4th INTERNATIONAL CONFERENCE
TRANSPORT SYSTEMS TELEMATICS TST'04

ZESZYTY NAUKOWE
POLITECHNIKI ŚLĄSKIEJ 2004
TRANSPORT z.55, nr kol. 1657

telematic, intelligent transport systems (ITS), architecture, functionality, electronic toll collection, object oriented, Unified Modeling Language, diagram, statechart

Aleš JANOTA[1]
Jiří ZAHRADNÍK[2]
Karol RÁSTOČNÝ[3]

ELECTRONIC TOLL COLLECTION – A PART OF THE SLOVAK ITS ARCHITECTURE

The paper deals with creating object oriented model of the Intelligent Transport System (ITS) with a special attention paid to systems of electronic toll collection (ETC), which are discussed here in more details as a part of created logic (functional and information) architecture. Even though it results from European FRAME architecture and some other national architectures, unlike them it is based on object-oriented approach. The authors show how the Unified Modeling Language can be used to create a proper model representing functional specification of the future telematic system.

ELEKTRONICZNE POBIERANIE OPŁAT DROGOWYCH – CZĘŚĆ SŁOWACKIEJ ARCHITEKTURY ITS

Referat traktuje o tworzeniu modelu zorientowanego obiektowo Inteligentnego Systemu Transportu (ITS) ze szczególnym uwzględnieniem systemów elektronicznego pobierania opłat (ETC), które są tu omówione bardziej szczegółowo jako część utworzonej (funkcjonalnej i informacyjnej) architektury logicznej. Wprawdzie wynika ona z europejskiej architektury FRAME i niektórych innych krajowych architektur, ale w przeciwieństwie do nich opiera się na podejściu obiektowym. Autorzy pokazują, jak Zunifikowany Język Modelowania może być wykorzystywany do tworzenia właściwego modelu przedstawiającego specyfikację funkcjonalną przyszłego systemu telematyki.

1. INTRODUCTION

Integration into the EU became a historical milestone on the way of the Slovak Republic towards building democracy and prosperity. This long-time process affects many areas of economic and social life in the country. One of them is efficient and safe transportation of people and goods. Implementation of ITSs is currently at various stages of

[1] Department of Control & Information Systems, Faculty of Electrical Engineering, University of Žilina, Veľký diel, 010 26 Žilina, Slovakia, ales.janota@fel.utc.sk
[2] Department of Control & Information Systems, Faculty of Electrical Engineering, University of Žilina, Veľký diel, 010 26 Žilina, Slovakia, jiri.zahradnik@fel.utc.sk
[3] Department of Control & Information Systems, Faculty of Electrical Engineering, University of Žilina, Veľký diel, 010 26 Žilina, Slovakia, karol.rastocny@fel.utc.sk

development in all member states. As any of new member countries, Slovakia has own specific problems related to transport that have need of urgent solution. The priority 1 is completion of transport infrastructure (e.g. see highway system Fig.1) requiring a lot of financial resources. Speaking of telematic applications in the light of recently adopted national transport policy there are several functional areas that should be preferably developed. They mostly cover deployment of traffic control systems, emergency control systems, safety systems, electronic toll collection systems, traffic information systems, advanced driver assistance systems, public transport systems and freight and fleet management systems. As in many other countries, full integration of future telematic applications can hardly be realised without creating proper system architecture. The following text deals with a possible approach to creating such architecture that provides the basis for working and workable telematic systems.

Fig.1. Slovak road network with existing highways indicated [8]

2. ITS SYSTEM ARCHITECTURE

Telematic systems use information, transport and communications technologies, in vehicles and/or within the infrastructure, to improve mobility while increasing transport safety, reducing traffic congestion, maximising comfort and reducing environmental impacts. The structure of the entire system that illustrates component elements (technologies and specific systems) and their relation to each other is called "system architecture". It reflects several different views of the system and use to be divided into different parts, e.g. functional architecture (processes within the ITS), information architecture (structure and hierarchy), physical architecture (allocation of physical devices), communication architecture (transmission environment among physical devices) and/or organisation architecture

(competencies of management levels). The main subject to discuss later is architecture describing functionalities of ITSs.

