METHODOLOGY USED TO CREATE SYSTEM ARCHITECTURE FOR ITS IN SLOVAKIA

Aleš Janota, Jiří Zahradník
Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina
Veľký diel, 010 26 Žilina, Slovak Republic, Phone: ++421-(0)41-513 3301, E-mail: {ales.janota|jiri.zahradnik}@fel.utc.sk

Summary The paper deals with an object oriented approach proposed by the authors for creation of the ITS system architecture in the Slovak Republic and shows how a reference architecture can be created as a base for future more detailed architectures (models). The authors characterise possible approaches, explain their relations to existing architectures and propose a methodology based on the Unified Modeling Language (UML). The main attention is paid to the logical part (logical view) of the system architecture that should result in the form of easy readable and understandable UML models.

1. INTRODUCTION

Intelligent transportation systems (ITSs) are typically large-scale and complex systems consisting of diverse technologies. Therefore, when building a system, it is essential to create a structure for the entire system in advance, and then develop specific systems conforming to the structure. 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). A number of nations have taken initiatives to create their own ITS Architectures. Significant effort has also been made in the field of standardisation, especially on the ISO level.

2. STATE-OF-ART OF CREATING ITS ARCHITECTURES

In order to stimulate the development of ITS (also called telematic systems) in Europe, the European Commission funded the KAREN Project that produced the first ITS Framework Architecture for Europe (the first version published in 2000). Consequently it has been refined within the FRAME Projects [1] to form the basis of ITS Architectures anywhere in Europe. In addition to Europe, system architectures for ITS are developed or improved in many other countries. At least some of them are worth mentioning, e.g. ITS in the USA [2], Japan [3], Australia [4], Canada [5], etc.

Generally, there are now 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).

The process-based methodologies were initially used by software engineers because they were already in use in all other fields of engineering. They have been used for all types of systems and in all stages of the life cycle for many decades and they are commonly well understood. Most systems produced in the ITS domain are, or will be, safety-related to some degree. All the current Standards, Guidelines and techniques available to assure safety of a system assume, or are based on, the use of a process oriented approach, e.g. [6]

For those who have been educated in discrete mathematics for many years the use of an OO methodology is as natural to use as a Process Oriented one is for an engineer steeped in function analysis and calculus. The distinctive feature of OO is the "object", which is an instantiation of a class, which combines the data, and behavioural properties of all the objects that might be in the class. By means of this encapsulation the detailed operations can be hidden, thus permitting the system designer to work at higher levels of abstraction, and closer to the application. Another advantage consists in re-use of objects and easier handling of complex systems.

Both approaches use functions and data but in different ways, with Process Orientation concentrating on the former and OO on the latter. It should be noted that, whilst both approaches aim to create a working system, but this does not guarantee a workable one [7]. Neither methodology is sufficient to produce a design for an entire system that includes organisational and institutional issues. Both approaches might need a Computer Assisted Software Engineering (CASE) tool to assist in the maintenance of consistency, especially when the system is large and/or complex.

3. METHODOLOGY PROPOSED

The following ideas concerning the presented methodology originate from solving the government project 472 VÚD/2002-1 „Intelligent Transport Systems within conditions of the Slovak Republic – Stage: Architecture of ITS services and justification of their allocation and levels according to the customers' needs", which the authors participate in.

Since Slovakia is to join the EU, it is legitimate to expect building and integration of the national ITS systems within the European space, respecting the European specifics and conveniences. Majority of national architectures used the ISO System Architecture, known as the TICS (Traffic Information and Control Systems) Reference Architecture [8], as their starting point. The main reason for its creation was to provide a platform for the other groups within the transport telematics part of ISO, rather than to act as a "World-wide Standard" [9].

Tab. 1. TICS fundamental services from ISO [10].

Traveller Information		Freight & Fleet	
1.	Pre-trip Information	18.	Commercial Vehicle Pre-clearance
2.	On-trip Driver Information		
3.	On-trip Public Transport Information	19.	Commercial Vehicle Administrative Processes
4.	Personal Information Services	20.	Automated Roadside Safety Inspection
5.	Route Guidance and Navigation	21.	Commercial Vehicle On-board Safety Monitoring
		22.	Commercial Vehicle Fleet Management
Traffic Management		**Public Transport**	
6.	Transportation Planning Support	23.	Public Transport Management
7.	Traffic Control	24.	Demand Responsive Transport Management
8.	Incident Management		
9.	Demand Management	25.	Shared Transport Management
10.	Policing/Enforcing Traffic Regulations		
11.	Infrastructure Maintenance Management	**Emergency**	
Vehicle		26.	Emergency Notification and Personal Security
12.	Vision Enhancement	27.	Emergency Vehicle Management
13.	Automated Vehicle Operation	28.	Hazardous Materials and Incident Notification
14.	Longitudinal Collision Avoidance	**Electronic Payment**	
		29.	Electronic Financial Transactions
15.	Lateral Collision Avoidance	**Safety**	
		30.	Public Travel Security
16.	Safety Readiness	31.	Safety Enhancement for Vulnerable Road Users
17.	Pre-crash Restraint Deployment	32.	Intelligent Junctions and Links

Regardless of what kind of methodology is to be used, the first step comprises definition of user requirements. For the ISO Architecture, the user requirements are defined as the TICS Fundamental Services [10]. They are divided into 32 Service Groups (Tab. 1).

