A service description framework for intelligent transport systems: applied to intelligent goods

Åse Jevinger[a] , Paul Davidsson[a,b], Jan A. Persson[a,b], Gideon Mbiydzenyuy[a] and Shoaib Bakhtyar[a]

[a]*Blekinge Institute of Technology, PO Box 214, SE-374 24 Karlshamn, Sweden*
[b]*Malmö University, SE-205 06 Malmö, Sweden*

ARTICLE INFO ABSTRACT

Keywords:
Service framework;
Service description;
Intelligent goods;
Service composition;
ITS

Purpose
The primary objective of this paper is to propose a framework for specifying and analyzing Intelligent Transport System (ITS) services.
Design/methodology/approach
Essential service information are identified and stated in a uniform way. A separation into an abstract description, focused on what the service does, and a concrete description, focused on how to perform the service tasks, is made.
Findings
The proposed a framework has been used for specifying and distinguishing ITS services and for different types on analyses, e.g. service composition/decomposition analyses and architecture analyses.
Original/value
The usefulness is illustrated by an architecture choice analysis applied on a specific intelligent goods service. Apart from ITS services, the framework has the potential to also apply to other types of information services.

1. Introduction

Representing services in a way that that reveals the essential information, such as, input, output and service functionality in a uniform way, is critical when conducting certain types of service analyses, e.g. in service composition/decomposition analyses, synergy analyses or architecture analyses. For each service involved, it must be possible to specify all information relevant for the analysis and the descriptions must furthermore enable a clear distinction between the different services (i.e. capture the characteristics of each service). The aim of the suggested framework is to fulfill these requirements. Moreover, as an additional contribution to service analysis, a service composition/decomposition approach is also included in the framework. Since the framework is focused on services and their descriptions, a short introduction to the service concept is given below.

Over the years, researchers within both business science and other fields have argued for different definitions of the service concept. Literature thereby contains an abundance of various definitions. The Organization for the Advancement of Structured Information Standards (OASIS) works for open standards for the global information society, and the organization has defined a service as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description"[1]. This definition is consistent with the suggested framework since the framework prescribes a uniform interface that can be used to access the service and to get information about constraints and policies. The service description thereby enables services to be accessed and exercised consistent with constraints and policies.

This paper focuses on Intelligent Transport Systems (ITS) and, in particular, services provided by such systems to different users. According to ISO 14813-1, 2007[2], an ITS (user) service consists of a product or activity targeted to a specific type of ITS user addressing a given user need. One of the fundamental challenges in the study of ITS services for various users has been to have a common, uniform and standard terminology as well as taxonomy for ITS services analogous to the Web Services Description Language. Existing literature about ITS services focuses on the description of enabling technologies, functionalities (or information) and operational characteristics of the services[3, 4, 5, 6]. Describing services with focus on the technologies obscure non-technical but important issues associated to such services. Using functionalities to describe services maybe ambiguous if the level of detail for each functionality is not the same[5]. Describing services based on operational characteristics enable such services to be categorized according to operational domain, e.g. safety[4, 5], but such an approach does typically not include detailed specifications of the services and analyses of such services may thereby be limited.

Given the multiplicity of ITS services, both emerging and existing, upfront knowledge that may facilitate implementation is increasingly of great value considering the high costs of field experiments. Further, there is a tendency among stakeholders to prioritize different services (hence the need to compare services)

due to limited financing[7, 8]. Consequently, the current ITS service specifications (as outlined in previous paragraph), raise a number of questions, including: 1) how can service composition/decomposition analyses be performed if the services are specified with different schemes, 2) how can service architecture options as well as alternative implementation options be analyzed if the service descriptions do not reveal appropriate information? However, to completely and unambiguously specify all information related to a service is very hard, if not impossible – and this goes for both machine processable descriptions and descriptions intended for human reading. A complete description is usually not needed though. Depending on the intended purpose of the descriptions, as well as the set of services in question, different levels of detail are required. As a fundamental requirement however, descriptions must enable a clear distinction between different services, i.e. different services must differ in their descriptions. In this paper a framework is proposed that can be used to describe ITS services. The service characteristics are reflected by a number of service description elements which are sufficient for our particular set of services and our analysis purposes. The framework has been applied on several services, primarily from three different fields within ITS (intelligent goods, e-freight, and vehicle centric services), and we therefore believe that the identified elements may apply for a large set of ITS service. It might even apply for other IT related services, in particular for services focused on information. The framework is primarily designed for information services, where input and output are important factors, and not for services in which, for instance, physical actions or changes of state are central. Other studies are more focused on these areas. In particular, Zinnikus et al.[9] work with software agents and describe services in terms of changes of state.

Additionally, apart from being used for specifying ITS services, the proposed framework can be used for different types of service analyses. In this paper we will show how an architecture choice analysis can be performed on a specific intelligent goods service. In summary, the primary objectives of this paper are as follows:

- Review different approaches for specifying services, in particular ITS services, and to understand how such frameworks are used for ITS service analysis.
- Based on the review, propose a framework for specification and analysis of ITS services that:
    - Supports uniform descriptions of ITS services, i.e. describes what an ITS service do (what service it provides to its users), what input it needs and what output it produces
    - Supports unambiguity, i.e., if two descriptions are the same, it's the same service
    - Supports various levels of detail
    - Is implementation (technology) independent
    - Supports the combination of services into complex services and vice versa
- Illustrate how the proposed framework can be employed for an architecture choice analysis (as a pre-study, the framework has been validated by employing it for different types of analyses on services from different ITS fields, see section 4).

This paper is structured as follows. In section 2, the results from the literature review are discussed. Section 3 presents the service description framework, which is subsequently applied to a specific intelligent goods service, in section 4. Finally, in section 5, some conclusions are drawn.

2. Literature review

The framework presented in this study concern the specification, including characteristics and functional perspectives, as well as the analysis of ITS services. In particular, the framework shows how services can be represented and their interactions and relationships identified. Our perspective on ITS services can be compared to the approach used for Service Oriented Architecture (SOA) in software engineering and computer science. The concept SOA is defined as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains", by OASIS[1]. Generally speaking, a SOA includes different services cooperating in a location transparent manner. A system is hereby composed of a set of interacting services, where each service has well-defined functionality and an interface that can be published and discovered[10]. This means that the service metadata need to be defined and communicated, and common specifications used for these purposes are WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description Discovery and Integration). These specifications do not support complete uniform descriptions of services but are limited to metadata in order to facilitate information exchange between services with different capabilities. According to the SOA Reference Model[10], the functionality of the SOA service, might be described either as machine processable descriptions or as human readable descriptions. The SOA however provides a potential approach that can be used to study a service at different levels (each service seen as a composition of subservices) by proving a means of uniformly specifying the interface between various service levels.

