

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATION SCIENCES



ANALYSIS of ITS ARCHITECTURES

Using eCall Example

Diploma Thesis

Mert Aksaç

May 2019



K620.....Department of Transport Telematics

MASTER'S THESIS ASSIGNMENT

(PROJECT, WORK OF ART)

Student's name and surname (including degrees):

Bsc. Mert Aksaç

Code of study programme code and study field of the student:

N 3710 – IS – Intelligent Transport Systems

Theme title (in Czech): **Použití architektury ITS při vývoji systému**

Theme title (in English): ITS Architecture Usage in a System Development

Guides for elaboration

During the elaboration of the master's thesis follow the outline below:

- Analysis of ITS projects lifecycle, management and architecture
- Analysis of legal and standardization framework for ITS system deployment
- State of the art in ITS architectures, benefits, usage, theory
- Comparison of the analysed approaches, examples, best practices
- Use of 2 ITS architecture approaches with one real world ITS project, conclusions

Graphical work range: standard


Accompanying report length: minimum of 55 pages

Bibliography: Jesty, P.; Bossom R. Using the FRAME Architecture for planning integrated Intelligent Transport Systems, IEEE Forum on Integrated and Sustainable Transportation Systems, 2011
 Perallos, A; et. all. Intelligent Transport Systems : Technologies and Applications, 2015


Master's thesis supervisor: **Ing.Petr Bureš,Ph.D.**


Date of master's thesis assignment: **May 12, 2017**
 (date of the first assignment of this work, that has be minimum of 10 months before the deadline of the theses submission based on the standard duration of the study)

Date of master's thesis submission: **May 28, 2019**
 a) date of first anticipated submission of the thesis based on the standard study duration and the recommended study time schedule
 b) in case of postponing the submission of the thesis, next submission date results from the recommended time schedule




Ing. Zuzana Bělinová, Ph.D.
 head of the Department of Transport Telematics





doc. Ing. Pavel Hrubeš, Ph.D.
 dean of the faculty

I confirm assumption of master's thesis assignment.



Bsc. Mert Aksaç
 Student's name and signature

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF TRANSPORTATION SCIENCES
DEPARTMENT OF TRANSPORT TELEMATICS



Diploma Thesis

ITS Architecture Usage in a System Deployment

Mert Aksaç

Supervisor:
Petr Bureš

28th of May 2019

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

Abstract

Since nearly all modern transportation applications maintain some sort of computerized solution, most of the outputs of those systems are in form of digital information. To make it easier to build and sustain transportation projects and to develop, gather and manage the data flowing between their separate parts, several ITS Architecture Frameworks have been developed around the world. In this paper, an example ITS, eCall will be analyzed in terms of ITS Architectures. Architectural Descriptions for eCall in FRAME and a corresponding Service Package in Arc-IT will be created to contrast the products and methodologies of both Architecture Frameworks. Additionally a way to develop FRAME will be proposed.

Keywords

Architecture, Architecture Description, Architecture Framework, System, System Engineering, Intelligent Transportation System, ITS, FRAME, Arc-IT, eCall, E Call, Mayday Notification Service, Enterprise Architect

Abstrakt

Většina výstupů moderních dopravních aplikací má formu digitálních informací. Aby bylo možné snadněji budovat a udržovat dopravní projekty a rozvíjet, shromažďovat a spravovat data mezi jejich jednotlivými částmi, bylo na celém světě vyvinuto několik rámcových ITS Architektur. V této práci je analyzován systém eCall v prostředí 2 rámcových ITS architektur. Popis systému eCall je vytvořen v evropské architektuře FRAME a americké architektuře Arc-IT. Tyto popisy jsou vzájemně porovnány a navržen postup vývoje evropské architektury za použití vizuálního nástroje Enterprise Architect.

Klíčová slova

Architektura, Popis architektury, Architektonický rámec, Systém, Systémové inženýrství, Inteligentní dopravní systém, ITS, FRAME, Arc-IT, eCall, E Call, Mayday Notification Service, Enterprise Architect

