A Mobility Restriction Authoring Tool Approach based on a Domain Specific Modeling Language and Model Transformation

Adalberto T. Azevedo Jr.[1], Fernando Benedito[1], Luciano Reis Coutinho[1], Francisco José da Silva e Silva[1], Marcos Paulino Roriz Junior[2] and Markus Endler[3]

[1]*Departament of Informatics, Federal University of Maranhao, Brazil*

[2]*Departament of Informatics, Federal University of Goias, Brazil*

[3]*Departament of Informatics, Pontifical Catholic University of Rio de Janeiro, Brazil*

Keywords: Mobility Management, Mobility Restrictions Monitoring, Authoring Tool, Domain-Specific Modeling Language, Model Transformation.

Abstract: There are many situations in which there is a need to monitor the location and behavior of people and/or vehicles in order to detect possible irregularities and control where they are located and how they move, such as in companies, public transportation and public security. In this paper, we present MobCons-AT (Mobility Constraints Authoring Tool), an authoring tool that allows the specification of mobility restrictions rules that must be followed by mobile nodes. Rules are specified through a Domain-Specific Modeling Language (DSML) called MobCons-SL (Mobility Constraints Specification Language). Once specified in MobCons-SL, these rules are automatically transformed into software artifacts that performs the detection of the mobility restrictions violations performed by mobile nodes. This approach allows faster delivery time and lower the cost for the development of software systems aiming the detection of mobility restrictions. This paper also describes the use of MobCons-AT in two case studies, showing its applicability for diverse mobility scenarios.

1 INTRODUCTION

In many situations it is necessary to monitor the location and behavior of people and/or objects in order to detect possible irregularities and to control where people, groups of people or vehicles are located and how they move (Zheng et al., 2014). For example, a mining company might be interested in restricting employee access to certain areas, depending on the mandatory use of safety equipment, prior training, or even the employee's function. Similarly, the company could restrict a vehicle's access and behavior in certain geographic area, limiting within the speed limits, proximity between vehicles or monitoring if they are staying on defined routes. In public transportation, it can be desired to verify if a bus following its route and a set of patterns, such as respecting a speed limit and maintaining a minimum distance between buses of the same line in order to maintain a regular frequency of them at bus stops. In the area of public safety, police vehicles could be monitored in order to control their area of coverage and circulation, working hours and location. Similarly, port companies may wish to restrict employee access to restricted areas for security reasons, as well as monitor ships movement. Finally, airports can control the movement of people and/or vehicles in areas of aircraft circulation.

In the literature there are several proposed software solutions related to the monitoring of mobile nodes and the vast majority of them follows a traditional software engineering approach (Antoniou et al., 2014; Al-Khedher, 2012; Behzad et al., 2014; Al-Taee et al., 2007; Joy et al., 2016; Punetha and Mehta, 2014; Almomani et al., 2011; Oliveira et al., 2013). Alternatively, the use of Authoring Tools is gaining more space in the software market mainly because it allows the reduction of software delivery time and its associated costs. Authoring Tools are systems that simplify the production of software content, that is, the users become authors without the need of developing a large amount of code lines.

The aim of this paper is to present an Authoring Tool called MobCons-AT that simplifies the complexity of developing applications for monitoring the mobility of nodes where mobility restrictions must be enforced. The proposed Authoring Tool is domain independent and is based on a domain specific language that allows end users to specify mobility restrictions and the context where they must be applied and uses model transformation techniques for the automatic generation of code. The development of MobCons-AT is part of the MobileAMP project.

Azevedo Jr., A., Benedito, F., Coutinho, L., Silva, F., Roriz Junior, M. and Endler, M.
A Mobility Restriction Authoring Tool Approach based on a Domain Specific Modeling Language and Model Transformation.
DOI: 10.5220/0007727905250534
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 525-534
ISBN: 978-989-758-372-8

The remainder of this paper is structured as follows: Section 2 discuss the fundamental concepts used in this work. Section 3.1 provides an overview of the MobileAMP project and its architecture. Section 3.2 presents the DSML created to specify mobility constraints, as well as its abstract syntax (metamodel) and its concrete syntax (a graphical notation). Section 3.3 describes the automatic generation of code, realized through the use of M2T (Model to Text) transformations[1]. Section 4 provides case studies aiming to show the effectiveness of the tool and of the developed language. Section 5, presents the related works, comparing them with the proposed approach. Finally, Section 6 describes the conclusions of the developed work an points for future efforts directions.

2 FUNDAMENTAL CONCEPTS

Researchers have long been dedicated to developing adaptive and intelligent authoring systems (Murray et al., 2003). Murray (Murray, 1999) has classified authoring tools into a number of categories, like Curriculum Sequencing and Planning, Tutoring Strategies, Domain Expert System, Multiple Knowledge Types and Special Purpose. The Special Purpose category is specialized in specific tasks or domains and can best support the author's needs for specific situations. In other words, this category of authoring tools may be based on specific domains and modeled by languages.