The SOA concept can be seen as an evolution of component-based architectures. A component-based architecture is simply made up of software components, meant to interact with other components. One of the ideas behind component-based architectures is that software should be componentized in order to reuse the components for different purposes[11]. These components may be distributed across many machines and a system represents a composite of these interacting components. A number of software component models have been presented in literature, which defines what components are, how they can be composed or assembled, how they can be deployed and how to reason about their quality[11]. The framework presented in this paper can be used for both composition and decomposition of services, as will be shown in the analysis part of the paper.

Representing a service in a way that reveals the essential information, such as, input and output, and the service functionality that together can be used to characterize a service constitute the perspective adopted in this study. Quite a lot of framework studies have been carried, with focus on different perspectives of ITS[12, 13, 14, 15, 16]. Two trends have emerged from the literature on ITS related framework studies which can be referred to as category based approach and a hierarchical based decomposition/composition approach. Both approaches have been used to address different analysis perspectives of ITS from analysis of service concepts for deployment[18], to analysis of service composition and specification that will answer questions related to architectures. These general approaches to different extent support some of the objectives of this paper (as stated above).

In the category based approach, focus is on the description of ITS services (and systems) so as to disclose essential information that enables such services to be associated together in groups addressing a common service application area (or domain), e.g. traffic management services. Hence, a category based approach enables the description of what the service do, especially the outputs expected from the service. Several reasons exist for attempting to categorize services. For instance, travel and traffic management, public transportation management, electronic payment etc. are suggested categories or groups to enable development of service repositories[5]. Others have concentrated on development area and application domain to enable development of system architecture[4]. To minimize variation in ITS service specification across regions and nations, ISO proposed a taxonomy consisting of 11 categories (purpose based) that identifies the domains within which the ITS services reside in the current perception of the ITS sector[2, 3]. When services are described using a category based approach, their description content reflects the category in question and not the internal details and functional behavior of the service. As such, a category based approach do not necessarily require that service descriptions are uniform, or advocate a detail specification of services that can enable them to be composed to complex services.

In the hierarchical based decomposition/composition approach, studies have focused on specification of ITS services from general to specific elements sometimes referred to as subservices. The Freight Functional Framework (FFF) proposed by the THEMIS project (air and rail freight sectors) recommends different roles, function profiles and information flows[13, 17]. Roles are further decomposed in a hierarchy according to different levels with more specific roles lower in the hierarchy. As such the hierarchical based approach provide one way of specifying a service in different levels as in the SOA approach, except that focus is shifted from the interface between services to the functionalities of the service. A framework for integrating existing services where such services have been decomposed into subservices with focus on information type, content, calculation process and visualization is proposed by the European project eMotion[14]. Service chains ranging from user request (PULL), subscription (PUSH) and monitoring (PULL) are considered. A difference in comparison to the THEMIS FFF is that no hierarchy of roles and functions is considered, instead, the technology aspect is accounted for in terms of basic services. In general a hierarchical based approach for ITS services provide one way not only to specify different levels of services (according to service functionality), but also to achieve a technology independent specification of services as well as to compose simple services to complex services.

As mentioned above, one of the fundamental challenges that have led to the development of frameworks has been to have a common, uniform and standard terminology as well as taxonomy for ITS services[2]. Within the Web Service community, research has reached quite far in these matters, addressing specific issues such as automatic discovery, composition and execution of web services. WSDL[19], for instance, focuses on machine readable descriptions of the protocol bindings and message formats required to interact with a web service. It is based on XML and it includes interface information about the available functions, data type information for the message requests and responses, binding information about the transport protocol to be used, and address information for locating the service (e.g. <xs:element name="body" type="xs:anyType" minOccurs="0"/> can be seen as a WSD for a service that can be called with name “body” with data,“any type” returning a minimum of zero). The use of WSDL, however, still requires human interaction to a large extent since the programmer has to manually search for and invoke appropriate web services in a useful manner. In other words, it does not fully describe web services in a machine-understandable fashion. As an extension to WSDL, Semantic Annotations for WSDL (SAWSDL) was released by W3C in 2007. SAWSDL is a lightweight specification for adding semantic annotations to WSDL. It aims at specifying how semantic annotation is accomplished (for instance, to annotate WSDL interfaces and operations with categorization information that can be used to publish a Web service in a registry) using references to semantic models, e.g. ontologies. Further, Web Service Modeling Ontology (WSMO) and Semantic Markup for Web Services (OWL-S) are two major frameworks for automation of web service usage with the help of semantic descriptions. WSMO and OWL-S provides support for further automation for discovering or composing services. Some use of SAWSDL can today be seen in a variety of systems, and at the same time researchers continue the work by developing generic extensions to SAWSDL in order to cater for additional functionalities, e.g. WSMO-Lite. However, in many cases the solutions adopted are application-specific and hence highly dependent on the technology. However, in addition to supporting machine readable interface specification, extensions of WSDL require the use of annotations for describing message communication between web services and facilitate the composition of such services into more complex services.

ITS services like any other services are composed from multiple technological elements or functions that may be common with other services[4]. In addition, services can be seen as building blocks that can be combined for deployment in a variety of ways. However, little work has been done that takes advantage of the composability of ITS services in order to facilitate service analysis and understanding. Composition of web services have proven to be a useful technique when services have to “co-operate” with each other in the sense of B2B[20, 21]. Such composition can either take place in real time during service operations as in the case of web services or in the planning stage similar to web service proactive composition[20]. Further, composition/decomposition approaches of ITS services are sometimes used to address architectural analysis, implicitly assuming that components offer well-defined services which are composed to meet system requirements similar to Architectural Description Languages (ADLs) for web services[20]. Thus approaches developed for decentralized web service compositions based on AND/OR trees[21] may be extended to analyze ITS services. In their proposed approach[21] a formalization of Input/Output Dependency, AND/OR Tree and AND/OR tree search algorithm was used for analysis of web service composition. We believe, however, it is important that a service framework does not only allow for composition/decomposition of services but also provide support for describing services uniformly and unambiguously at different levels in a technology independent fashion. In Table 1 we summarize our views on how the different approaches to service related frameworks either fully, partially or do not address the objectives of the proposed framework in this paper. Under the various framework approaches we refer the reader to the examples discussed above, e.g. WSDL, covers the different variants such as SAWSDL, WSMO etc.