Contents

| | |
|--|----|
| Chapter 1 - Introduction | 55 |
| Chapter 1.1 - What are architecture descriptions why do we need them?..... | 1 |
| Chapter 1.2 - The European ITS Framework Architecture | 5 |
| Chapter 1.3 - Architecture Reference for Cooperative and Intelligent Transportation | 12 |
| Chapter 2 - Methodology | 20 |
| Chapter 3 – The Practical Analysis | 55 |
| Chapter 3.1 - The eCall Example | 21 |
| Chapter 3.2 - Creating an Architecture Description of The eCall Example in FRAME..... | 23 |
| Chapter 3.3 - Creating an Architecture Description of The Mayday Notification Services in Arc-IT | 39 |
| Chapter 4 - A New Tool for Modelling the Architectures | 55 |
| Chapter 4.1 - Introducing Enterprise Architect | 55 |
| Chapter 4.2 - Describing eCall on Enterprise Architect | 56 |
| Chapter 4.3 - Proposal: Modelling FRAME on Enterprise Architect | 61 |
| Chapter 4.4 - Creating an Architecture Description of The Mayday Notification Services in Enterprise Architect..... | 72 |
| Chapter 5 - Conclusion | 75 |
| References..... | 77 |

Chapter 1 - Introduction

Chapter 1.1 - What are architecture definitions and why do we need them?

Any kind of human endeavour requires cooperation between a number of individuals and requires them to make use of various tools. Today one hardly imagines any accomplishment in any major domain of human activity such as building, manufacturing, governing, even in education and entertainment, where it doesn't require many people to work together in small or large groups and to utilize certain instruments, machines or appliances that most probably require some sort of specialized knowledge to be operated well, and in a cooperative manner. There were major projects throughout history that one immediately sees a great need of orchestration. Construction of Roman Roads, Aqueducts and The Great Pyramids all required great management and organization. Similar effort continued throughout human history, mostly with construction projects, military organization etc. As man continued to advance in science, to develop knowledge in fields such as biology, physics and social science the need for organized complexity grew bigger in means of survival, making understanding of environment, coping with and surviving in it. We developed the need for "continuity and unity in a reality fragmented and desegregated into disciplines, languages, approaches and conceptions." (Checkland 1981)

Average citizens of the 21st century spend their entire lives surrounded by establishments that employ a vast array of methods and mechanisms, internal and external systems while fulfilling their activities and operations in all aspects of daily life, such as finance, production and procurement, supply chain and logistics, research and development and communications etc. These systems often coexist and cooperate, most of the time in seemingly distant parts of the same organization or even in different organizations and on an international level. An average person is surrounded by a massive network of systems at their service or which they serve to, to make their livings. Systems today bring together unprecedented opportunities and solutions that depend on skills and knowledge from many different fields, on the other hand since everything is getting smart and connected to each other, they are getting more and more complex and challenging to build and utilize.

The level of complexity is rather high in the field of Intelligent Transportation Systems where "advanced sensor, computer, electronics, and communication technologies and management strategies are being applied in an integrated manner to improve the safety and efficiency of the surface transportation system"^[1].

The domain of solving system problems is system engineering. System architecting is one of the tools and methods offered in system engineering. A systems architecture is a conception and a formal description of that system^[2]. System Architecting is defining a system and its fundamental concepts and properties, as a whole, in its environment and throughout its lifecycle. It is an effort to document and communicate what the system(s) will be consisted of, how the system(s) will behave, how they will be constructed and maintained and finally how they will be retired at the end of their use. System Architecting is a response to the conceptual and practical difficulties of the description and the design of complex systems^[3].

The simplest way to describe a system is by using language – English –. The other ways include graphics, charts and diagrams.

Systems are described through models. Today's modern architectures allow to model systems in their entirety, from their simplest element to the highest level, the interaction between their elements and the relationship with their environment. Architectures for systems can be modelled in many levels and in many different views so that all of its features are defined and all concerns related to them are covered. It is possible to visualize all system technologies and also all other aspects rather than solely the technology, even the roles and responsibilities of the people and institutions gathered around the technologies, goals and objectives and procedures and protocols may be included in the models in various ways. The state of the art products enables to investigate the architectures in very high level allowing to view the models of the systems reacting with their environment, and very low level, displaying how data flow from one process to another, in a single tool.

The international standard ISO-IEC-IEEE 42010 proclaims that the work products of system and software architecting are architecture descriptions, which are products expressing a systems architecture, be it through plain language or complicated models that consist of many layers.

There is no single definition for what should be included in a system's architecture description. The architecture of a system constitutes what is essential for and the fundamental properties of the system of interest and the definition for what is essential or fundamental varies from system to system. An architecture for a system may contain any or all of the following: elements of the system, how these elements are arranged and integrated, principles of their design and organization and how the system should evolve throughout its life cycle. A single system may have more than one architectures, for when the system is in different environments or for different phases of the system's life cycle. A single architecture may have been expressed through several distinct architecture descriptions.

Stakeholders are defined as the parties who have interest with the particular system(s). Stakeholders plan and invest in the system(s) to satisfy certain needs and to benefit from them in one way or another. A system may have various purposes for different stakeholders. The stakeholders' interests in the system(s) are expressed as Stakeholder Concerns. The concerns may manifest the stakeholders' needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues pertaining to the system. A system's architecture addresses these concerns.

