swimlane the constraint is applied to all the activities contained within the swimlane.

In the subsequent fragment figure 4b one activity has a static location constraint that points to two location instances. This means that the activity can be performed at either U.K. or USA.

It is also possible that two or more activities that are not within the same swimlane are attachted to the same location constraint (or even the same set of location constraints) as shown in figure 4c.

The fragment in figure 4d shows a shortcut for a rule for dynamic location constraints. As already mentioned above it is possible that one rule has several source activities and/or several target activities. In the example fragment shown there is a rule with two source activities that creates a positive constraint for a single target activity.

4. EXAMPLES

It would be unrealistic to have a mobile process where all forms of location constraints introduced in this article can be found. We therefore sketch several processes from different application domains to exemplify the individual types of constraints.

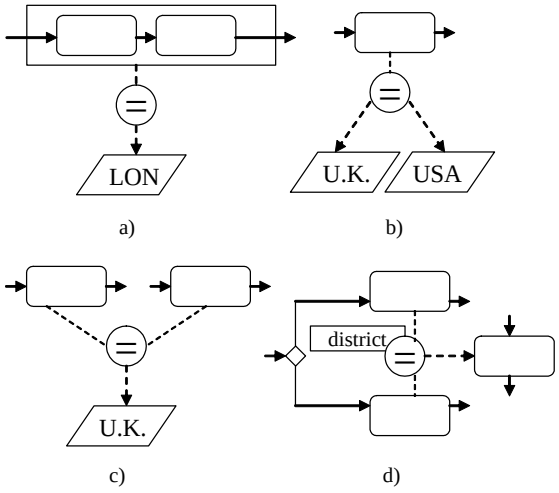

Figure 4: shortcuts notations

In figure 5 a generic process for sending a mobile service technician to a site to perform there some kind of repair work is drawn. One special feature of this scenario is that the site where the repair work has to be performed isn't known in advance. This process could be found in a company whose main business is to perform maintenance work on a road network; but it could also be a process of a company that sends service technicians to customer's residences or premises to fix some technical device, e.g. central heating system or electrical installation. The process is started when a telephone operator at one of the local branches of that company receives a telephone call from a police patrol or car driver that reports a damage on a road segment, e.g. damaged crash barrier, missing road signs, damaged road surface or street lighting. The following activity is *on-site inspection* to find out if there is really a problem; if no problem is found then the next activity is *post processing*, which has to be performed at another local branch of the compay. This activity includes subactivities like writing a report or sending a bill and also includes an evaluation how well the case was handled; because of the latter the policy of the company demands that the post processing is performed in a local branch in another region than the region where the process was started. In the diagram this is expressed by a negative rule with source activity *receive call* and target activiy *post processing*; the location class associated to that rule is *Region*. If the inspector can verify that there actually is a problem at the reported location then the next activity to be performed is *reparation*. For this activity a team of special craftsmen (e.g. electrician for malfunctioning street lighting or welder for broken crash barrier) is sent to the site. When the craftsmen have finished the work the inspector has to visit the site again to ensure that the repair work was performed properly. If this is not the case the process activity *reparation* has to be performed again. Since the road damage can be reported for virtually any location and isn't known in advance a dynamic location constraint defined by a positive rule is employed. This rule says that the location

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

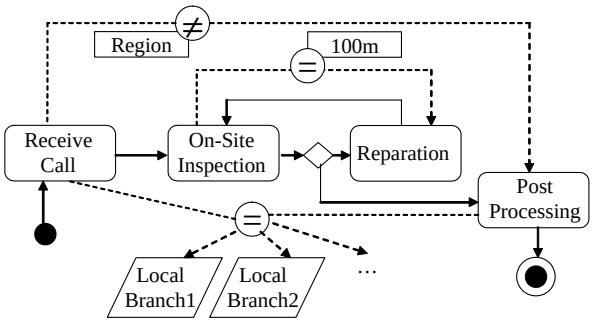

Figure 5: Example process: Mobile service technician

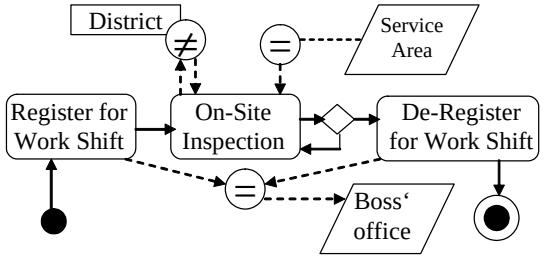

Figure 6: Example process: On-site inspections at different locations

where the activity *on-site inspection* is performed defines the location where subsequent invocations of the activities *on-site inspection* and *reparation* have to be performed.