Table 1. Our view on the different approaches to service related frameworks

Framework approach	Description of what service does	Uniform service description	Unambiguous service description	Different levels of details	Technology independent description	Combination to complex services
SOA	partially	partially	no	yes	Partially	no
Categorical	partially	no	no	no	Partially	no
Hierarchical	partially	no	no	yes	yes	yes
WSDL	partially	yes	yes	partially	no	yes

From the above review, this study aim to suggest a framework for ITS services that support the stated objectives as much as possible.

3. A description framework for ITS services

The service description framework has been developed primarily for describing and analyzing ITS services. Below we list a number of different types of potential analyses that we suggest can be performed based on the framework:

- Service composition/decomposition
- Information and functionality synergies analysis
- Service quality analysis
- Physical architecture analysis

This list is however not exhaustive and we believe that the framework can be used for more areas than presented here.

3.1. Abstract service description

Service descriptions are required in order to analyze, explain or compare different services. These descriptions should be as precise and condensed as possible. In particular, the descriptions must enable a distinction between different services in a set and they must furthermore be flexible enough to provide as much information as needed to perform intended analyses. In this paper, we propose a set of mandatory elements that can be used to specify ITS services. The collection of these elements represents an Abstract service description since it is only concerned with *what* the service does, leaving out the question of *how* for the moment. The elements are Input, Output, Processing, Information Entities, Triggers and Procedures. The last three elements represent sets that are used by the first three element sets to describe a service.

According to the proposed framework, an Abstract service description is specified by the tuple <I,O,P,IE,T,R>, containing the following six elements:

Input: $I = \{i_1, i_2,\ldots, i_{ia}\}$, where *ia* is the number of input parameters and $i_k = <what, when>$, *what* ϵ *IE* and *when* ϵ *T*

Output: $O = \{o_1, o_2,\ldots, o_{oa}\}$, where *oa* is the number of output parameters and $o_k = <what, when>$, *what* ϵ *IE* and *when* ϵ *T*

Processing: $P = \{p_1, p_2,\ldots, p_{pa}\}$, where *pa* is the number of processing parts and $p_k = <what, when>$, *what* ϵ *R* and *when* ϵ *T*

Information Entities: $IE = \{ie_1, ie_2,\ldots,ie_{ea}\}$, where *ea* is the number of information entities relevant to the service and ie_k is an information entity

Triggers: $T = \{t_1, t_2,\ldots,t_{ta}\}$, where *ta* is the number of triggers relevant to the service and t_k is a trigger

Procedures: $R = \{r_1, r_2,\ldots,r_{ra}\}$, where *ra* is the number of procedures relevant to the service and r_k is a procedure such that $r(In) = Out$, where $In \subseteq IE$ represents potentially multiple inputs and $Out \subseteq IE$ represents potentially multiple outputs

The IE set represents the set of Information Entities that might serve as input or output. The T set specifies all events that serve as triggers for input/output generation or processing activities (e.g. a trigger for when to measure a temperature). A service may be reactive meaning that a processing activity must be triggered by an external action, or proactive with for instance a periodic behavior triggered by an internal clock. These behaviors are reflected by the events in T. R includes all procedures needed to describe what a service does. The three sets IE, T and R, are domain specific (i.e. different subject domains have different sets of possible values) and their contents (the components) can be described using for instance natural language (see section 4) or pseudocode. In order to compare and relate different services to each other, common ontologies are, however, required.

The above defined sets may actually be empty. For instance, services with no input data will be described without the Input element. Please note that the Abstract service description is focused only on what the service does, not how (the aspects concerning *how* a service performs a task are addressed by the Concrete service description below). In particular, the procedures in the Processing element must strictly concern what the service does, not how the service is implemented. Therefore, an appropriate level of detail of the Processing element information must be found. For our purposes, the Processing element can be expressed with relatively procedures (see section 4), or even in natural language. However, more advanced formal methods might sometimes be needed and in such situations the Z-language or some other formal specification language can be used.

During the development of this framework a substantial amount of services, primarily from three different ITS fields (intelligent goods, e-freight and vehicle centric services) have been examined and described according to the Abstract service description. We therefore believe that the identified service description elements may apply, as a fundamental implementation independent service description framework, for a relatively large set of ITS services. One of the major advantages with using this framework is that it provides a high-level condensed and purposive description that can easily be used for different types of analyses.

The Abstract service description is illustrated below with a simple example, a real-time vehicle tracking service. The elements in the R, T and IE sets have self-explaining names and explicit explanations of them have therefore been left out.

IE = {vehicleID, currentPosition, currentStatus, time}

T = {timeInterval, startServiceReq}

R = {ReadPosition() = currentPosition , ReadStatus() = currentStatus }

$I = \{i_1\}$, where i_1 = <vehicleID, startServiceReq>

$O = \{o_1, o_2\}$, where o_1 = < currentPosition, timeInterval>, o_2 = < currentStatus, timeInterval>

P = {p_1, p_2}, where p_1 = <ReadPosition, timeInterval>, p_2 = <ReadStatus, timeInterval>

The input to the service is the ID of the vehicle and the service is initiated by a start request holding the input data. The service will then provide the current position and status of the vehicle every time interval. The IE *time* can be seen as an internal information entity and is therefore not included in the input or output elements. In general, input parameters can be excluded from the definition if they can be seen as system internal since these might be unknown (for instance, alternative input parameters might exist, depending on which implementation solution is used).

In the above example, p_1 and p_2 does not explicitly specify the input/output parameters connected the two procedures since these are already defined by *R*. However, if the outputs of a procedure serve as inputs to another, it is possible to let p_1 and p_2 include this information as well (see section 4).