The problem of specifying mobility constraints can be treated by authoring tools. To do so, these problems domain are described by a model. A model is an abstraction of a system often used to replace the system under study (Ludewig, 2003). In the last decades, many techniques and modeling languages have been proposed to support the design and development of complex software systems. More recently, models have become more than documentation artifacts, assuming central roles in the software engineering process, in an approach known as Model-Driven Engineering (MDE). In addition to the benefits mentioned above, models also allow - through complex techniques such as metamodeling, model transformation, code generation or model interpretation - to create or automatically run software systems based on these models. For the Object Management Group (OMG)[2], "metamodels are model models". As for (Favre and NGuyen, 2005), "metamodels are model language models". Based on such concepts, we can define metamodel as a model that defines the structure of a Modeling Language (Da Silva, 2015).

The simplicity with which the user defines his specifications is best obtained through a Modeling Language, that is, a set of all possible models that are in accordance with the abstract syntax of that language, represented by one or more concrete syntax and that satisfy a given semantics. The definition of a modeling language usually begins by capturing and identifying the application main domain. This task represents the domain analysis phase to build the modeling language (Mernik et al., 2005). The result of this activity produces the abstract syntax of the modeling language, which corresponds to a metamodel with all the concepts identified at the metadomain level. The concrete syntax of a modeling language refers to its notation, that is, how users will learn and use it, whether by reading or writing and designing the models. Thus, the success of a modeling language will depend on the right balance between simplicity, expressiveness, ability to write, readability, learning, and effectiveness. Finally, a modeling language can be classified as a General-Purpose Modeling Language (GPML) or Domain-Specific Modeling Language (DSML) (Mernik et al., 2005; Fowler, 2010). A GPML has broader and widespread use in different fields of application. On the other hand, DSMLs use some constructs or concepts closer to their application domain, which is usually easier to read, understand, validate and communicate.

Once the entire specification is modeled, it can be transformed into artifacts through a standard Transformation process. Models can be created manually or generated automatically through a process of converting a source model to a destination model called Transformation between models. OMG's Model to Text (M2T) Transformation is one of the key transformations that generate or produce software artifacts, typically source code, XML (Extensible Markup Language), and other text files, from models. The most common technique for this class of transformations is known as code generation, and there are several solutions and techniques (Guinelli et al., 2014). It can be specified through different languages, such as conventional programming languages, as well as specialized model transformation languages, for different purposes and with different modeling paradigms such as Acceleo[3] and ATL (Atlas Transformation Language)[4]. Through transformation rules, concrete software artifacts of the models produced by the domain language can be generated.

[1]M2T Transformation: https://www.eclipse.org/modeling/m2t/

[2]OMG: http://www.omg.org/index.htm

[3]Acceleo: http://www.eclipse.org/acceleo/

[4]ATL: https://eclipse.org/atl/documentation/

3 THE MobCons-AT AUTHORING TOOL

MobCons-AT is an authoring tool for end-user programming that is part of the MobileAMP project and allows the specification of rules that define mobility restrictions that must be followed by mobile devices, with automatic code generation and near real time detection of the defined restrictions violations. Through the implementation of a DSML called MobCons-SL, a central part of the tool, MobCons-AT allows users to graphically and textually model scenarios involving this domain.

The section begins by giving an overview of the MobileAMP project followed by the description of the developed DSML and the transformation process for software artifact generation. Finally, an end-user prototype is presented.

3.1 The MobileAMP Project

The MobileAMP project focuses on the development of software that allows the real-time analysis of activity-mobility patterns from the processing of context data streams generated from people, vehicle and other moving entities (also known as mobile nodes). Unlike most previous work, which uses a statistical approach to store data and process it off-line, MobileAMP stands out by discovering such patterns in real time through the use of CEP (Complex Event Processing) data-stream processing primitives. The data-stream events in the current project stage provide information of update locations, mobile node type and timestamp. However, other context data such as speed and acceleration should be exploited in the future. The general system architecture proposed in the context of this project is composed of four main components, as shown in Figure 1.

- **Authoring Tool:** software module that implements a DSML, called MobCons-SL, that allows the user to specify mobility restrictions at a high level abstraction. These rules are transformed into artifacts that access the MobCons library classes;
- **Mobile Nodes:** entities running a mobile application that can acquire sensor data, such as location, velocity, timestamp, and send them through data stream for further analysis;
- **Cloud:** executes the CEP engine that receives the real-time events from the mobile nodes, processes the data stream and generates derived events based on rules that identify the mobility constraint violations;
- **Monitoring:** a visualization tool that provides a user interface that is responsible for receiving and displaying alerts of mobility restrictions violations performed by mobile nodes. Events describing mobility restrictions violations can be logged in a database along with the mobile nodes location updates for proving a play back feature allowing the visualization of past actions.