Another generic example process is shown in figure 6: this process is appliciable when a mobile worker has to perform a series of on-site inspections, but it has to be enforced that each inspection is done at a different place. This could be the case if samples (e.g. soil) have to be taken or a night watchman has to prove with a mobile computer that he actually visited different places at the premises or building he has to watch. The process starts when the mobile worker reports to start the shift. Then he performs a series of actual inspection activities. Because this activity should performed at each location only once, a rule is assigned to that activity that creates a negative district. Source and target activity are the same for that rule. At the end of the shift the worker visits again the office of his superior. There are three static constraints in the diagram: *Register for Work Shift* and *De-Register from Work Shift* are assigned to the boss' office. The activity *On-site Inspection* has not only the rule but also a static location restriction that binds it to the service area.

In figure 7 a document workflow where security labels are assigned can be found. The process is started when a document (e.g. internal report of a military intelligence service) is uploaded to a central information system with a mobile computer. After this the document has to be reviewed before the final assessment can be perfomed. To this last activity a static location constraint based on security labels is assigned: this constraint demands that the activity can only be performed at locations that are classified as "Top Secret"

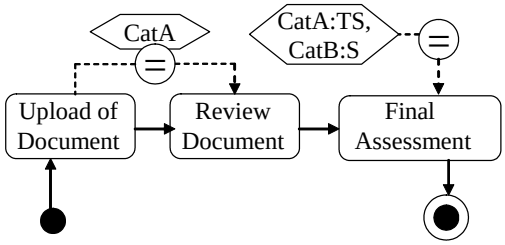

Figure 7: Example process: Document workflow with Security Levels

(TS) according to category "A" (CatA) and at least as "Secret" (S) according to category "B" (CatB). Further, there is a dynamic location constraint represented by a rule with "upload of document" as source activity and "review document" as target activity. This rule assigns a positiv location constraint to the activity "review document" that demands that this activity can only be performed at locations that are classified with at least the same security level according to category "A" (CatA). For example, if a document is uploaded at a location classified as "Secret" it is not allowed to perform the review activity with a mobile computer that stays at a location classified below, e.g. "Confidential".

5. LOCATION CONSTRAINTS FOR UML USECASE DIAGRAMS

Location constraints can also be used to supplement UML usecase diagrams. The purpose of usecase diagrams is to state which functions an information system has to provide. Such a function is called a "usecase" and denoted by an ellipse that contains a textual label, e.g. "access document", "restart system" or "create new customer record". Further, a usecase diagram shows the different types of actors (or roles) depicted as matchstick men; examples for actors are administrator, secretary or software developer. Usecases are connected with lines to those actors that are allowed to invoke that usecase. Further it is possible to assign usecases to individual systems by drawing a rectangle around the respective usecases.

In figure 8 a usecase diagram is shown that is annotated with location constraints: *Usecase 1* has a static location constraint that points to location *loc 3*, i.e. this usecase can only be invoked when the respective actor with his mobile computers stays within that location. Another static location constraint is assigend to the system border that contains all the three usecases in the diagram. This constraint points to *loc1* so that any usecase contained by that system can only be invoked when the current actor stays within that location.

It is also possible to model dynamic location constraints by defining rules. In the depicted diagram *Usecase 2* is the source for a rule that assigns a negative location constraint to *Usecase 3*; this spatial extent of this constraint are defined by that location of *LocclassA* that containts the actor's location when *Usecase 2* is invoked for the first time.

There are also two roles in the diagram, namely *Role 1* and *Role 2*. The first role is connected to *Usecase 1* and *Usecase 2*, i.e. users that are assigned to *Role 1* are only allowed these two usecases. Role 2 is only allowed to in-

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

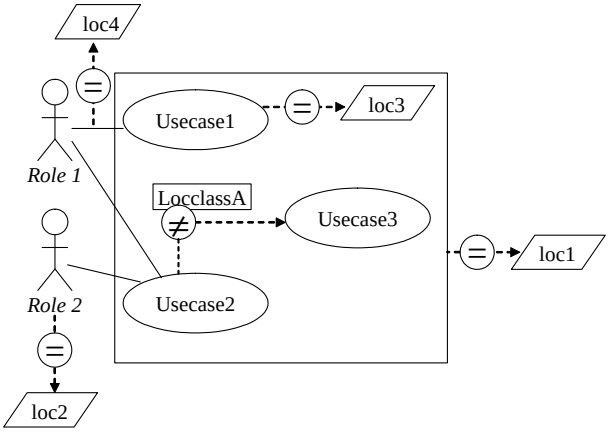

Figure 8: Location constraints for an Usecase Diagram

voke *Usecase 2*. However, since *Role 2* has a static location constraint that points to location *loc2* the permissions assigned with that role can only be used when the user with his mobile computers currently stays within the respective location. There is also a static location constraint pointing to the connection line between *Role 1* and *Usecase 1*. This says that users with *Role 1* can only invoke *Usecase 1* when they stay within *loc4*; however, the right to invoke *Usecase 2* is not restricted by that constraint.