3.2. Concrete service description

The service description framework basically consists of two different service description levels; the Abstract service description and the Concrete service description. These two represent different levels of specification such that the Concrete service description is a further specification of the Abstract service description. Apart from describing what the service should do, the Concrete service description is also concerned with *how* the service should fulfill these tasks (though the aim is not to capture the complete description of how a service should operate but only the aspects needed for the desired purpose of the description) . Since a number of alternatives to this may exist, there might be several Concrete service description instances. Each service description level is defined by a set of mandatory elements and since the Abstract service description elements describe the functional requirements that must be fulfilled, the Concrete service description also incorporates these elements. However, the information specified for each element may be more extensive in the Concrete service description, for instance additional input parameters might be needed for the specific solution. In particular, the Processing element may, in the Concrete service description, also include information about how to perform specific tasks. In addition to this, the Concrete service description includes an optional *where* part of the service description elements, which specifies the origin or destination of the input/output information (which might differ from their current physical location), and the location of the processing units. The *where* part thereby introduces architectural aspects to the Concrete service description level. In summary, the aspect of *how* the service fulfills its tasks is captured in the Concrete service description by additional procedures as well as more specified information about them, i.e. extended and more specified P and R sets (with focus on *how*). It is also capture by extended and more specified I, O and IE sets, additional information about origin or destination of input/output and location of the processing units.

Different values of the service description elements indicate different services. The elements of the Concrete service description are described below. Please note that the Concrete service description requires the specification of a 4:th set , L, which reflects the different location alternatives on input origin, output destination and processing location. Also note that the number of output parameters are equal in the Abstract and Concrete service descriptions, whereas the number of input parameters may be larger in the Concrete service description.

According to the proposed framework, a Concrete service description is specified by the tuple <I,O,P,IE,T,R, L>, containing the following seven elements:

Input: $I = \{i_1, i_2, \ldots, i_{ib}\}$, where $ib \geq ia$ is the number of input parameters and i_k = <*what*, *when*, *where*>, *what* ∈ *IE*, *when* ∈ *T* and *where* ∈ *L*

Output: $O = \{o_1, o_2, \ldots, o_{ob}\}$, where $ob \geq oa$ is the number of output parameters and o_k = <*what*, *when*, *where*>, *what* ∈ *IE*, *when* ∈ *T* and *where* ∈ *L*

Processing: $P = \{p_1, p_2, \ldots, p_{pb}\}$, where $pb \geq pa$ is the number of processing parts and p_k = <*what*, *when*, *where*>, *what* ∈ *R*, *when* ∈ *T* and *where* ∈ *L*

Information Entities: $IE = \{ie_1, ie_2, \ldots, ie_{eb}\}$, where $eb \geq ea$ is the number of information entities relevant to the service and ie_k is an information entity

Triggers: $T = \{t_1, t_2, \ldots, t_{tb}\}$, where $tb \geq ta$ is the number of triggers relevant to the service and t_k is a trigger

Procedures: $R = \{r_1, r_2, \ldots, r_{ra}\}$, where $rb \geq ra$ is the number of procedures relevant to the service and r_k is a procedure such that $r(In) = Out$, where $In \subseteq IE$ represents potentially multiple inputs and $Out \subseteq IE$ represents potentially multiple outputs

Locations: $L = \{l_1, l_2, \ldots, l_{lb}\}$, where *lb* is the number of location alternatives relevant to the service and l_k is a location alternative

The Abstract service description example above can be further specified according to the Concrete service description, as shown below. Please note that the description naturally can be even further expanded and specified in details, if required.

IE = {vehicleID, currentPosition, currentStatus, time}

T = {timeInterval, startServiceReq}

R = {ReadGPSPosition() = currentPosition , ReadODBStatus() = currentStatus}

L = {back-office, vehicle}

$I = \{i_1\}$, where i_1 = <vehicleID, startServiceReq, back-office>

$O = \{o_1, o_2, o_3\}$, where o_1 = < currentPosition, timeInterval, back-office >, o_2 = < currentStatus, timeInterval, back-office >, o_3 = < currentPosition, timeInterval, back-office >

$P = \{p_1, p_2\}$, where p_1 = <ReadGPSPosition, timeInterval, vehicle>, p_2 = <ReadODBStatus, timeInterval, vehicle>

The start request is initiated by back-office, the processing units are located on the vehicle and the outputs are destinated to back-office. The additional output o3 is needed since there might be several vehicles serving back-office with their respective position and status. Back-office needs to know which vehicle the output belongs to. This output is not included in the Abstract service description since it might not be necessary in other implementations of the service.

The two procedures specify how the position and the status should be read (through GPS and the ODB port). No input parameters are needed into the two procedures since they are located on the vehicle, and in is the position and status of the very same vehicle that should be read.

3.3. Service composition/decomposition, basic approach

In this section we present a composition/decomposition approach, which is based on the two service description levels, with Abstract service description instances alternated by

Concrete service description instances. For simplicity, the Abstract service description is denoted *Service description level A*, in this section, and the Concrete service description is denoted *Service description level B*.

In this approach, a service decomposition always starts with a topmost A-level instance describing what the service should do, i.e. the requirements of the service. The set of A-level input parameters may, as indicated above, exclude input parameter that can be seen as internal to the system, and one of the objectives with the approach is therefore to support the identification of all required input parameters, including the internal ones. As a second step, the top-most A-level instance will be broken down into B-level instances, one for each solution alternative. Each B-level instance hereby specifies complete lists of inputs corresponding to the different options. The input parameters specified in the superior A-level instance must be included in the corresponding B-level instances. Moreover, the optional *where* parts of all B-level service description elements have been excluded from the basic approach, since it should be able to serve as a foundation for a variety of analyses, including those without architectural aspects. However, if architectural aspects are required, the *where* parts can easily be included when applying the approach. In this case, the different B-level instances, representing different solution alternatives, might contain different values of the *where* part of the elements, i.e. different solution alternatives might reflect different placements of for instance the procedures (an example of this is shown in section 4).

All input parameters related to the B-level instances must be retrieved/received from somewhere. If the B-level input parameters must be produced by a secondary service (i.e. subservice), additional A-level instances should be defined describing all subservices needed to produce these input(s) (unless a discontinuation of the analysis is desired at this level). Since several subservices might be needed to produce the inputs to a B-level instance, there may be several A-level instances corresponding to one superior B-level instance. Based on the mandatory service description elements, all A-level instances corresponding to one superior B-level instance are hereby complements (ANDs, e.g. providing as outputs the different inputs required by the superior level B instance) whereas all B-level instances corresponding to a superior A-level instance are substitutes (ORs, e.g. requiring different inputs but providing the same outputs). There is one exception to these rules however. If a redundancy in the system is required, implemented using several different B-level instances which all produce the same output, then they are all complements since they are all required. Redundancy may be motivated by, for instance, a requirement to make a system more robust and reliable. The complement/supplement relationships may also, naturally, change with additional, self-defined service description elements.

Please note that exactly one B-level instance from each B-level split are required to implement a corresponding service (since they are substitutes). The composition/decomposition approach is flexible in the sense that an analysis might be terminated at any level. The point of termination naturally depends on the desired level of analysis but also on the type of service in question.

