An Application of UML for Road Traffic Management System by Implementing Extensive Mechanism: Stereotypes

Monika Singh,
Faculty of Engineering (FET),
Mody University of Science and Technology

Ruhi Saxena
Faculty of Engineering (FET),
Mody University of Science and Technology

***Abstract*- This paper describe the extension mechanisms of the UML in real time system and it introduces a new approach of using Stereotypes and shows its utility by implementing in road traffic management system. This paper also discusses the process of specifying, translating and verifying UML specifications for road traffic management system. Different kinds of existing UML Meta-models used in road traffic management system are analyzed UML traffic system by implementing the Stereotypes will be created based on the real time system using the analysis report.**

***Keywords- UML, Extensible mechanism, safety critical system, road management system*.**

I. INTRODUCTION

The Unified modelling language (UML) [1] [3] has become a de-facto standard notation for analysis and design models of object oriented software system. It has been observed that graphical representation of model is easily accessible and understandable to the user. The primary gap between the developer and the user has been easily fulfilled by the graphical description.
Some of the benefits of UML are listed below:

- UML helps visualise, and document models of systems or processes, including their structure and design, in a way that meets the requirements specifications.
- Helps stakeholders understand what the system will be and what are the possible options available.
- It is language and platform independent.
- UML assembles the important aspects of a system while omitting the rest - abstraction mechanism - mapping of elements onto a Model.
- UML allows developers to quickly assemble programs from existing components and operations.
- It defines a wide set of concepts and diagrams to communicate information effectively. These are applicable to most domains

In its current form UML is comprised of two major components: a Meta-model and a notation
The Meta-model
UML is unique in that it has a standard data representation. This representation is called the metamodel. The meta-model is a description of UML *in* UML. It describes the objects, attributes, and relationships necessary to represent the concepts of UML within a software application. This provides CASE manufacturers with a standard and unambiguous way to represent UML models [2].
The Notation
The UML notation is rich and full bodied. It is comprised of two major subdivisions. There is a notation for modeling the static elements of a design such as classes, attributes, and relationships. There is also a notation for modeling the dynamic elements of a design such as objects, messages, and finite state machines
Table 1: Use of UML diagram for different level of System properties

Display the Boundary of a system	Illustrate the boundary of a system	Represent the static structure of a system	Model the behaviour of a system	Reveal the physical implementation architecture	Extend your functionality
Use case Diagram	Collaborations Diagram Sequence Diagram	Class Diagram	State Transition Diagram	Component Diagrams Deployment Diagram	Stereotypes Packages

II. WHY WE NEED EXTENSIBLE MECHANISMS

The graphical modeling elements and relationships defined for UML diagrams are sometimes too limited for certain modeling tasks [6]. The following are the problems with the UML diagrams:

- UML brings a set of notations and concepts that meet the needs of typical software modeling projects but some users

have found UML unable to express their modeling needs. (non software systems)
- Flexibility should be added to construct and document more heterogeneous and complex systems.
- UML lacks features that would allow attaching non-semantic information to models.
- Component models and architectural frameworks (JavaBeans, CORBA Component Model and COM+ cannot be modelled easily with UML.

Therefore to overcome above mentioned problems we need the extensible mechanisms.

III UML EXTENSIBILITY MECHANISMS

UML is an extensible language, in the sense it provides mechanisms (stereotypes, tagged values, and constraints) that allow introducing new elements for specific domains if it is needed, such as web applications, database applications, business modeling, software development processes, etc.
Due to limitations of UML diagrams, it is then desirable that the precision of the UML models can be increased, to provide information for automated analysis or to specify the intent of a diagram more precisely.
UML extension mechanisms are used to extend UML by:
- adding new model elements,
- creating new properties,
- and specifying new semantics

Moreover, it is often preferred that the UML can be extended to create new domain-specific modeling notations [4]. Currently, the two extension mechanisms that exist for UML 2.1 are profiling, also called a lightweight extension mechanism, and a heavyweight mechanism, as defined by the specification of the Meta Object Facility (MOF)[5][4].