2.1. RESOURCES

Generally, there are two different principal approaches being used for the design of functional and/or information parts of ITS system architectures:

- Function, or Process Oriented methodologies, using data flows, functional decomposition etc. (applied in Europe, USA, Canada);
- Object Oriented (OO) methodologies, using objects, classes, abstraction, inheritance, encapsulation etc. (applied in ISO Reference Architecture, Japan, and Australia).

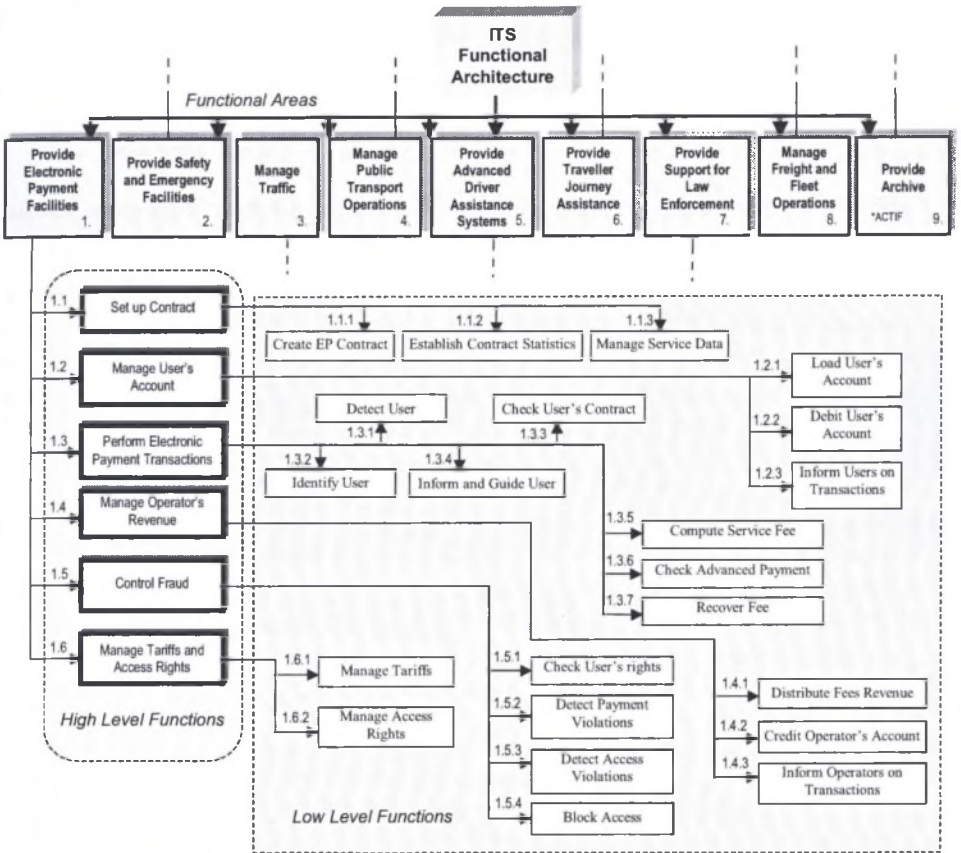

Fig.2. Hierarchical diagram of the functional area *Provide Electronic Payment Facilities*

Both approaches use functions and data but in different ways, with Process Orientation concentrating on the former and OO on the latter. The European Commission funded the KAREN Project that implemented the Process Oriented approach and produced the first ITS Framework Architecture for Europe (the first version published in 2000). Consequently it has been refined within the FRAME Projects [2] to form the basis of ITS Architectures anywhere

in Europe. The Functional Architecture [1] forms part of the European ITS Framework Architecture and therefore shares its characteristics. It defines and describes what functionality needs to be included in a system that can fulfil the requirements of the European ITS Architecture user needs. It also shows how its functionality links to the outside world and in particular the users of the system and the data that is used within the system. The data description is sometimes included in a separate Information Architecture, but for the FRAME project, this has been subsumed into the Functional Architecture. At its highest level, the Function Architecture consists of eight Functional Areas - each area is given a name and a number. French national architecture (ACTIF) includes another functional area (No. 9) called „Provide Archive" (see Fig.2). Each functional area has a simple textual description and contains functions of two types (High Level Functions and Low Levels Functions) whose purpose and activities are related. To show how the functions relate to each other within their Functional Area, the hierarchy diagrams have been created. In Fig.2 there is an example of the hierarchical diagram of the Functional Area "*Provide Electronic Payment Facilities*", a part of which (specifically function 1.3) is discussed later in the paper.