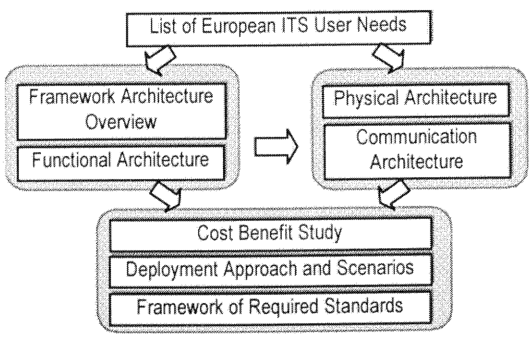

Fig. 1. European ITS Framework Architecture Documents.

The ISO TICS Fundamental Services were used as one of the main starting points for the definition of the European ITS User Needs. However, all other subsequent activities looking towards the European ITS Framework Architecture were based on the process-oriented methodology and resulted in a set of document as shown in Fig. 1 [11]. Since the choice made by ISO has been to use OO methodology and to describe its architecture using the UML, actual comparison with the current European ITS Framework Architecture is difficult (Tab. 2). Some of documents can hardly be created since certain kinds of diagrams (e.g. a context diagram) are not supported by the OO methodology.

Tab. 2. Comparison of the functionality in the ISO and European Architectures according to [9].

ISO TICS Reference Architecture	European ITS Functional Architecture	
Traveller Information	1	Provide Electronic Payment Facilities
Traffic Management	2	Provide Safety and Emergency Facilities
Vehicle	3	Manage Traffic
Freight & Fleet (Commercial Vehicle)	4	Manage Public Transport Operations
Public Transport	5	Provide Advanced Driver Assistance Systems
Emergency	6	Provide Traveller Journey Assistance
Electronic Payment	7	Provide Support for Law Enforcement
Safety	8	Manage Freight and Fleet Operations

There are several identifiable stages in the architecture development process; these include development of *Reference Architecture*; *Logical Architecture*; and Physical *Architecture*. The Reference Architecture is a concise generic framework, which guides the development of more concrete system architectures. It will be used in the future by particular industry groups to develop specific, logical and physical architectures in a cohesive manner. A logical architecture elaborates the conceptual behaviour, and in so doing it provides more detail about the modularity. A physical architecture is reached when the actual distribution of the system modules is defined, thus leading to important implications for communications.

The Reference Architecture, created using the UML, can be broken down into three generalised key concepts that describe the extent of the system and how it is intended to work. These key concepts include "Applications", "Actors" and "Interactions", i.e. the way the applications and actors interact with each other to perform certain tasks (Fig. 2).

Applications are referred to as use cases and actors represent any person or organisation that interacts with the ITS system. In rough terms Use cases will correspond to the eight high level Functional Areas in both the European ITS Framework and US National ITS Architectures. The interactions required between applications (use cases) and actors to perform certain types of tasks are referred to as packages - the major packages are: Transport, Roadway, Vehicles, Events, Operation Interfaces, Roadway Peripherals, Payment, Vehicle Interfaces and Travel terminals. Nine actors represent the outside world – they interact with the architecture and all of them have a hierarchy of other actors underneath them. With the hierarchies included, the total number of actors becomes 33. For illustration, Fig. 4 shows the hierarchy of the actor "User".

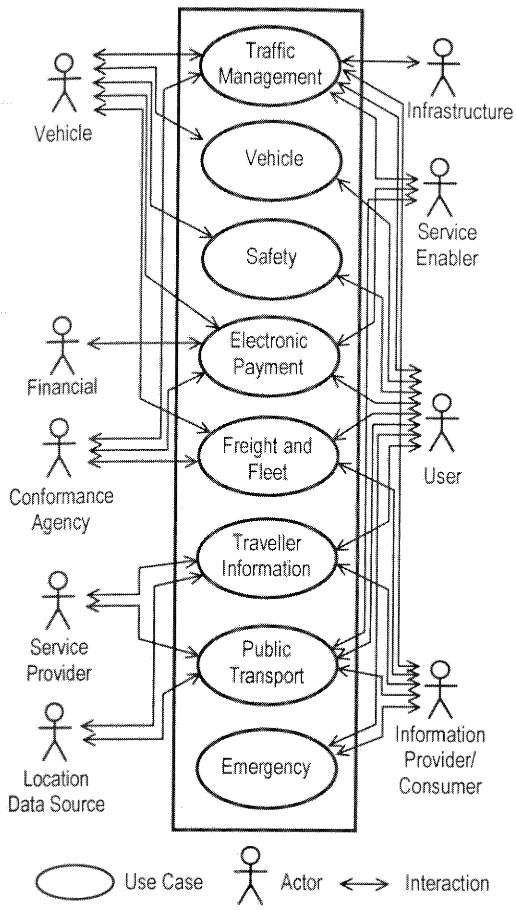

Fig. 2. Top Level Use Case Diagram.