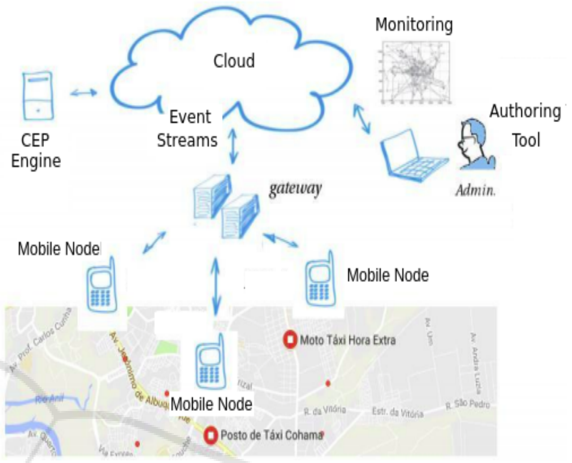

Figure 1: MobileAMP project: General Architecture.

The project software system provides a library, called MobCons, that provides an API for definition of mobility restrictions and the context where they must be applied and transform them into CEP rules that can detect restrictions violations. The library also dynamic instantiates the generated CEP rules in a CEP engine that process the data stream generated by mobile nodes and outputs derived events whenever mobility restrictions violations are detected. The output is consumed by the Monitoring Model. The MobCons-AT tool, main focus of this paper, provides a high level DSML used by the end-user to easily specify mobility restrictions and a set of transformation rules that transforms the DSML statements into MobCons code.

3.2 The MobCons-SL Language

To abstract the complexity of specifying mobile nodes mobility restrictions, MobCons-AT implements a DSML called MobCons-SL, which allows users to model the most diverse scenarios involving this subject. The DSML created is composed of an abstract syntax, represented by a metamodel, and by a concrete syntax in accordance with this metamodel, represented by a graphical notation.

3.2.1 Abstract Syntax - Metamodel

The proposed metamodel was developed using the EMF (Eclipse Modeling Framework)[5] standard. EMF is a common standard for data models that many technologies and frameworks are based on. It is able to produce artifacts for various languages and has a number of compatible tools. The developed metamodel is presented in Figure 2.

The `Specification` class (painted yellow in Figure 2) corresponds to the metamodel first level and it is a starting point for the specification of mobility restrictions. From the Specification class it is possible to define `Context`, `Restriction`, `TemporalCondition` and `Rule`.

`Context` (blue in Figure 2) define for which mobile nodes a constraint must be applied and can be of the following types, according to the metamodel:

- **MU:** Used to apply a restriction to a single mobile node;
- **Group:** Used to group two or more mobile nodes to which it can be assigned a given restriction;
- **MUType:** Defines a context for all mobile nodes of a specific type (CAR, BUS, PERSON, etc);
- **Area:** Defines a geographical area, which can be circular, a rectangle or a polygon. A mobility restriction can be applied to an area and should be obeyed by all mobile nodes when moving within its boundaries.

`Restriction` (red in Figure 2) represent limitations imposed on the mobility of mobile nodes. The proposed metamodel allows the specification of the following restrictions types:

- **Access:** Restricts the mobility of mobile nodes by detecting when they access a restricted area that must be avoided;
- **Permanence:** Restricts the mobility of mobile nodes by detecting when they leave the boundaries of a given area;
- **Speed:** Imposes a maximum and/or minimum speed limit to mobile nodes;
- **Path:** Defines a path to be followed by mobile nodes, detecting when they deviate from the provided path given a threshold;
- **MaxDistance:** Defines a maximum distance allowed between two `Contexts`;
- **MinDistance:** Defines a minimum distance allowed between two `Contexts`;

[5]EMF:http://projects.eclipse.org/projects/modeling.emf

- **Punctuality:** Defines the expected time in which a mobile node must achieve a particular geographical area.

The metamodel also provides `TemporalCondition`. These classes **(in gray color)** are optional and allow to specify when a mobility constraint must be applied. It allows specify a date, days of the week, and/or time bands for a given restriction. For example, it is possible to define that a bus must travel at a maximum speed of 60 km/h from Monday to Friday (between 8 am and 6 pm) but on Saturdays and Sundays it can speed up to 80 km/h.

`Rule` (green in Figure 2) are associations between `Context`, `Restriction` and `TemporalCondition`, allowing, for example, to define a scenario such as:

All vehicles and motorcycles (`MUType Context`) can only transit within the Federal University (`Area Context`) at a maximum speed of 30 km/h (`Speed Restriction`), on Mondays, Wednesdays and Fridays, from 6 am to 11 am (`TemporalCondition`).

3.2.2 Concrete Syntax - Graphical Notation

Based on the described metamodel we developed a graphical representation used by the Authoring Tool user to specify the mobility restrictions in a easy way. In order to derive a concrete model representation based on the abstract metamodel syntax it is necessary to map the elements of the abstract metamodel to concrete graphical elements that represent them. Table 1 shows the graphical notation of the MobCons-SL language, where it can be observed that all objects presented in the metamodel (abstract syntax) are represented by a different icon. It is up to users to design the model using the graphical notation based on their knowledge of the application domain. When designing the graphical notation, our objective was to achieve a balance between simplicity, expressiveness, ability to write, readability, learning and effectiveness.