A. LIGHTWEIGHT EXTENSION

The lightweight extension mechanism uses profiles to extend the UML. It consists of three main constructs:
- stereotypes
- tagged values
- constraints

It is called a lightweight extension mechanism, because it provides pure additions to the UML and does not change anything to the semantics of the metamodel elements, nor changes its structure nor adds new elements [5]. Whether or not to choose a lightweight metamodel extension technique depends on a number of factors. According to Desfray [6], a profile based technique should be chosen when:
• The domain is not subject to consensus, many variations and points of view exist.
• Many changes and evolutions may occur.
• The domain may be combined with other domains in an unpredictable way.
• Models defined in the domain may be interchanged with other domains.

Stereotypes
Among all, the stereotypes are most important. A stereotype is a model element that denotes additional values (based on tagged values), additional constraints, and optionally a new graphical representation (an icon). A stereotype allows us to attach a new semantic meaning to a model element. A stereotype is represented as a string between a pair of guillemots (<<>>), but it can also be rendered by a new icon.

Tagged Values
Just like a class has attributes, a stereotype may have properties, which are referred to as tag definitions. Once the stereotype is applied to a model element, the values of these properties are referred to as tagged values.
Tagged values are only supposed to extend model elements as attributes of stereotypes [7]. Tagged values can be useful for adding properties about
- code generation
- version control
- configuration management
- authorship
- etc.

Constraints
Besides tagged values, a stereotype may contain constraints with which the extended model can be restricted semantically. They are specified between braces {and} and can be expressed in any kind of language (e.g. English, OCL). A constraint is an assertion and is therefore not executable. An example of a predefined constraint is the {required} constraint on the extension relationship for stereotypes.

B. HEAVYWEIGHT EXTENSIONS

Whereas lightweight extensions can only provide pure additions to the UML, heavyweight extensions can also change the semantics of the UML. This is done by explicitly adding new metaclasses and other metaconstructors which can introduce new behaviour [30]. This is in contrast with the lightweight stereotypes, which can only extend existing metaclasses and do nothing by themselves. A heavyweight extension technique should be applied when [4]:
• The domain is well defined and has a unique well accepted set of main concepts.
• A model realized under the domain is not subject to be transferred into other domains.
• There is no need to combine the domain with other domains.

Because of the fact that a heavyweight extension not just extends the language, but is also capable of changing it, it can be discussed whether the designation 'extension' is appropriate here. A heavyweight 'extension' can even change the complete syntax and semantics of a language and thus essentially defines a new language. It is for this reason that it is very hard to combine multiple heavyweight extensions in a single model. Different languages do not combine very well. Furthermore, it is very hard for a tool to provide support for heavyweight extensions.

IV. SAFETY CRITICAL REAL TIME SYSTEM DOMAIN- ROAD TRAFFIC MANAGEMENT SYSTEM (RTMS)

Road transportation is an important economic force that faces many problems. Traffic congestion, environmental pollution and safety are becoming increasingly unaccepted by society. The introduction of new infrastructure is important but not sufficient. Traffic demand is increasing, while constructing new road infrastructure is limited due to environmental, social and financial constraints. In order to cope with these challenges, a possible solution is to manage and to control road traffic by developing Road Traffic Management Systems (RTMS).

RTMS make use of real-time data acquired from the road network in order to reduce traffic congestion and accidents, and to save energy and preserve the environment [9].

The Road Traffic management System (RTMS) has three active actors. Basically the Admin who uses the dataset and give the complaints, important suggestions which are under taken by the traffic police. Traffic police maintains the information which is provided by the users (Admin, vehicle owners).

Based on the above scenario, the UML diagram is implemented in figure 1, figure 2, and figure3.

Use case diagram (figure 1, figure 2) captures the functional requirement of the system and it's the interaction between the actor the system.