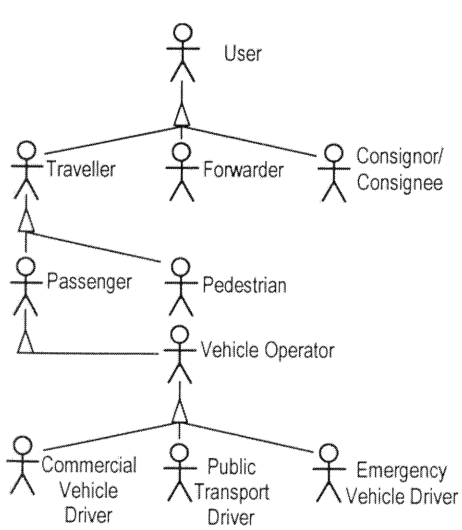

Fig. 3. Hierarchy of Actors of the Type "User".

All these diagrams help to answer the questions: "What applications are included in the system?"; "Who are the actors (users and application providers) included in the system?"; and "How do they interact to perform and provide certain services within the system?".

In addition to these key concepts, other UML diagrams are used, mostly *Class diagrams* (define the abstract elements that comprise the Reference Architecture) and *Sequence (Interaction) diagrams* (describe how the implied objects of the system cooperate to provide the services defined in the use case).

4. USAGE OF METHODOLOGY

One of the key ITS application areas determined for early implementation in Slovakia concerns the service "Electronic Payment". It covers fee collection systems concerning vehicle related transport services such as toll, parking and route guidance as well as fare collection and advanced payments for a wide range of services (Fig. 4.).

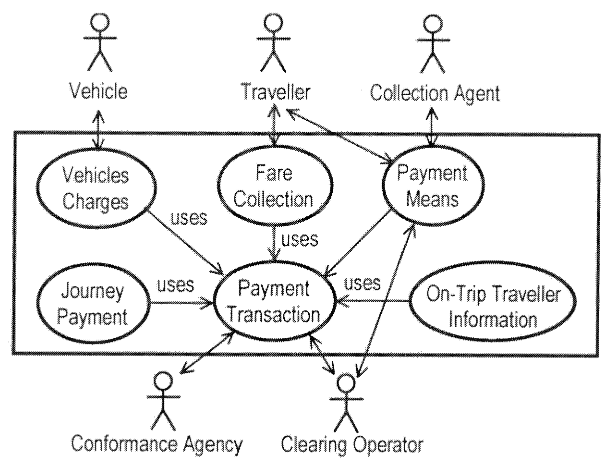

Fig. 4. Electronic Payment Use Case Diagram.

Payment Means: This transaction captures the expression of a contract between the Traveller and the external Issuer via a Collection Agent that allows the Traveller to access the services available in the payment system, e.g. an account in a credit card system or an electronic purse. The Collection Agent transacts the credit to an electronic purse used by a Traveller. A Clearing Operator receives registration of a credit payment means for a Traveller from the resource to be used. *Fare Collection:* These transactions collect fares from urban public transport users for the use of current public transport services, and advanced payments for public transport services and for other (Yellow Pages) services. The point of collection may be pre-embarkation, or on-board. These transactions use the Payment Transaction use case. The Traveller must provide the payment instrument. *Vehicle Charges*: These transactions provide facilities for the electronic fee collection from vehicles as they pass through roadside collection points. They use the Payment Transaction use case. The Vehicle is sensed by a Roadway Peripheral and must carry a payment instrument. *Payment Transaction:* These transactions maintain a centralised store of data on the prices (tariff) being charged for tolls, spaces at parking lots and fares. They enable Vehicle Operators and Travellers to pay for tolls, fares and parking lot charges, plus other (Yellow Pages) services in advance and in the case of Travellers, as part of their journey planning facilities.

The logic of the Electronic Payment transactions is on principle described in the sequence diagram in Fig. 5.