3.3 Transformation

A fundamental aspect defined by MobCons-AT is the possibility of generating code automatically through the use of M2T (Model to Text) transformations using as input the user provided model developed using the graphical notation. The transformation rules defined in MobCons-AT receive as input the concrete model (generated according to the metamodel and composed of all specifications made by the user through the MobCons-SL language) and generate as

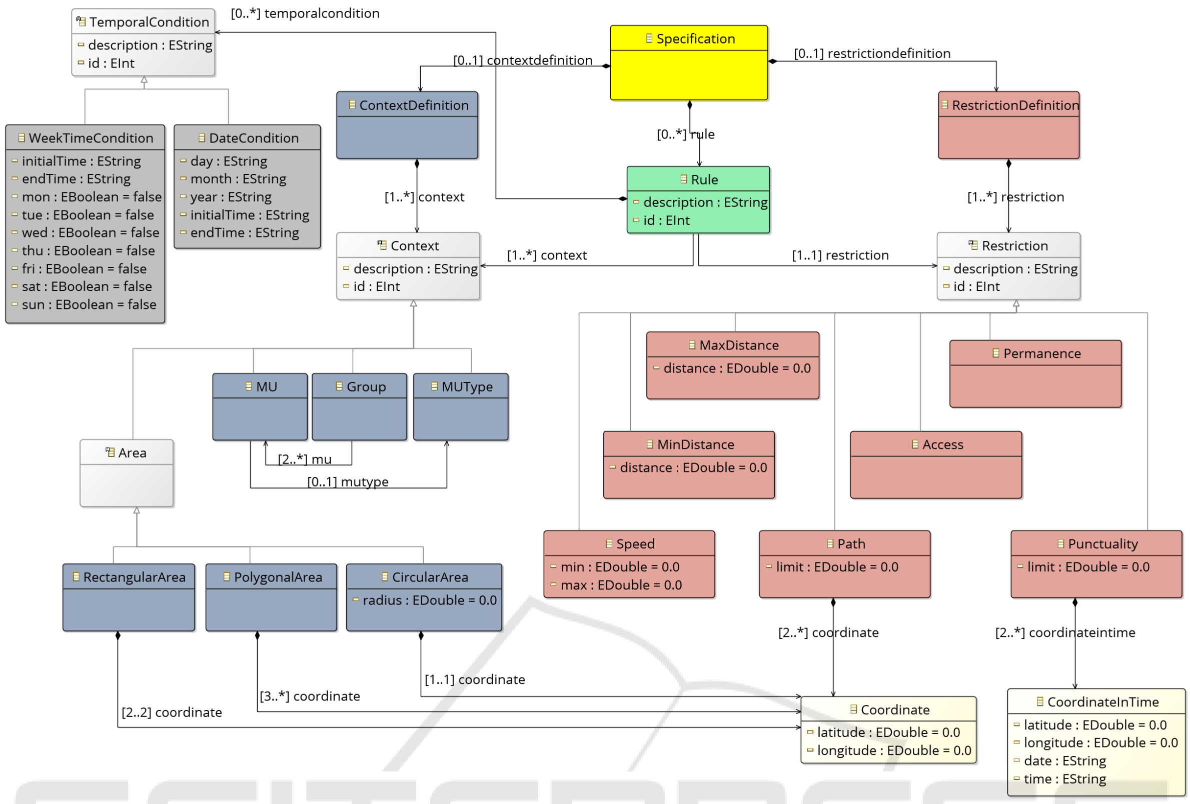

Figure 2: The MobCons-SL Metamodel.

Table 1: Graphical Notation.

	Object	Icon
Contexts	MU	🚶
	Group	👥
	MUType	🚗
	CircularArea	◎
	RectangularArea	▭
	PolygonalArea	⬠
Restrictions	Speed	◉
	Path	🗺
	Access	👤
	Permanence	↓
	MaxDistance	📍
	MinDistance	📍
	Punctuality	⋮
Temporal Condition	WeekTimeCondition	🕒
	DateCondition	📅
Rules	Rule	✗

output artifacts code in Java[6] that instantiates classes of the `MobCons` library responsible for the generation of CEP queries that are instantiated in the CEP engine running in the cloud.

The automated process of model transforming occurs through a set of defined rules. These rules are structured in the form of templates and queries, each template being responsible for a transformation stage and each query used to query values or collections of values defined in the template. The transformations used in MobCons-AT were implemented using the Acceleo tool, chosen for its easy use and its integration with the Eclipse IDE through a plug-in.

The transformation is started by the instantiating of the class `Specification`. From this class, all specified `Rule` classes are searched, and for each the transformation process looks for the `Restriction` classes, `Context` and possible `TemporalCondition` classes. In other words, the developed routine searches for all the specified rules, obtaining from each one, the constraint, contexts and temporal conditions (if they exist) associated.

To demonstrate the transformation process developed, consider the scenario showed in Section

[6]Java: https://www.oracle.com/java/index.html

3.2.1. To model this example, the MobCons-SL language generates the following objects: a `RestrictionDefinition` class and a `Speed` constraint, a `ContextDefinition` class, and a `PolygonalArea` context of at least three `Coordinate` objects. The association between the context and the constraint is done by creating a `Rule` class consisting of a `WeekTimeCondition`. The graphic notation that represents the concrete input model generated can be seen in Figure 3.