An architecture view is the description of the system addressing concerns particular to some group of stakeholders or describing some certain aspect, property or perspective of the system. An architecture is best understood through multiple views. An architecture description includes and identifies one or more architecture views.

An architecture view expresses the architecture of the system in accordance with an architecture viewpoint. Architecture viewpoints govern the views and establish conventions for constructing, implementing and analyzing the view to address concerns framed by that viewpoint. The conventions that Architectures Viewpoint include languages, notations, model kinds, design rules, and/or modelling methods, analysis techniques and other operations on views.

Each Architecture View brings together one or more architecture models. Architecture models describe some or all parts of the systems(s) of interest following certain Model Kinds. A Model Kind set conventions and rules to model systems and how the models should be interpreted. Model Kind could be one of the following: language, design languages, mathematical and statistical models, use case modelling, data flow diagrams, class diagrams, interactive demonstrations, state models etc.

The ideas presented above will be conceptualized below (see Figure 1).

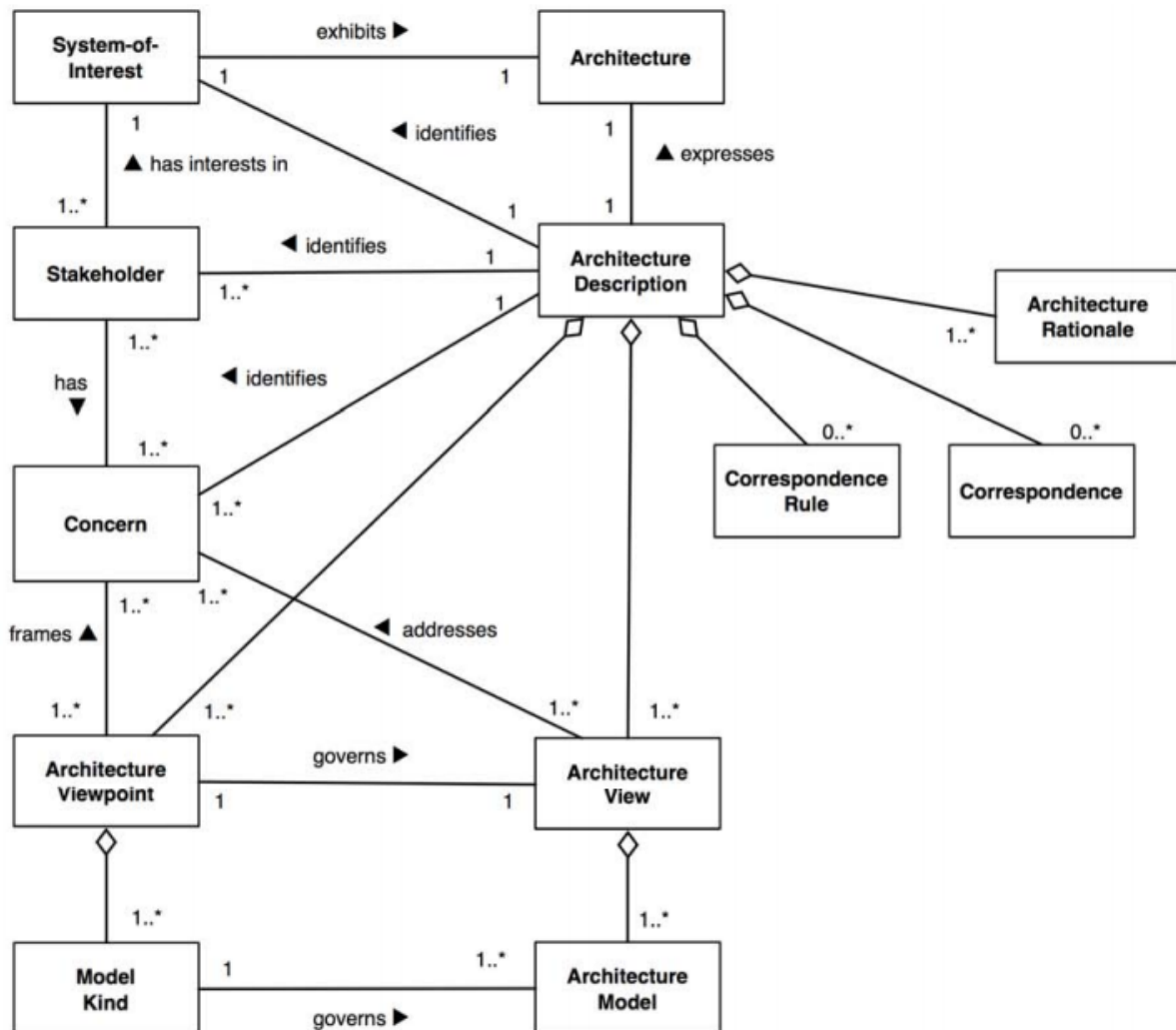


Figure 1 - The conceptual model of an architecture description (source: [2])