Using the above approach it is possible to decompose a service, and to conduct various types of analyses on the resulting structure. Such a structure can be illustrated using for instance trees, "black input/output boxes" or tables. In this paper we will primarily illustrate the composition/decomposition approach by building tree structures (i.e. AND/OR trees), where the tree nodes correspond to A- and B-level instances. The leaves will always accommodate information entities or B-level instances. Please note that each service, represented by an A-level instance, might be used both as a subservice and as an individual service. Thus, for a service composition the tree will be continued upwards instead of downwards.

An example of a tree based on the approach is shown below. Since the service description elements identify the A- and B-level instances, two different services may be represented by the same tree structure. In order to separate the trees, we therefore also include some element values; input and output. We have, however, in this particular example only included the *what* part of the Input and Output elements, for readability reasons. In the tree, the following symbolic representations are used:

- An A-level instance is denoted as: $A_i^{I\rightarrow O}$, with input set *I* if mandatory inputs exist, output set *O* and index *i*
- A B-level instance is denoted as: $B_i^{I\rightarrow O}$, with input set *I*, output set *O* and index *i*
- ORs are represented by:
- ANDs are represented by:

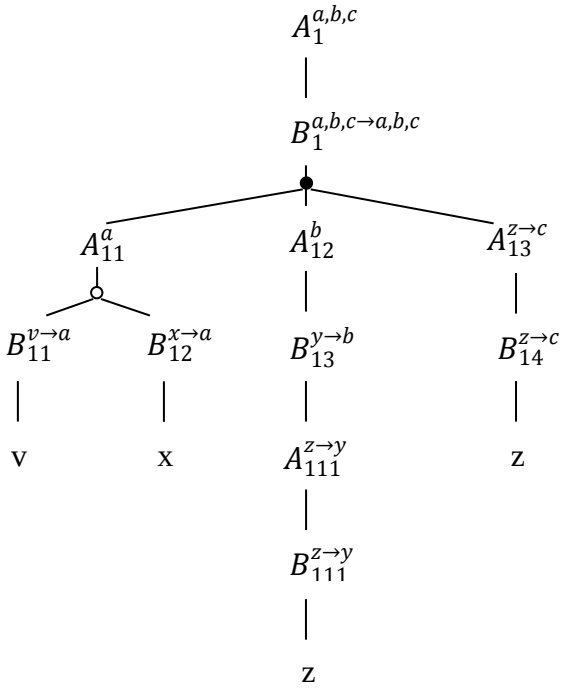

Figure 1: Composition/decomposition, basic approach

As Figure 1 shows, the service description of A_1 does not include any input parameters. This means that no input requirements have been set, which possibly opens up for more alternatives that might being explored in the analysis. A_{13} on the other hand specifies the input parameter requirement *z* which is fulfilled by B_{14}. The tree furthermore shows that A_1 needs subservice B_1, which takes the same input as output parameters. B_1 might for instance be responsible for assembling and delivering the output information to some specified recipients. B_1 in turn is dependent on all of the services described by A_{11}, A_{12} and A_{13}. A_{11} can be implemented using either B_{11} or B_{12}. As can be seen, both B_{111} and B_{14} in the figure need the same input

parameter (z). One conclusion that can be drawn by this tree is that B_{111} and B_{14} potentially may be merged into the same subservice, represented by a new B-level instance taking z as input and producing y and c as output. This would also mean that A_{111} and A_{13} would be merged into one A-level instance, with both y and c as output parameters. If the new merged subservice was already known as an existing subservice when building the AND/OR tree, the merged instances would have been inserted from the beginning. In general, when building AND/OR trees, depending on what type of analysis is performed there might be several considerations that must be taken into account. For instance, there might be several architectural limitations affecting the structure of the tree. All these preconditions must be specified before using the composition/decomposition approach.

3.4. Service composition/decomposition, alternative approach

Depending on the purpose of a particular service composition/decomposition, an alternative approach, based on the two service description levels, may be applied. As in the previous approach, the service description starts with a topmost A-level instance describing what the service should do and which output it should deliver. However, instead of performing the analysis stepwise backward, searching for services to deliver input to the previous service, a service is further broken down into sub-services, which in turn are broken down into even smaller sub-services and so forth. This means that the top-most A-level instance is broken down into B-level instances which may be both complements and alternatives to each other. These B-level instances may in turn be broken down into further B-level instances and the previous alternation between A- and B-level instances does hereby not exist in this approach. In principal, the two approaches are identical since a service description based on one of the approaches can be transformed into a service description based on the other. The example tree in Figure 1 can hereby be transformed into the tree shown in Figure 2. We believe that this alternative approach might be preferred when a known service needs to be broken down into subservices, whereas the original approach is beneficial when the complete scope of a service needs to be investigated. Please note that the tree corresponding to the first approach actually indicates which parts of the service (i.e. B-level instances) need to communicate.

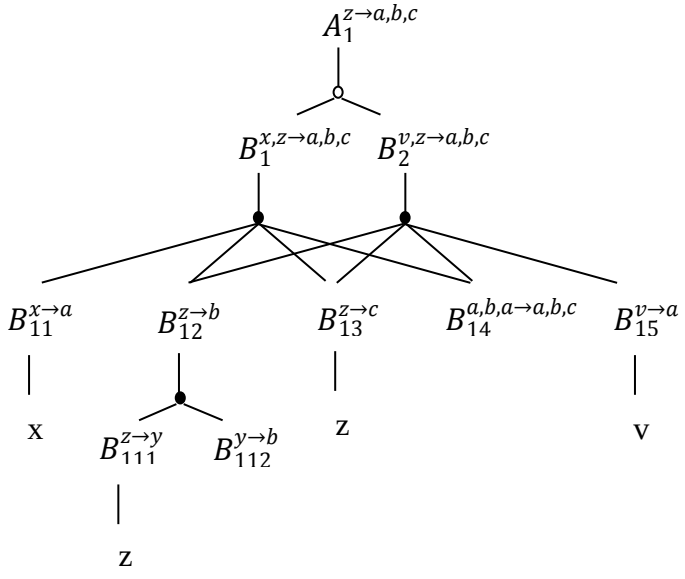

Figure 2: Composition/decomposition, alternative approach

This quality is further explained and utilized in the architecture analysis (see section 4).