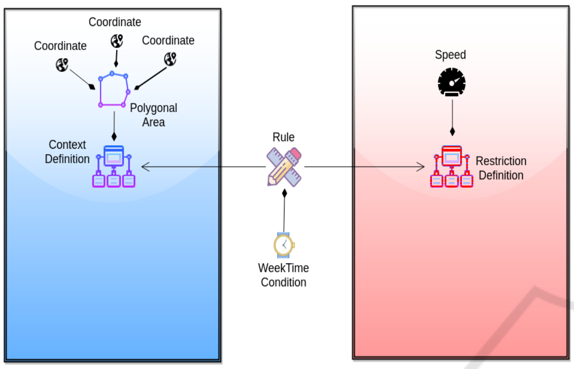

Figure 3: Concrete input model, based on the example of Section 3.2.1.

From this point, the objects defined in the concrete input model are transformed into the objects implemented in `MobCons`. The code of Listing 1 shows the artifact generated after the transformation of the concrete model, where it is possible to observe the mapping between the classes of the metamodel and the classes of the `MobCons` library. Lines 3 and 4 are responsible for the instantiation of the library main constructor followed by the calls to the constructors of the `SpeedRestriction` (line 6), `AreaContext` (line 7), and `ConstraintTimeClause` (line 9) classes. Line2 12 to 15 are responsible for binding the mobility restriction to the contexts and temporal conditions and generating the correspondent CEP rules by calling the **newMC()** method. This method is also responsible for dynamically instantiating the CEP rules into the CEP engine. Some simple mobility restrictions (such as speed limit) generate a single CEP rule, while more complex ones (such as route and punctuality) require several CEP rules for proper monitoring of mobile nodes.

3.4 The MobCons-AT Prototype

The MobCons-AT tool graphical interface was developed using the Sirius[7] tool, chosen for its easy use and its integration with Eclipse IDE through a plug-in.

[7]Sirius: https://www.eclipse.org/sirius/

```
1   public class MobCons {
2     public static void main(String[] args) {
3       MampCoreMobconsCore mobconsCore = new MobconsCore();
4       mobconsCore.init(true);
5
6       SpeedRestriction restriction11 = new SpeedRestriction(30.0);
7       AreaContext contextArea11 = new AreaContext(
8           new Point2D.Double(10.0, 30.0), 10.0);
9       ConstraintTimeClause weekTimeCondition11 =
10          ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
11          "06:00", "11:00",2,4,6);
12      mobconsCore.newMC(restriction11,
13                        contextArea11,
14                        weekTimeCondition11);
15    }
16  }
```

Listing 1: Artifact generated after transformation, based on the example of Section 3.2.1.

The simple to use graphical interface allow users to focus in the business domain and the provided transformation model avoids the necessity of writing code lines.

The proposed graphical interface allows to visually represent the concepts defined in the MobCons-SL language. Figure 4 shows the modeling of a mobility restriction scenario using the prototype.

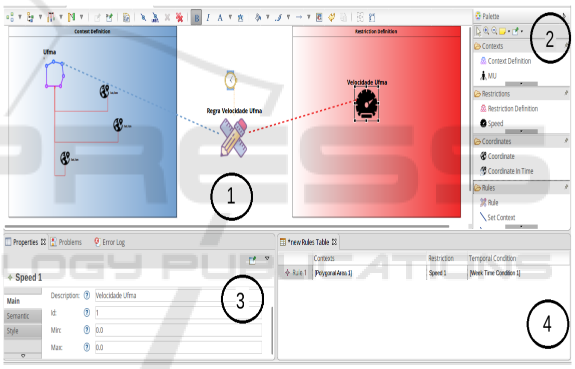

Figure 4: MobCobs-AT Prototype.

The tool window is divided basically into four parts:

1. An area for creating the models;
2. A palette of modeling elements, divided into
   - *Contexts*: elements that represent the contexts, that is, `ContextDefinition`, `MU`, `Group`, `MUType`, `CircularArea`, `RectangularArea` and `PolygonalArea`;
   - *Restrictions*: elements that represent the constraints, that is, `RestrictionDefinition`, `Speed`, `Path`, `MaxDistance`, `MinDistance`, `Access`, `Permanence` and `Punctuality`;
   - *Coordinates*: elements that represent the coordinates, that is, `Coordinate` and `CoordinateInTime`;
   - *Rules*: elements that represent the rules, their relationships with contexts and constraints, and

the types of temporal constraints, that is, `Rule`, `Set Restriction`, `Set Context`, `WeekTime Condition` and `Date Condition`.

3. Fields for textual edition of values to complement the graphical notation;
4. A rules visualization table, which shows in tabular format all contexts, constraints and temporal conditions that comprise the model being developed.

4 CASE STUDIES

This Section aims to demonstrate the expressiveness of the MobCons-SL language to describe mobility restrictions in diverse scenarios. It also demonstrate the functionality of MobCons-AT transformation rules to generate software artifacts that allows near real time monitoring of the defined mobility restrictions. We want to verify, therefore, that using the provided DMSL the user will be able to correctly model the scenarios and that the tool will correctly produce artifacts according to the model specified for each scenario. For this, two case studies were developed. The chosen case studies encompass `Rule` with a large number of `Context` and `Restriction`, as well as `TemporalCondition`.