6. RELATED WORK

6.1 Modelling of Mobility with UML

In [2] a UML profile for activity diagrams to model mobility is introduced. Using this profile it is possible to model how objects are moved by activities to other locations. In UML activity diagrams objects can be depiced as input or output of activities, so this UML profile introduces a new stereotype <<move>> for activities and introduces a attribute *atLoc* for objects. There are two notational variants: in the *responsibility centered variant* the explicit statement of the *atLoc*-attribute is used to show the location of an object. In the *location centered variant* there are boxes that represent particular locations. If an object is drawn within a box this says that the object is at the location represented by that box.

Stefanov and colleagues [16] developed an UML profile for activity diagram that is able to supplement the diagram with business intelligence objects. For example, using their profile it is possible to state that an activity requires data from a *data warehouse*, a *data mart* or an *operational data store*.

In [10] a UML profile to model mobility is described. However, this profile extends sequence diagrams rather than activity diagrams. The basic notion of this approach is to use a generalized version of life lines for objects that represent locations; these generalized life lines are depicted as boxes, so it is possible to show that a object stays currently at a particular location.

6.2 Modelling of Mobile Processes without UML

In literature we found some approaches to model mobility in processes that are not based on UML:

In [17] a modelling approach based on "service blueprinting" is introduced. The purpose of this method is to discover potentials for the improvement of inter-organizational mobile processes through the employment of mobile devices with the capability to perform direct data exchange, e.g. use of Bluetooth for direct exchange of documents between a travelling saleman and his customer. To identify activities in a mobile process where direct data exchange would be advantageous the method tries to find activities when an actor isn't in reach of his own stationary computer infrastructure.

Another method for process modelling focusing on mobile-specific aspects is *Mobile Process Landscaping* by Köhler and Gruhn [9]. One important component of this method is a graphical notation to model processes that helps to identify activities in the process that would benefit if they where enacted with the support of mobile technologies. In this notation an organisation unit like an enterprise or a group of people is drawn as a grey box. Processes are depicted as white boxes which are connected by lines to show interactions. If such an interaction is an external one this is a hint that there might be potential for the employment of mobile technologies. A further hint for this is if a process cannot clearly be allocated to a single organizational unit.

6.3 Further related Concepts

There are a couple of papers describing data models for location-aware access control, see [7] for an overview. Almost all of these papers provide extensions for role-based access control, so it is possible to assign a location constraint to a role in a permission model for example. However, in these works no graphical notation is provided and these data models are not process aware.

Some authors describe implementations of workflow management systems (WfMS) that are especially designed to support actors working with mobile computers, e.g. [1]. However, these WfMS don't support the definition of static or even dynamic location constraints; also, in these papers no special graphical notation to model mobile-specific aspects of workflows can be found.

The idea of *binding of locations* and *separation of locations* was inspired by the well established security principles *binding of duties* and *separation of duties* and their application to workflow systems [18]. *Binding of duties* means that the actor who performed a particular activity is also obliged to perform certain other activites; e.g. if employee Alice in a company received a support inquiry via telephone and a workflow had to be initiated to find the solution for that inquiry Alice is also obliged to call the customer to inform him about the solution because the company's policy is to provide *one face to the customer*. The opposite case is *separation of duties*: this principles means that if a user performed a particular activity during a process he might not be allowed to perform particular subsequent activites in the process even if he has the general permission to perform these activities. An example for this would be a workflow system to support the review process of research papers submitted to a scientific conference that allows that even an actor that has the role *reviewer* is allowed to submit a paper; however, if this actually happens it should be

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002

enforced that a reviewer cannot review his own paper.

The idea of assigning different security labels to locations is described in [15]. However, the location model in this paper doesn't support different classes of locations. Further, there is no support for process-specific aspects.

Access control based on the user's location leads to the question how secure the employed locating system is with regard to manipulation attempts. An overview on different approches to prevent or detect this *location spoofing* is given in [8].

7. CONCLUSION & FUTURE WORK

In this article we motivated and introduced a novel approach to model specific aspects of business processes that make use of mobile technologies. An extension to UML activity diagrams was presented that enables the modeller to express constraints concerning the location where individual activites during a process have to be performed or are not allowed to be performed. Two general classes of location constraints were distinguished: *Static constraints* are assigned during administration time before the execution of the process. In contrast to this *dynamic constraint* are derived during the runtime of a process. One way to obtain these dynamic location constraints is to employ rules which define that the location where one or more source activities were performed has to be the location where target activities have also to be performed (binding of location) or are not allowed to be performed (separation of location).