Consider Figure 1 and Figure 2 which describes the Use case diagram with basic function of RTMS using the stereotypes:

Use case description:

1. Vehicle Owner:
   - Login
   - Complaints
   - Licence received

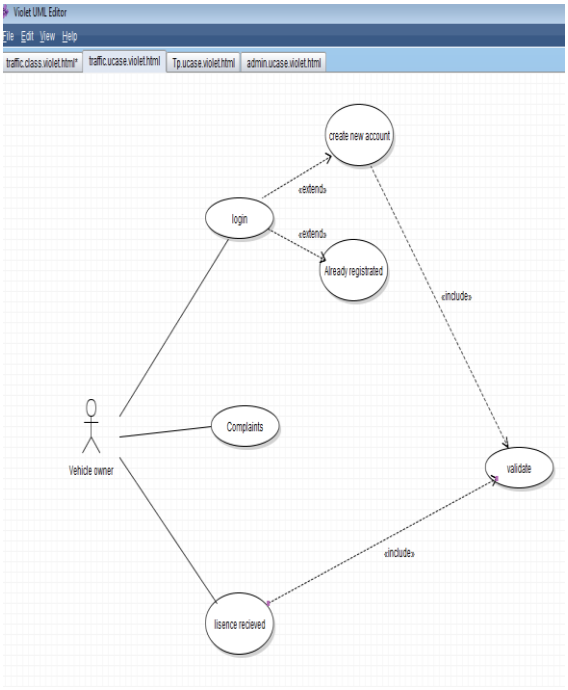

Fig. 1: Use case Diagram of Vehicle owner

2. Traffic Police:
   - Record Traffic Signals
   - Control entire traffic
   - Check and handle complaints

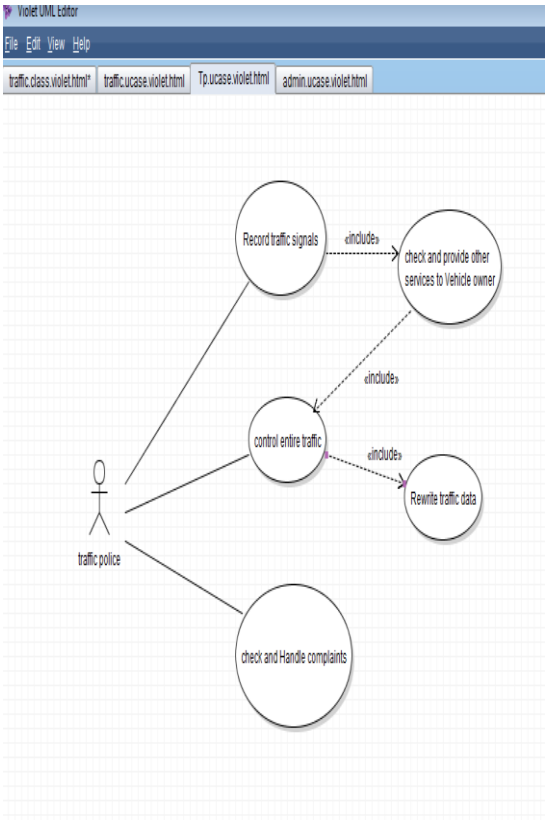

Fig. 2: Use case Diagram of Traffic Police

Class diagram describes the static designs which in turn help us to understand the functional requirements of the system, how the system is composed from the description of classes. A class diagram shows a set of classes, interfaces collaborations and their relationships.

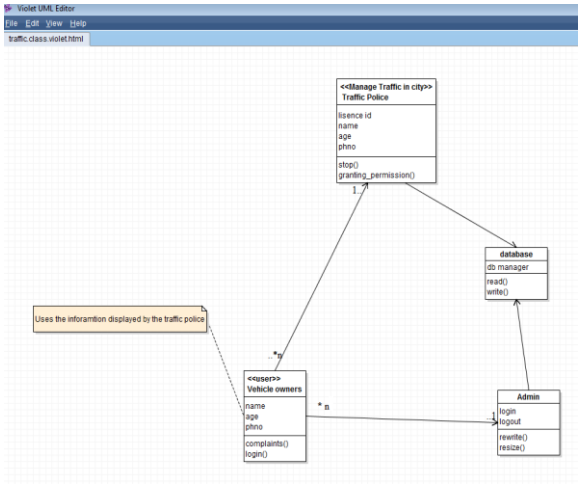

Fig. 3: Class Diagram of RTMS

V. IMPORTANCE OF EXTENSIBLE MECHANISM: SETEROTYPE IN SAFETY CRITICAL APPLICATION

According to *Kopetz,* a real time safety critical system is a computer system in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instant at which these results are produced [8].