“Conceptualization of a system’s architecture, as expressed in an architecture description, assists the understanding of the system’s essence and key properties pertaining to its behaviour, composition and evolution, which in turn affect concerns such as the feasibility, utility and maintainability of the system.” (ISO-IEC-IEEE 42010)

Architecture descriptions are used by the parties that create, utilize and manage modern systems to improve communication and cooperation, enabling them to work in an integrated, coherent fashion. They provide a strategic framework for design choices, development plans and investment decisions. There are many benefits of using architecture descriptions to communicate

about systems. They empower all participants to understand the same system. The work can be broken down into smaller pieces that may be started being built simultaneously instead of suspending execution waiting for completion of parts as it would be in a “piecemeal” development. The architecture descriptions provide stakeholders with strong documentation. Project owners would possess clearly described, optimized requirements for the required service that they can directly include to their tenders. Having solid stated requirements would provide a “basis for a common understanding of the purpose and functions of the systems, thus avoiding conflicting assumptions”^[4] while tendering, contracting or implementing projects. The main advantage of architecture descriptions is that they allow the system(s) to be tested and evaluated way earlier than when they will be actually built. This makes it possible to ensure the system can be planned in a logical manner, integrates well with other systems, meets the desired performance levels and has the desired behaviour as well as to decrease any cost that would be implied to correct errors or make changes in the systems. Spotting and correcting errors on a plan would be tens or hundreds even thousand times less costly than trying to correct them after the systems are built and deployed.

The architecture descriptions are aimed to be technology free, not suggesting any particular product or technology but rather specifying the required function to be satisfied. They define “what” must be done, not “how” it will be done. This approach allows the architecture to remain effective over time. The functions the system performs remain the same while technology evolves. This approach also serves to create a fair business environment by guaranteeing open market i.e. not forcing practitioners to use a specific equipment, brand or product in their projects.

Another advantage of defining system architectures before building them is to identify common elements, standards and processes throughout the system. This allows system builders to eliminate excessive investment in redundant parts of their systems and cluster similar functions to be performed by the same part or equipment.

Creating architectures for specific domains was proven to be successful given their ability to allow reuse strategies^[5].

“An Architecture Framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community.” (ISO-IEC-IEEE 42010)

There are architecture frameworks created in field of ITS. The purpose of this paper is to analyze and compare two of those architecture frameworks, the American Arc-IT and European FRAME, products developed through decades on both sides of Atlantic to support today’s connected and cooperative ITS.

Chapter 1.2 - The European ITS Framework Architecture (FRAME)

Increasing safety and efficiency of transport and traffic was long regarded as a common interest by the EU member countries even before the union was officially established. Significant efforts were made in funding the research and development in the area. European Commission's Third Framework Program for Research and Technological Development between 1990-1994 was concluded with emphasis "To contribute to the development of integrated trans-European services in the field of transport, using advanced IT and communications to improve the performance (safety and efficiency) of passenger and goods transport services, and at the same time reduce the impact of transport on the environment"^[6]. A Road Transport Telematics High Level Group was established by the EC in the Fourth FP^[7] to assist in the development of a long term strategy and an action plan for deployment of road transport telematics. It was this group's decision that a framework ITS architecture to be created to provide a methodical basis for ITS implementations throughout Europe, and the extensive work to create one was initiated when the European Council of Ministers approved this decision. Following the outcomes of these works EC funded the "Keystone Architecture Required For European Networks - KAREN" project, to create the European ITS Framework Architecture - "FRAME" which was first published in October 2000. The underlying aim of this initiative was to promote the deployment of (mainly road-based) ITS in Europe by producing a framework which would provide a systematic basis for planning ITS implementations, facilitate their integration when multiple systems were to be deployed and ensure interoperability, including across European borders^[8].