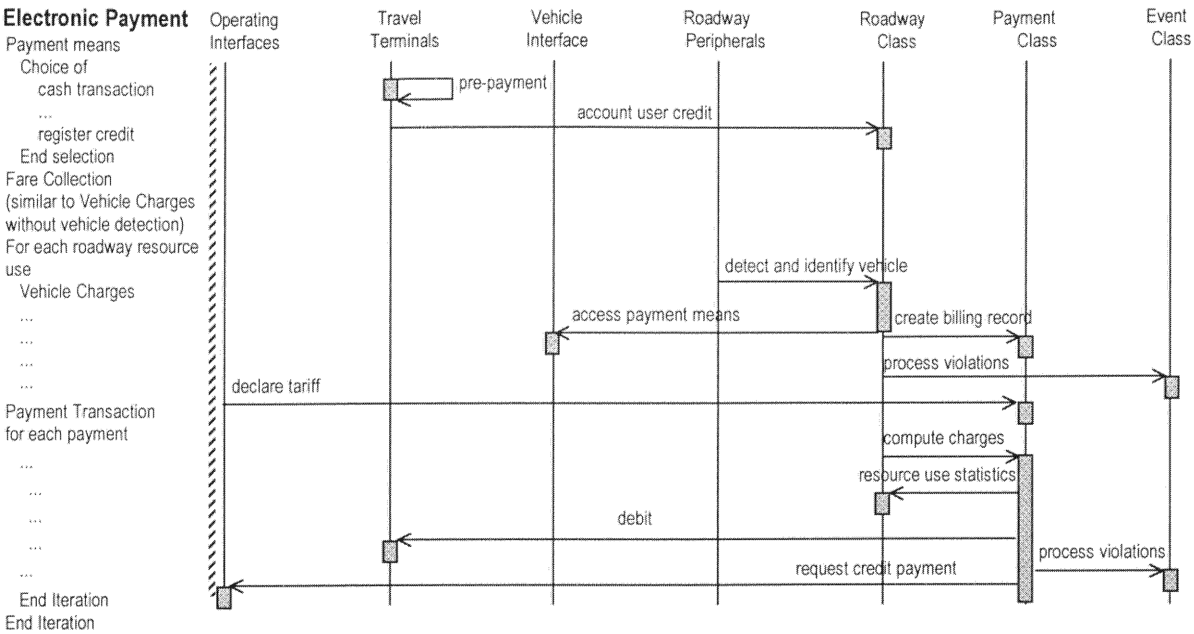

Fig. 5. Sequence Diagram Showing the Main Interactions for Electronic Payment.

Payment Means covers any transaction that takes payment or establishes credit with a TICS service provider. Cash transactions may prime an electronic purse or debit from the purse. These transactions occur at will and any time a Traveller actor makes an up front payment. Each resource use (or fare) met by credit payment involves transactions of two use cases. Vehicle Charges (or Fare Collection) set up the accounting at the start. Payment Transaction is invoked at the end of service. There is dependency on the Event Class for processing payment violations. The Roadway Class is dependent on the Payment Class for all transactions, which involve payment for use of resources. Operations of the Vehicle Interfaces class provide the payment means for vehicles. Operations of the Travel Terminals class support other Traveller payment. Operations of the Operating Interfaces class provide interface to the Financial actors.

5. CONCLUSIONS

Use of the UML ensures that one comprehensive model and general methodology can be used for the development of architecture and standards through to the actual implementation of ITS software and systems in the transport network. Thus:

- Every complex system is best approached through a small set of nearly independent views of a model - no single view is sufficient;
- Every model can be expressed at different levels of fidelity; and
- The best models are connected to reality.

Therefore it as proposed to adopt a methodology based on UML for documenting the Slovak Reference Architecture being created. A commercially available CASE tool is to be used to implement this approach.

The work has partially 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".

REFERENCES

[1] FRAME Projects. http://www.frame-online.net
[2] ITS America.http://www.itsa.org
[3] System Architecture for ITS in Japan http://www.iijnet.or.jp/vertis/e-frame.html
[4] ITS Australia, http://www.its-australia.com.au
[5] Canadian ITS Architect., http://www.its-sti.gc.ca
[6] STN EN 61508 Funkčná bezpečnosť elektrických/ elektronických/ programovateľných elektronic-kých bezpečnostných systémov. Časť 1: Všeobecné požiadavky
[7] Functional Architecture–Main Document. Version 1.1. European ITS Framework Architecture, March 2002 (148 p.)
[8] ISO TC204 WG1, Transport Information and Control Systems - Reference Model Architecture(s) for the TICS Sector - Part 2: Core TICS Reference Architecture, ISO/PDTR 14813-2, 1998.
[9] European ITS Framework Architecture Overview (D 3.6) Issue 1, August 2000
[10] ISO TC204 WG1, Transport Information and Control Systems - Reference Model Architecture(s) for the TICS Sector - Part 1: TICS Fundamental Services, ISO/TR 14813-1, 1998.
[11] Planning an Intelligent Transport System – A Guide to System Architecture. European Communities. Why you need one and how to create it. Issue 1, October 2002 (20 p.)