2.2. PROCEDURE USED TO CREATE OO FUNCTIONAL SPECIFICATION

The procedure applied for creating functional requirements specification has consisted of several steps. The first step comprised informal definition of functional requirements.

2.2.1. INFORMAL SPECIFICATION OF FUNCTIONAL REQUIREMENTS

As a primary source of information we have used FRAME Functional Architecture that includes lists of functional requirements contained in definitions of both High Level Functions and especially in definitions of Low Level Functions [1], [10]. For the sake of illustration, in Fig.3 there is an example of specification of functional requirements given for the Low Level Function „1.3.1 Detect User" mentioned in Fig.2. As we can see, functional requirements are expressed in an informal way. Depending on the viewpoint of a system, they can be treated as functions the system has to fulfil (from the viewpoint of the system) or they can be treated as features, the system is offering (viewpoint of the user). Features have to be represented in notations, which have to be intuitively understood.

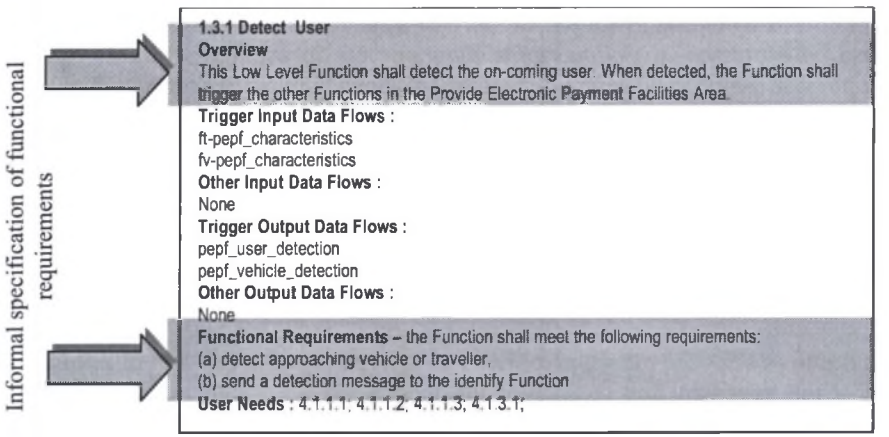

Fig.3. Example of informal functional requirements specification (according to [1])

2.2.2. CHOOSING SUITABLE FORMALISMS

The next step has consisted in selection of suitable formalisms to express different views at the system. Graphical and/or textual formalisms are well suited. For all of these views, the Unified Modeling Language (UML) offers appropriate means of notation. All diagrams of the UML have been standardized by the OMG [6] and all of them are abstract, containing a wealth of constructs to fit into nearly every application domain. The ***Unified Modeling Language*** is a ***notation***; that is a set of diagrams and diagram elements that may be arranged to describe the design of a software system. UML is not a ***process***, nor is it a ***method*** comprising a notation and a process. From available diagrams we have chosen to apply Use Case diagrams (not discussed here), Sequence diagrams, Statechart diagrams and Class/Object diagrams. The proposed methodology was discussed in more details in [4], or applied to the parking system in [9].

Use Case diagrams (not discussed in this paper) show the interactions between use cases and actors. Use cases represent system functionality, the requirements of the system from the user's perspective, i.e. a use case is a description of a set of sequences of actions, called scenarios that a system performs to yield an observable result of value to an actor. A use case describes what a system (subsystem, class, or interface) does but does not specify how the system internally performs its tasks. Actors represent the people or systems that provide or receive information from the system. The difference between an actor and an individual system user is that an actor represents a particular class of user rather than an actual user.