4. Application of the service description framework on a potential intelligent goods service

This section illustrates the service description framework by applying it to a potential intelligent goods service. Apart from generating a service description, a very simple architecture analysis is performed to illustrate the basic composition/decomposition approach. The aim of the analysis is to identify the information entities needed as well as possible combinations of where to place different processing parts of a service. Based on these, the required communication paths will also be investigated. In order to provide a simple and clear analysis, alternative solutions other than different processing locations, have been excluded.

The main purpose with the selected service is to inform about the priority of goods. This information can be used to facilitate decisions about which goods to load onto a vehicle. The service is activated by an incoming request which includes the accountable ID and itinerary of a departing vehicle. Here, accountable IDs reflect different organizations, for instance haulage contractors, assigned to be responsible for the goods on different transport links. If the two inputs match the upcoming accountables and the itinerary of the goods, an answer is generated by the service with the priority (otherwise another vehicle should be selected).

Below, the service description according to the framework is presented. The description starts with a number of tables specifying the *IE*, *T*, *R* and *L* sets necessary for this particular description and analysis (see Tables 2-5). These sets are a result of the subsequent analysis. Table 6 shows all identified Abstract and Concrete service description instances. A_1 represents the main service to be analyzed whereas A_2 represents a sub-service. The corresponding service descriptions can be found in the table. Figure 3 shows the tree resulting from the basic service composition/decomposition approach. The "+" mark on some of the information entities indicates possible plurality, i.e. one or several information elements of the same type. Furthermore, each Concrete service description instance has a G, V, I or M attached to it, which show the *where* part of the Processing element.

Table 2: *IE* = {IE1, IE2, IE3, IE4, IE5, IE6}

Information entity	Description
IE1	Unique ID of the goods
IE2	Priority of the goods
IE3	Itinerary of the goods, including stop positions, delivery time windows and accountable IDs for each transport link
IE4	Next destination of the goods, must be updated at every stop
IE5	Accountable ID associated with a vehicle
IE6	Itinerary of a vehicle

Table 3: *T* = {ERequest, EPush, EPull, AnsReady}

Event	Description
ERequest	An incoming request is received
EPush	Information entity is received
EPull	Information entity is retrieved
AnsReady	An answer has been created and is ready to be delivered

Table 4: *R* = {CollectPresentItemData, FindMatchingItems, GetItemsData, GetPresentItemsData, GetSurroundingGoodsData, IdentifyPresentItems, MatchDestAcc}

Procedure	Description
CollectPresentItemData() = ($IE1^+$, $E2^+$, $IE3^+$, $IE4^+$)	Asks another service process for information about IE1, E2, IE3 and IE4 of all present goods items. Returns this information.
FindMatchingItems(IE5, IE6) = ($IE1^+$, $IE2^+$)	Finds and returns IE1 and IE2 from all present goods items whose itinerary matches the input parameters.
GetItemsData($IE1^+$) = (IE1, E2, IE3, IE4)	Finds and returns IE1, IE2, IE3 and IE4 for every IE1 parameter by accessing stored information about this.
GetPresentItemsData() = ($IE1^+$, $E2^+$, $IE3^+$, $IE4^+$)	Finds and returns IE1, E2, IE3 and IE4 from all present goods items.
GetSurroundingGoodsData() = ($IE1^+$, $E2^+$, $IE3^+$, $IE4^+$)	Finds and returns IE1, IE2, IE3 and IE4 of all nearby goods items, which are identified through broadcast request.
IdentifyPresentItems() = ($IE1^+$)	Finds and returns IE1 of all goods items present by accessing stored information about this.
MatchDestAcc($IE1^+$, $IE2^+$, $IE3^+$, $IE4^+$, IE5, IE6)= ($IE1^+$, $IE2^+$)	Searches for IE4 in IE3 for each input goods item and identifies all upcoming destinations. Compares these destinations to the destinations listed in the planned vehicle itinerary. Also compares the accountable IDs for each transport link. Returns corresponding IE1s and IE2s for each match.

Table 5: *L* = {Sender, Driver, Loader, Process, G, V, I, M}

Location entity	Description
Sender	Sender of the goods
Driver	Driver of a transporting vehicle
Loader	Loader of goods into a transporting vehicle
Process	Another service process
G	Goods level
V	Vehicle level, i.e. transporting vehicle
I	Infrastructure level, e.g. back-office systems
M	Local mobile terminal