4.1 Mining Company Scenario

A mining company wants to limit the maximum speed of vehicles within its iron ore yard to 40 km/h. Such speed restriction must be carried out between 07 am and 12 am, from Monday to Friday. The vehicles of the outsourced company Formula 1 should not exceed 60 km/h in all dependencies of the company. All vehicles must maintain a minimum distance of 60 meters from each other, for safety purposes. In addition, excavator-type vehicles should remain restricted to the iron ore yard and their displacement should follow a predefined route.

The Figure 5 shows a possible modeling to specify the mobility restrictions of the proposed scenario. In the Appendix Section, it is shown the artifact generated after the transformation process.

4.2 Public Safety Scenario

The Department of Public Security wants to control the use of its police vehicles. Following the rules of the Secretariat, each car must cover a predetermined area within a specific time period, which must be from 06 am to 18 pm, Monday to Friday, and from 10 am

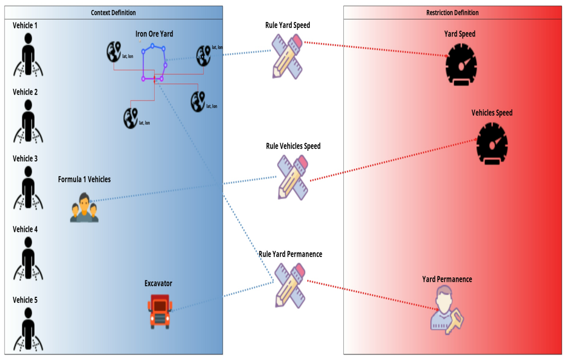

Figure 5: Mining Company Scenario.

to 22 pm on Saturdays and Sundays. The areas of operation of the vehicles should be: Itaqui-Bacanga neighbourhood, Downtown and Beaches. Except for times when they are called for emergencies or flagrant offenses, vehicles should only make rounds in their area of operation, at a maximum speed of 40 km/h.

The Figure 6 shows a possible modeling to specify the mobility restrictions of the proposed scenario. In the Appendix Section, it is shown the artifact generated after the transformation process.

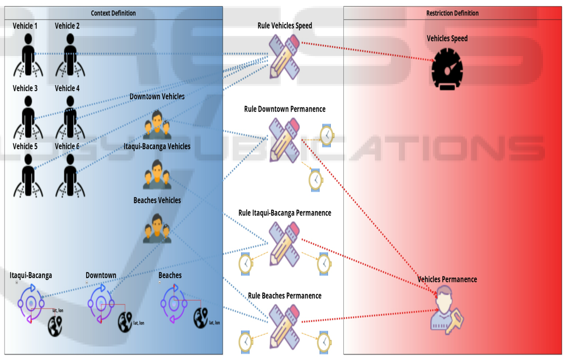

Figure 6: Public Safety Scenario.

4.3 Discussion

This Section illustrates the expressiveness of the Mobcons-SL language to describe mobility restrictions of mobile nodes in two case studies: a mining company and public safety. The use of the Mobcons-AT allowed the user to model the required mobility restrictions using a high level language based on a graphical notation. It has been shown the language flexibility for expressing diverse mobility restrictions and conditions, including temporal ones, that can be easily combined with a flexible set of contexts, ranging from individual mobile nodes, groups of mobile nodes, mobile nodes types and geographical areas.

The use of transformation rules allowed the automatic generation of non trivial software artifacts that use Complex Event Processing rules for near real time detection of the defined mobility restrictions violations, avoiding the time and effort necessary for writing that code.

As showed in this two case studies, the use of an Authoring Tool, such as Mobcons-AT, allows the user to focus in the domain expertise instead of programming software skills and contributes to the reduction of software delivery time and its associated costs.

5 RELATED WORK

To the best of our knowledge, there is no report in the literature of an authoring tool that implement a Domain-Specific Modeling Language to specify mobility restrictions to be applied to mobile nodes. In this section, we present related works that allow monitoring of mobile nodes, but are not based in the development of a DSML and the exploitation of model transformations.

There are some papers that investigate how to acquire the geographic position of a vehicle and send it to a central node that displays their location (Antoniou et al., 2014; Al-Khedher, 2012; Behzad et al., 2014). Others can also detect and alert when a monitored mobile node exceeds a certain speed limit (Al-Taee et al., 2007; Joy et al., 2016), when they leave a certain area (Punetha and Mehta, 2014), or can combine this two types of mobility restrictions (Almomani et al., 2011). The work described in (Oliveira et al., 2013) emits alerts when a defined route deviation occurs.

Most of the works are focused on vehicle tracking with the exception of (Punetha and Mehta, 2014), that was developed for tracking people, and (Oliveira et al., 2013), that was developed for tracking cargo. We should also note that most of the work associate mobility restrictions (speed limits, bounded area or route) to the monitored mobile nodes. (Joy et al., 2016), in the other hand, associates a speed limit to a specific area (or road).