Sequence diagrams are used to show the flow of functionality through a use case, i.e. to show the interactions between objects in the sequential order that those interactions occur. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Sequence diagrams can be used to document how objects in an existing system currently interact. This documentation is very useful when transitioning a system to another person or organization. In Fig.4 there is an introductory part of the Sequence diagram describing behaviour of the system for electronic toll collection. In a similar way different scenarios representing other functionalities of the system can be described. This kind of diagram usually shows sequential

order of interactions between objects; however, in our sequential diagram classes are used instead of objects by reason of simplification. The same information as in Sequence diagrams could also be expressed in *Collaboration diagrams* but in a different manner and with a different purpose (objects and actor interactions are shown without reference to time).

Statecharts [3] are a commonly known visual design notation for specifying the behaviour of state-based reactive and embedded systems. A statechart diagram consists of states, transitions, events, and variables and provides a way to model the various states in which an object can exist. It extends finite automata by the concepts of *hierarchy* (OR states, used to describe sequential behaviour), *concurrency* (AND states, used to describe parallel behaviour) and *compound transitions* (connected via pseudo-states for a concise modelling of complex transitions). Transitions connecting states are labelled by terms of the form $e[c]/a$, where e is an event that triggers the transition if it occurs, and c is a Boolean guard that has to be evaluated to *true* for the transition to fire. The right part a contains a list of actions that are executed if the transition fires. Examples for actions are assignments to variables, method calls, or sending of events. In Fig.5 there is an example of the statechart diagram for the class *cUserDetection*.

The structure of a Statechart diagram is similar to an *Activity diagram.* Activity diagrams may be used in requirements gathering to illustrate the flow of events through a use case – they define where the workflow starts, where it ends, what activities occur during the workflow, and in what order the activities occur. An activity is a task that is performed during the workflow.

Class diagram shows the static structure of the model. Although it is called a class diagram, it may also contain other elements besides classes that exist in a model, such as capsules, protocols, packages, their internal structure, and their relationships to other elements. Class diagrams may be organized into (and owned by) packages, but the individual class diagrams are not meant to represent the actual divisions in the underlying model. A package may then be represented by more then one class diagram. The static structure of our model can be described using the class diagram shown in Fig.6.