Stereotypes are one of the extensibility mechanisms in UML. Example <<include>> <<extend>> distinguish dependencies in use case diagrams. A stereotype may be associated with a Class or Association (or other model element) in the definition of a profile.

The main aspects in which a stereotype can modify the metaclass definition are:

- Presentation such as, first character of stereotype in lower case in front of elements name enclosed in guillmets; icon or display.
- Additional properties for classes and attributes. The most important properties for an attribute (in addition to the name) are Type, Default Value, Multiplicity, and Unique.
- Additional constraints: In a profile only stereotypes can own constraints. The constraint can be named. In the CASE tool Rational Rose the constraints can be written in Java or OCL syntax.
- Stereotypes can specialize or generalize other stereotypes.

A common set of stereotypes for real-time system design includes the following methods [10]:

- *Subsystem:* A subsystem is an abstraction of a complex component.
- Passive Class: passive classes can only change their state when they are requested by other objects, by means of a method or operation invocation.
- *Protected Class:* protected classes can only Change their internal state through atomic operations. They represent data or other objects which are used by more than one active object
- *Cyclic Classes:* Cyclic objects are used to represent periodic behaviour. They have an independent thread of control,
- *Resource Classes:* Data resources can be represented with protected classes, but there are other resources, like CPU, input output devices, or memory, that impose constraints on the system execution.

VI. CONCLUSION

In this paper, a detailed UML documentation for road traffic management system using stereotypes is given. The UML diagram used in the documentation are Use Case and Class diagrams. The UML extension mechanisms provide not only a means for communication but also a framework for the knowledge and experiences of the individuals within a development environment.

Stereotype describes the behaviour/state of individual objects. Use case diagram of different user (Traffic Police, Vehicle Owner, and Admin) and Class diagram of road traffic management system are explained in this paper by using the stereotypes. However, functions of the road traffic management system described in this paper are limited to basic functionality.

REFERENCES

[1]. Grady Booch, James Rumbaugh, Ivar Jacobson, “The Unified Modeling Language User Guide”, (1999).

[2]. Rumbaugh, I. Jacobson and G. Booch: *The Unified Modeling Language Reference Manual, Second Edition*, Addison Wesley, 2006.

[3]. B.Selic and J.Rumbaugh, “*UML for modeling complex real-time systems”,* Technical report, Object Time, 1998.

[4]. Jorge Enrique Pérez-Martínez, “Heavyweight extensions to the uml metamodel to describe the c3 architectural style”, Software Eng. Notes, 2003.

[5]. David S. Rosenblum, “Lightweight Extension Mechanisms for UML”, Lecture notes Advanced Analysis and Design (GS02/4022), 2005.

[6]. Phillippe Desfray, “UML Profiles versus Metamodel extensions: An ongoing debate”, 2000.

[7]. OMG, “Unified Modeling Language: Infrastructure version 2.0 formal/05-07-05”, Object Management Group, March 2006.

[8]. Hermann Kopetz, “Real-Time Systems, Design Principles for Distributed Embedded Applications.UML

2003-theunified modeling language: modeling languages and applications", 6th international conference, San Francisco, CA, USA, October 20-24, 2003: proceedings.
[9]. B.Selic and J.Rumbaugh: *UML for modeling complex real-time systems,* Technical report, Object Time, 1998.

[10]. Abdelouahed Gherbi and Ferhat Khendek, "*UML Profiles for Real-Time Systems and their Application*", in Journal of Object Technology, vol.5, 2006.