As can be seen, previous works adopt a strategy based on traditional software development and propose a specific software already deployed that, of course, could be extended with a specialized team of programmers. The proposed approach described in this paper, on the other hand, provides an Authoring Tool that implements a DSML which allows the specification of a flexible set of mobility restrictions, temporal conditions and the context where they must be applied. Through the use of model transformation, software artifacts are automatic generated allowing near real time detection of mobility restrictions violations. Beyond that novel approach, the provided DSML allows the specification of a set of mobility restrictions richer than previous works that can be flexibly combined in different contexts.

6 CONCLUSION AND FUTURE WORK

The technological advances in the area of mobile and ubiquitous computing opens up opportunities for the gathering, distribution, and analysis of diverse sensor data such as location, speed, acceleration, temperature, and so on. This data is crucial for the understanding of the dynamics of individuals, enterprises, and even whole cities. In this paper we focused in mobility management and, more specifically, in the definition and monitoring of mobility restrictions that must be followed in given contexts. There are many applications in both public and private sectors that require such monitoring capabilities.

Although there are several work developed in this area, they follow the traditional approach of software engineering, deriving a specific software system that, to be altered or extended, requires a specialized team of software developers.

In this paper we propose a novel approach for developing such systems by providing an Authoring Tool, called Mobcons-AT, that implements a DSML (Mobcons-SL) which allows the specification of a flexible set of mobility restrictions, temporal conditions and the context where they must be applied. Through the use of model transformation, software artifacts are generated automatically, allowing near real-time detection of mobility constraint violations. The automatic software generation reduce the cost, time and human resources (software developers) required for building systems focusing the online monitoring of mobility restrictions.

The use of Mobcons-AT was validated through case studies based in two real-world scenarios. The scenarios demonstrate the language flexibility for expressing diverse mobility restrictions and conditions, including temporal ones, that can be easily combined with a flexible set of contexts, ranging from individual mobile nodes, groups of mobile nodes, mobile nodes types and geographical areas. In addition, a graphical notation allows users to easily describe the mobility restrictions without requiring much technical skills. The use of Mobcons-AT transformation rules allows the automatic generation of non trivial software artifacts that use Complex Event Processing rules for near real time detection of the defined mobility restrictions violations, avoiding the time and effort necessary for writing that code. In this way, the proposed solution, contributes to the reduction of cost and time required for the development of mobility restriction management systems.

Despite the various objectives achieved during the development of this work, there are some possibilities for improvements that have been observed. Future works include the development of experimental evaluation for analyzing the automatic generated system scalability in respect to the number of monitored mobile nodes and the modeling of complex mobility patterns, such as concentration, dispersion, congestion, leadership, and meeting.

ACKNOWLEDGEMENTS

This research is part of the INCT of the Future Internet for Smart Cities and was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brazil (CNPq) proc. 465446/2014-0, Fundação de Amparo à Pesquisa do Estado de São Paulo - Brazil (FAPESP) proc. 14/50937-1 and proc. 15/24485-9, Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão - Brazil (FAPEMA) proc. UNIVERSAL-00987/17.

REFERENCES

Al-Khedher, M. A. (2012). Hybrid gps-gsm localization of automobile tracking system. *arXiv preprint arXiv:1201.2630*.

Al-Taee, M. A., Khader, O. B., and Al-Saber, N. A. (2007). Remote monitoring of vehicle diagnostics and location using a smart box with global positioning system and general packet radio service. In *2007 IEEE/ACS International Conference on Computer Systems and Applications*, pages 385–388.

Almomani, I. M., Alkhalil, N. Y., Ahmad, E. M., and Jodeh, R. M. (2011). Ubiquitous gps vehicle tracking and management system. In *2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6.

Antoniou, A., Georgiou, A., Kolios, P., Panayiotou, C., and Ellinas, G. (2014). An event-based bus monitoring system. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2882–2887.

Behzad, M., Sana, A., Khan, M., Walayat, Z., Qasim, U., Khan, Z., and Javaid, N. (2014). Design and development of a low cost ubiquitous tracking system. *Procedia Computer Science*, 34(Supplement C):220 – 227.

Da Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155.

Favre, J.-M. and NGuyen, T. (2005). Towards a megamodel to model software evolution through transformations. *Electronic Notes in Theoretical Computer Science*, 127(3):59–74.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

Guinelli, J. V., de Souza Rosa, A., Pantoja, C. E., Choren, R., Friburgo-RJ-Brasil, N., and de Janeiro-RJ-Brasil, R. (2014). Uma metodologia para apoio ao projeto de banco de dados geográficos utilizando a mda. *X Simpósio Brasileiro de Sistemas de Informação*.

Joy, S. P., Sunitha, V. S., Devi, V. R. S., Sneha, A., Deepak, S., and Raju, A. J. (2016). A novel security enabled speed monitoring system for two wheelers using wireless technology. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–7.

Ludewig, J. (2003). Models in software engineering–an introduction. *Software and Systems Modeling*, 2(1):5–14.

Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10:98–129.