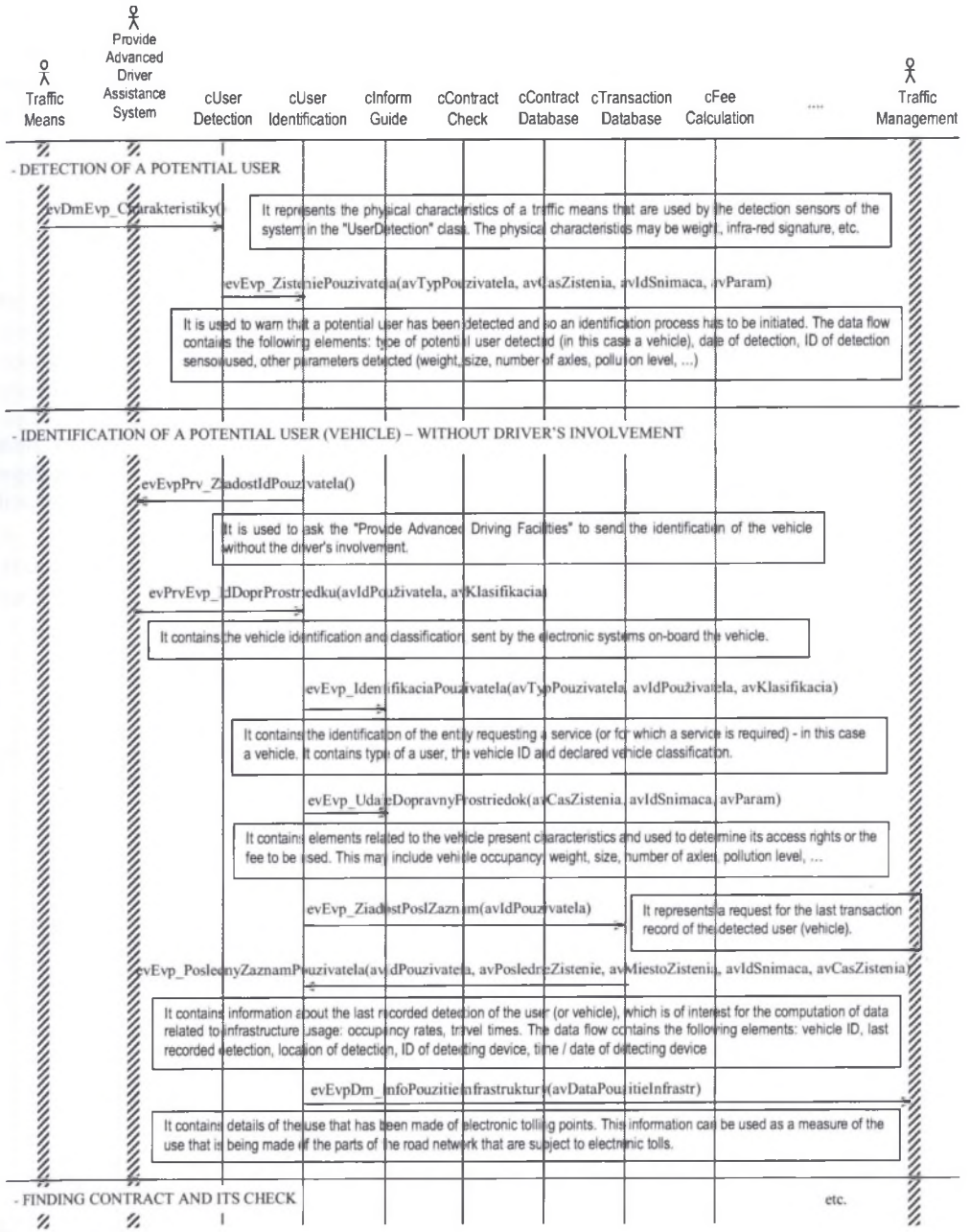

Fig.4. An introductory part of the Sequence diagram – electronic toll collection

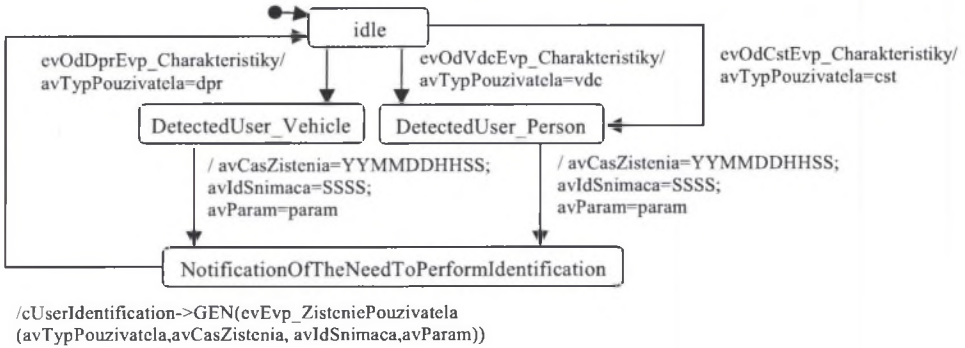

Fig.5. Statechart for *cUserDetection*

To complete survey of diagrams available in UML, the implementation phase of system development usually uses *Component diagrams* and *Deployment diagrams*. The former show the mapping of classes to implementation (software) components, the latter show the physical layout of the network and where the various components will reside.