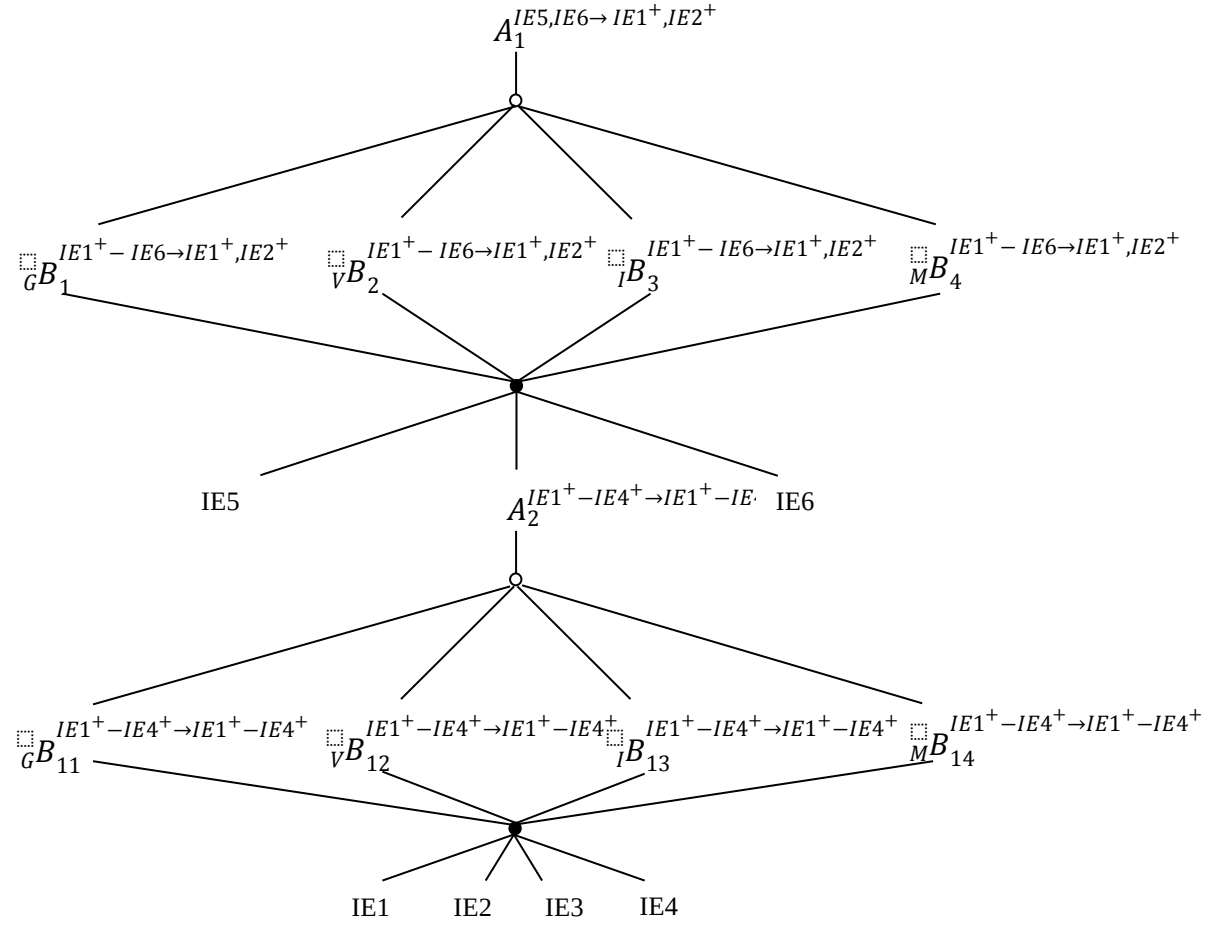

Figure 3: Decomposition tree of the intelligent goods service; inputs, outputs and location are shown for each instance

Table 1: Identified Abstract and Concrete service description instances

Ins.	Input (I)	Output (O)	Procedures (f)	Processing (P)
A_1	i_1 = <IE5, EPush> i_2 = <IE6, EPush>	o_1 = <IE1$^+$, AnsReady> o_2 = <IE2$^+$, AnsReady>	FindMatchingItems(IE5, IE6)	<f, ERequest>
B_1	i_1 = <IE1$^+$, EPull, Sender>	o_1 = <IE1$^+$, AnsReady, Loader>	MatchDestAcc(	<f, ERequest, G>
B_2	i_2 = <IE2$^+$, EPull, Sender> i_3 = <IE3$^+$, EPull, Sender>	o_2 = <IE2$^+$, AnsReady, Loader>	CollectPresentItemData(), IE5, IE6)	<f, ERequest, V>
B_3	i_4 = <IE4$^+$, EPull, Sender> i_5 = <IE5, EPush, Driver>			<f, ERequest, I>
B_4	i_6 = <IE6, EPush, Driver>			<f, ERequest, M>
A_2	i_1 = <IE1$^+$, EPull> i_2 = <IE2$^+$, EPull> i_3 = <IE3$^+$, EPull> i_4 = <IE4$^+$, EPull>	o_1 = <IE1$^+$, AnsReady> o_2 = <IE2$^+$, AnsReady> o_3 = <IE3$^+$, AnsReady> o_4 = <IE4$^+$, AnsReady>	GetPresentItemsData()	<f, ERequest>
B_{11}	i_1 = <IE1$^+$, EPull, Sender> i_2 = <IE2$^+$, EPull, Sender>	o_1 = <IE1$^+$, AnsReady, Process> o_2 = <IE2$^+$, AnsReady, Process>	GetSurroundingGoodsData()	<f, ERequest, G>
B_{12}	i_3 = <IE3$^+$, EPull, Sender>	o_3 = <IE3$^+$, AnsReady, Process>	GetItemsData(	<f, ERequest, V>
B_{13}	i_4 = <IE4$^+$, EPull, Sender>	o_4 = <IE4$^+$, AnsReady, Process>	IdentifyPresentItems())	<f, ERequest, I>
B_{14}				<f, ERequest, M>

Within ITS, three basic building blocks are sometimes used for characterizing ITS services; Data Acquisition, Data Processing and Information Distribution[22]. These building blocks can be identified in Figure 3. A_1 is responsible for the Information Distribution whereas A_2 is responsible for Data Acquisition. Both A_1 and A_2 involve Data Processing.

Even though the analysis performed in this section is, as stated above, very simple, it illustrates a number of outcomes from using the service description framework. For instance, the tree identifies a sub-service required by the main service, which can be placed in a separate location (see Figure 3). It also exposes all information entities needed and which instance they are required by. Since the *where* parts of the framework is included, the tree furthermore indicates the necessary communication paths. For instance, the connection between B_1 and B_{11} (via A_{11}) implies that IE1 and IE2 need to be transferred from B_{11} to B_1.

Table 7 shows the different processing location alternatives and the resulting communication paths, derived from the above analysis. In this analysis all locations have been assumed to possess all necessary capabilities (memory storage, processing communication capabilities etc.), in order for the evaluation to cover all the different alternatives. Furthermore, for simplicity the information entities are assumed to be present wherever needed, i.e. they might be stored redundantly. For a more comprehensive analysis, the location of the information entities can be investigated by inserting additional columns for these as well. The "-" sign in the last column indicate a required communication path. A local mobile terminal is assumed to generate the initial priority request.

Solution 11 in Table 7 represents a completely centralized solution, where the local mobile terminal sends its request to the infrastructure level. Information about which goods items are present, for instance in the terminal, must thereby be stored at the infrastructure level. After processing the request, the infrastructure level sends an answer. Solution 1 on the other hand is a pure intelligent goods solution. The local mobile terminal broadcasts its question to the nearby goods. The goods processes and answers the request. The rest of the solutions require varying degrees of intelligence on the goods and the vehicles, and they represent different levels of viability. In particular, in solution 16 the mobile terminal has all information and performs all processing tasks, which might be seen as an unrealistic scenario.