It would be interesting to further enhance the model so that constraints concerning other relevant context parameters (e.g. type of mobile device, quality of wireless connection, time) can also be expressed. For example, it could be requested that particular activities are only performed within 9am and 18pm (because daylight is necessary) or that an activity can only be performed with a mobile computer that has currently a wireless communiation connection providing a bitrate of at least 100 KBit/sec.

We further envision a special graphical editor that support drawing of diagrams according to our UML profile. This editor should support working with geographical data for the definition of static location constraints. Further this tool should be capable of detection inconsistencies with regard to spatial restrictions assigend to activities. An example for such an inconsistency would be if a dynamic rule creates a negative location constraints for an activity that already has a positive static location that lies within the spatial extent of the negative constraint. This obviously constitutes a contradiction, so the modeller using the tool should be informed about it.

8. REFERENCES

[1] G. Alonso, R. Günthör, M. Kamath, D. Agrawal, A. E. Abbadi, and C. Mohan. Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients. *Distributed and Parallel Databases*, 4(3):229–247, 1996.

[2] H. Baumeister, N. Koch, P. Kosiuczenko, and M. Wirsing. Extending Activity Diagrams to Model Mobile Systems. In *Proceedings of NetObjectDays (NOD)*, pages 278–293, Erfurt, Germany, 2002.

[3] D. E. Bell and L. J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, The MITRE Corporation, Bredford, MA, USA, 1976.

[4] T. H. Davenport. *Process Innovation. Reengineering Work through Information Technology*. Ernst & Young, 1993.

[5] M. Decker. A Security Model for Mobile Processes. In *Proceedings of the International Conference on Mobile Business (ICMB 08)*, Barcelona, Spain, July 2008. IEEE.

[6] M. Decker. A Location-Aware Access Control Model for Mobile Workflow Systems. *International Journal of Information Technology and Web Engineering (IJITWE)*, 4(1):50–66, January-March 2009.

[7] M. Decker. Location-Aware Access Control: An Overview. In *Proceedings of Informatics 2009 — Special Session on Wireless Applications and Computing (WAC '09)*, pages 75–82, Carvoeiro, Portugal, 2009.

[8] M. Decker. Prevention of Location-Spoofing. A Survey on Different methods to Prevent the Manipulation of Locating-Technologies. In *Proceedings of the International Conference on e-Business (ICE-B)*, pages 109–114, Milano, Italy, 2009.

[9] A. Köhler and V. Gruhn. Analysis of mobile business processes for the design of mobile information systems. In *EC-WEB 2004*, pages 238–247, Zaragoza, Spain, 2004.

[10] P. Kosiuczenko. Sequence diagrams for mobility. In *ER/IFIP8.1 Workshop on Conceptual Modelling Approaches to Mobile Information Systems Development (MobIMod 2002)*, Lecture Notes in Computer Science, Volume 2784, pages 147–155. Springer, 2003.

[11] A. Küpper. *Location-based Services – Fundamentals and Operation*. John Wiley & Sons, Chichester, U.K., 2007. Reprint.

[12] R. Lake, D. S. Burggraf, M. Trninic, and L. Rae. *GML. Geography Mark-Up Language. Foundation for the Geo-Web*. John Wiley & Sons, Chichester, England, 2004.

[13] Object Management Group. *Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*, 2007.

[14] Object Management Group. *Unified Modeling Language (OMG UML), Superstructure, V2.1.2*, 2007.

[15] I. Ray and M. Kumar. Towards a Location-based Mandatory Access Control Model. *Computers & Security*, 25(1):36–44, 2006.

[16] V. Stefanov, B. List, and B. Korherr. Extending UML 2 Activity Diagrams with Business Intelligence Objects. In *Proceedings of Data Warehosuing and Knowledge Discovery (DaWaK 2005)*, pages 53–63, Copenhagen, Denmark, 2005.

[17] M. Stender and T. Ritz. Modeling of B2B mobile commerce processes. *International Journal of Production Economics*, 101(1):128–139, 2006.

[18] J. Wainer, P. Barthelmess, and A. Kumar. W-RABC – A Workflow Security Model Incorporating Controlled Overriding of Constraints. *International Journal of Cooperative Information Systems*, 12(4):455–485, 2003.

Digital Object Identifier: 10.4108/ICST.MOBIMEDIA2009.8002
http://dx.doi.org/10.4108/ICST.MOBIMEDIA2009.8002