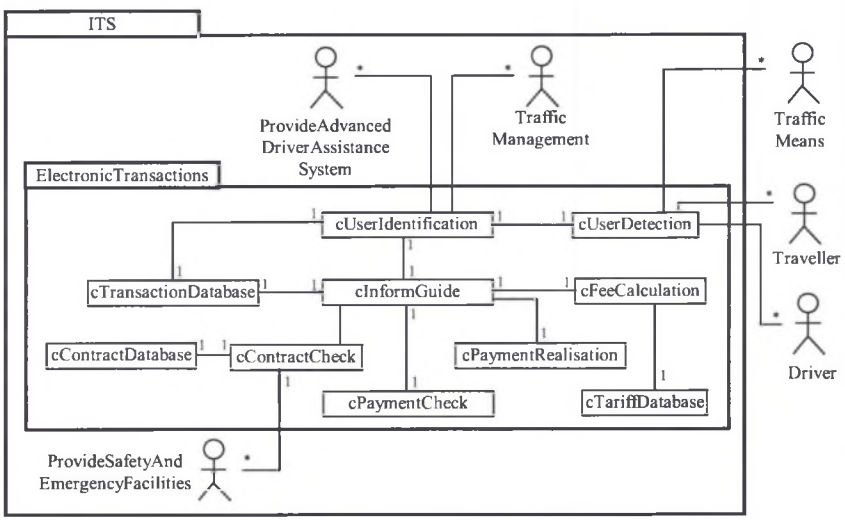

Fig.6. Class Diagram (classes drawn in a simplified view)

2.2.3. CHOOSING A PROPER SOFTWARE TOOL

It's doubtful that anyone would be working with UML these days without the aid of a modelling tool, because these tools are to software design what a word processor is to writing. Currently, more than 80 different UML tools are available [5], each conforming more less to the UML standard. If one looks in detail, not even two of them have exactly the same syntax and semantics. The model partially presented in this paper related to that part of the ITS that is responsible for *Providing electronic payment transactions* as a part of the functional area

Provide electronic payment facilities (as presented in Fig.2). It was created using Rhapsody 4.0 whose graphical symbols are slightly different from the UML standard (the final version of diagrams is expected to be processed in Rational Rose tool not available yet in time of writing this paper. Diagrams presented in the paper were re-drawn manually because of translating some terms from Slovak to English language and making some simplifications.

3. CONCLUSIONS

The use of UML ensures that one comprehensive model and general methodology can be used for the development of functional architecture and standards through to the actual implementation of ITS software and systems in the transport network. Thanks to this every complex system is best approached through a small set of nearly independent views of a model (no single view is usually sufficient). The use of all kinds of diagrams is not obligatory, we apply only use case diagrams, sequential diagrams, class/object diagrams and (if required) statechart diagrams to create functional requirements specification; non-functional and context requirements are to be specified in a different form.

The work has been supported by the Grant Agency of the Slovak Republic VEGA, grant No. 1/1044/04 "Theoretical Foundations for Implementing e-Safety Principles into Intelligent Transportation Systems"; and the RTD Project "Technologies and services for intelligent transportation in the Slovak Republic".

BIBLIOGRAPHY

[1] European ITS Framework Architecture. Functional Architecture. D3.1 Annex 1 - Function Specifications. Ver. 1.1 March 2002, p. 282
[2] FRAME Projects. http://www.frame-online.net
[3] HAREL, D., Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8(3): 231–274, 1987
[4] JANOTA, A., ZAHRADNÍK, J., Methodology Used to Create System Architecture for ITS in Slovakia. In: Advances in Electrical and Electronic Engineering, Vol. 3, No. 2, 2004, p. 105-108
[5] JECKLE, 2004. http://www.jeckle.de/umltools.htm
[6] OMG Unified Modeling Language Specification 1.3. Object Management Group
[7] http://www.omg.org/cgi-bin/doc?formal/00-03-01
[8] Map of the Slovak Road Network for International Transport, 2002 http://www.telecom.gov.sk/vud/idic/index.htm
[9] PIRNÍK, R., HUDEC, R., NAGY, P., Modelovanie detekčných systémov v aplikácii PARK za pomoci nástroja RATIONAL ROSE. In: Vedecká konferencia s medzinárodnou účasťou „Nové smery v spracovaní signálov VII", 12.-14. 5. 2004, Tatranské Zruby, p. 176-179
[10] SVÍTEK, M. et al.: ITS v podmínkách dopravně-telekomunikačního prostředí ČR (802/210/108). Technická správa za rok 2002 - Příloha 2. Informační ITS architektura ČR, ver. 1.0, 2002, pp. 355

Reviewer: Ph. D. Stanisław Krawiec