Murray, T., Blessing, S., and Ainsworth, S. (2003). *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software*. Springer Science & Business Media.

Oliveira, R. R., Noguez, F. C., Costa, C. A., Barbosa, J. L., and Prado, M. P. (2013). Swtrack: An intelligent model for cargo tracking based on off-the-shelf mobile devices. *Expert Systems with Applications*, 40(6):2023 – 2031.

Punetha, D. and Mehta, V. (2014). Protection of the child/ elderly/ disabled/ pet by smart and intelligent gsm and gps based automatic tracking and alert system. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2349–2354.

Zheng, Y., Capra, L., Wolfson, O., and Yang, H. (2014). Urban Computing: Concepts, Methodologies, and Applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):38:1—-38:55.

APPENDIX

```java
public class Mobcons {
  public static void main(String[] args) {
    MobconsCore mobconsCore = new MobconsCore();
    mobconsCore.init(true);
    SpeedRestriction restriction11 = new SpeedRestriction(40.0);
    AreaContext contextPolygonalArea11 = new AreaContext(
          new Point2D.Double(10.0,10.0),
          new Point2D.Double(20.0,20.0),
          new Point2D.Double(30.0,30.0),
          new Point2D.Double(40.0,40.0));
    ConstraintTimeClause weekTimeCondition11 =
          ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
          "07:00:00", "12:00:00",2,3,4,5,6);
    mobconsCore.newMC(restriction11,
                contextPolygonalArea11,
                weekTimeCondition11);
    SpeedRestriction restriction22 = new SpeedRestriction(60.0);
    GroupUMContext contextGroup12 = new GroupUMContext(1,2,3,4,5);
    mobconsCore.newMC(restriction22,contextGroup12);
    AreaRestriction restriction13 = new AreaRestriction(true,
          new Point2D.Double(10.0,10.0),
          new Point2D.Double(20.0,20.0),
          new Point2D.Double(30.0,30.0),
          new Point2D.Double(40.0,40.0));
    KindOfMUContext contextKindOfMU13 =
        new KindOfMUContext("Escavadeiras");
    mobconsCore.newMC(restriction13,contextKindOfMU13);
  } //end of the method main
} //end of the class
```

Listing 2: Artifact generated by the scenario 4.1.

```java
public class Mobcons {
  public static void main(String[] args) {
    MobconsCore mobconsCore = new MobconsCore();
    mobconsCore.init(true);
    SpeedRestriction restriction11 = new SpeedRestriction(40.0);
    SingleUMContext contextMU11 = new SingleUMContext(1);
    mobconsCore.newMC(restriction11,contextMU11);
    SingleUMContext contextMU21 = new SingleUMContext(2);
    mobconsCore.newMC(restriction11,contextMU21);
    SingleUMContext contextMU31 = new SingleUMContext(3);
    mobconsCore.newMC(restriction11,contextMU31);
    SingleUMContext contextMU41 = new SingleUMContext(4);
    mobconsCore.newMC(restriction11,contextMU41);
    SingleUMContext contextMU51 = new SingleUMContext(5);
    mobconsCore.newMC(restriction11,contextMU51);
    SingleUMContext contextMU61 = new SingleUMContext(6);
    mobconsCore.newMC(restriction11,contextMU61);
    AreaRestriction restriction12 = new AreaRestriction(true,new
        Point2D.Double(400.0,400.0),10.0);
    GroupUMContext contextGroup12 = new GroupUMContext(1,2);
    ConstraintTimeClause weekTimeCondition12 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "06:00:00","18:00:00",2,3,4,5,6);
    mobconsCore.newMC(restriction12,contextGroup12,weekTimeCondition12);
    ConstraintTimeClause weekTimeCondition22 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "10:00:00","22:00:00",1,7);
    mobconsCore.newMC(restriction12,contextGroup12,weekTimeCondition22);
    AreaRestriction restriction23 = new AreaRestriction(true,new
        Point2D.Double(10.0,10.0),5.0);
    GroupUMContext contextGroup23 = new GroupUMContext(3,4);
    ConstraintTimeClause weekTimeCondition33 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "06:00:00","18:00:00",2,3,4,5,6);
    mobconsCore.newMC(restriction13,contextGroup23,weekTimeCondition33);
    ConstraintTimeClause weekTimeCondition43 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "10:00:00","22:00:00",1,7);
    mobconsCore.newMC(restriction13,contextGroup23,weekTimeCondition43);
    AreaRestriction restriction34 = new AreaRestriction(true,new
        Point2D.Double(1000.0,1000.0),20.0);
    GroupUMContext contextGroup34 = new GroupUMContext(6,5);
    ConstraintTimeClause weekTimeCondition64 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "06:00:00","18:00:00",2,3,4,5,6);
    mobconsCore.newMC(restriction14,contextGroup34,weekTimeCondition64);
    ConstraintTimeClause weekTimeCondition74 =
        ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
        "10:00:00","22:00:00",1,7);
    mobconsCore.newMC(restriction14,contextGroup34,weekTimeCondition74);
  } //end of the method main
} //end of the class
```

Listing 3: Artifact generated by the scenario 4.2.