Table 2: Location and communication path analysis

No	B_x	where	B_{xx}	where	Req.*	Com. req.
1	B1	G	B11	G	M	M-G
2	B2	V	B11	G	M	M-V, V-G
3	B3	I	B11	G	M	M-I, I-G
4	B4	M	B11	G	M	M-G
5	B1	G	B12	V	M	M-G, G-V
6	B2	V	B12	V	M	M-V
7	B3	I	B12	V	M	M-I, I-V
8	B4	M	B12	V	M	M-V
9	B1	G	B13	I	M	M-G, G-I
10	B2	V	B13	I	M	M-V, V-I
11	B3	I	B13	I	M	M-I
12	B4	M	B13	I	M	M-I
13	B1	G	B14	M	M	M-G
14	B2	V	B14	M	M	M-V
15	B3	I	B14	M	M	M-I
16	B4	M	B14	M	M	none

*) Location of the requester

5. Conclusions and future work

This paper summarizes a review of framework approaches that are currently used for specifying ITS user services. The review shows that most ITS related framework studies either follows category based or hierarchical based decomposition/composition approach. In the former the main aim is to be able to categorize ITS services into different domains whereas in the latter, the focus is to specify the various components of the service. In order to be able to compare, and in some situations also analyze, different services a precise and condensed description of each service in question is needed. We have in this paper proposed a framework for specifying and distinguishing (primarily) ITS services. As an extension of this, the framework can also be used to analyze services from different perspectives. In particular, a composition/decomposition approach has been presented and an architecture analysis has been performed. The architecture analysis is based on a relatively simple intelligent goods service since the objective was to, in a comprehensive way, show an example of how this type of

analysis can be performed. In the analysis we used tree structures to represent the service, which revealed the required communication paths between different parts of the service. It also enabled an investigation of different placements for different parts of a system. As a result, an exhaustive set of possible solutions, covering both functional and architectural aspects, to the service were identified. This type of analysis can be used to identify the capability requirements of different parts of a system for different service solutions. It might thereby serve as input to the discussion of advantages and disadvantages with solutions based on intelligent goods. Finally, as a part of the service analysis, a uniform definition of the selected intelligent goods service was also presented.

Even though the service framework requires concrete presentation of processes, it focuses on the process rather than the technology for the process. It also allows for a variation in the scope of services but it does provide a systematic composition/decomposition that can reveal if there are differences in the specification of two services. While the framework was proposed from the mindset of ITS services, it should be possible to apply it for non-ITS services. In the future the Information Entities can be used to assess the availability of relevant useful information existing for different services. Possibilities of using tree structures for assessing of benefits and costs of services can also be explored. Synergy analysis of Information Entities and service processes can furthermore be analyzed if the services are described following the suggested framework.

Acknowledgement

We wish to thank ITS Sweden and the Swedish National ITS Postgraduate School, Vinnova (The Swedish Governmental Agency for Innovation Systems), and Trafikverket (The Swedish Transportation Administration) for funding this research. The article is a result of work done in the course “Software Technologies for ITS” offered by the Swedish National ITS Postgraduate School, 2011.

References

1. MacKenzie, M.C.; Laskey, K.; McCabe, F.; Brown, P.F.; Metz, R.. *Reference Model for Service Oriented Architecture 1.0*, OASIS Standard, 2006.
2. ISO/TR 14813-1. *Intelligent Transport Systems-Reference model architecture(s) for the ITS sector-Part 1. ITS Service Domains, Service Groups and services*, 2007.
3. ISO/TR 14813-1. *Intelligent Transport Systems-Reference model architecture(s) for the ITS sector-Part 1. ITS Service Domains, Service Groups and services*, 1999.
4. ITS Japan. *System architecture for ITS Japan-detailed definition of user services*, ITS Japan together with National Police Agency, Ministry of International Trade and Industry, Japan, 1999.
5. FHA-US- DOT. *ITS user services document*, report prepared for the Federal Highway Administration-US DOT, 2005.
6. Williams B. *Intelligent Transport Systems Standards*, Artech House Publishers, 2008.
7. Jung, B.D.; Kwon, Y.; Kim, H.; Lee, S.W. *A Study on Determining the Priorities of ITS Services Using Analytic Hierarchy and Network Processes*. Advances in Hybrid Information Technology Lecture Notes in Computer Science, 2007; Vol. 4413, pp. 93-102.
8. Khademi, N.; Mohaymany, A.S.; Shahi, J. *Intelligent Transportation System User Service Selection and Prioritization Hybrid Model of Disjunctive Satisfying Method and Analytic Network Process*, Journal of the Transportation Research Board, 2010; Vol. 2189, pp. 45-55.
9. Zinnikus I.; Komazec, S.; Facca, F. *An integration of semantically enabled service oriented architectures and agent platforms*, of Lecture Notes in Business Information Processing, 2012; Vol. 98, pp. 76-94.
10. World Wide Web Consortium (W3C). *Web Services Glossary*, http://www.w3.org/TR/ws-gloss/.
11. Lau K.; Wang Z. *Software Component Models*, IEEE Transactions on Software Engineering, 2007; Vol. 33(10), pp. 709-724.
12. Verweij K.; Ter Brugge R; Ballhaus, F. *General Evaluation Framework to assess the impact of ATT applications in freight transport*, Towards an Intelligent Transport System, first world congress on applications of transport telematics and intelligent vehicle highway systems; Paris, 1994; Vol. 3, pp. 1452-9.
13. TRD International SA. *Synthesis of a common Intermodal Architecture Framework taking into account Traffic Management Systems (TMS)*, Presentation of the Freight Functional Framework, THEMIS Project, 2000-2004.
14. eMotion Project. *Service Definition*, Deliverable No. D 2, 2006.
15. Lee, D.; Kang, H.; Kim, D.; Han, K. D*evelopment of a telematics service framework for open services in the heterogeneous network environment*, 11th International Conference on Advanced Communication Technology, ICACT, 2009; Vol. 1, pp. 741-747.
16. Veera Ragavan, S.; Ponnambalam, S.G.; Ganapathy, V.; Teh, J. *Services integration framework for vehicle telematics*, Lecture Notes in Computer Science, 2010; Vol. 6425, pp. 636-648.
17. THEMIS Project WP3. *Assessment of the Freight Functional Framework and the feasibility of business cases*, 2002 (internal document).
18. Tsao J. *A Framework for Evaluating Deployment Strategies for Intelligent Transportation Systems*, Intelligent Transportation Systems Journal, 2001; Vol. 6, pp. 141-173.
19. Chinnici R.; Moreau, J.-J.; Ryman, A.; Weerawarana, S. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C Recommendation, 2007.
20. Higels, S.; Lewis, D. *State of the Art: Service Composition*, M-Zones Deliverable 1, 2003; pp. 88-136.
21. Ma, X.; Dong, B.; He, M. *AND/OR Tree Search Algorithm in Web Service Composition*, IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008; Vol. 2, pp. 23-27.
22. PIARC. *ITS Handbook*, 2nd ed., Andrew Barriball, 2004.