The European ITS Framework Architecture was created to support ITS deployment in all member countries of the European Union and also ITS deployment in other parts the world, if ever the project owners would like to benefit from such an architecture. To conform to the precepts of subsidiarity that was imposed by the general principles of European Union law and considering that the lifespan of transportation projects usually exceed the lifespan of technologies to be used^[9], the Frame Architecture and all of its components were built in a technology independent manner stating what is done by the systems to be deployed but not stating what mechanisms should be employed by the systems in order to do so. Keeping in mind that various countries are sometimes regulated by separate laws and operating in non-identical markets, it was aimed not to mandate any specifications regarding the products or systems that are to be developed or deployed; or any operational or organizational structure to its user. Instead FRAME was designed to be a "framework architecture", focusing on capturing features that its user would require from ITS and guiding its user to derive definitions of the functions that the systems will have to perform to fulfil these requirements. By making the standardized definitions of these requirements and the corresponding functions available to its user, the FRAME Architecture is produced to help its user as a tool to "convert" Stakeholder Concerns as it is defined in ISO-IEC-IEEE 42010, to descriptions of the functionality of the systems to be deployed to constitute a "Functional" view. These descriptions can be used in detailed design of the systems and their components or to produce specifications that would be included in the Calls for Tenders for development or procurement of the systems and their components.

There are two products of Frame Architecture, a list of User Needs which describes what is wanted from the ITS and a Functional Viewpoint of the systems that satisfy those needs, a comprehensive list of all functions that needs to be executed by the systems which would be developed or deployed to satisfy the User Needs.

To provide a systematic starting point to any ITS implementation, the KAREN Group listed all possible features that may ever be required from ITS after analyzing all existing ITS, through the outcomes of preceding EC projects, other national ITS architectures and from the knowledge derived out of the experience of the group members. The list of features then was narrowed down to define User Needs, a set of formalized statements written in English language that describe what features required from ITS. Below is an example of a User Need;

6.2.0.7. The system shall be able to know where it is in the transport network, and hence provide the position of the vehicle or person carrying it.

The User Needs are created to be as short and abstract as reasonably possible, clear in their meaning, consistent with each other and characterized by testable properties so that it would be possible to consider them as the smallest possible unit to describe a system, its functions, physical or organizational structure. They are coined to describe one feature per statement. User Needs are written from the systems' point of view, and not from the users', in order to enable the architecture to be built on a common ground that is shared by all of its users regardless of their purpose of using it. The User Needs are entry points to the FRAME Architecture since they are providing specifications for what functions will be performed by the ITS.

Having all the features that would be required by the stakeholders defined with the User Needs, KAREN Group then listed all functions that must be performed by ITS in order to fulfil those requirements and created a Functional Viewpoint describing the conceptual structure of the logical behaviour of the systems and defines what functionality should be provided by the systems that would fulfil the User Needs. Since most of the functions related to ITS require data collected by other parts of the ITS for their functionality and send out their data to be processed by other parts at the end of their processes, a series of Data Flow Diagrams structured in hierarchical levels was decided to be the best model to be used by the Functional Viewpoint to describe the Functional View^[10]. Data Flow Modelling is a typical "Model Kind" when it is necessary to show how the information flows through a sequence of processing steps in data intensive business processes^[11].

The highest level description of the Functional View of FRAME is the Context Diagram (see Figure 3) which depicts the ITS as a single item and shows its relation with the entities outside of the system boundaries that interacts with ITS through designed interfaces. These entities provide information to ITS to be processed or make use of the data produced by ITS. Since the sources of the input data or users of the output data of the systems would be located where the diagrams finish or terminate in a DFD representation of the, they are known as Terminators (see Figure 2).

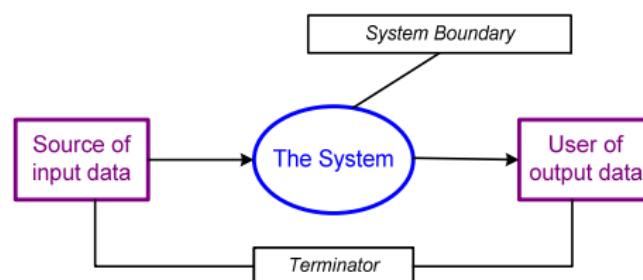


Figure 2 – Basic Context Diagram

Figure 2 - Conceptual Representation of Terminators (source: [12])

A Terminator can be humans (actors), other systems or various other entities from which data can be obtained such as the atmosphere, or road surface. A Terminator may refer to a Parent Terminator which contains a subset of terminators, each corresponding to variations of a specific type of terminator such as Driver: Public Transport Driver, Emergency Vehicle Driver, Private Driver or corresponding to different roles related to that type of terminator, Consignor/Consignee: Freight Shipper, Principal.

The interaction between the elements of the Functional Viewpoint is described with Data Flows. Each Data Flow is defined between a source and a destination Terminator and carries a specific set of data. Throughout the Functional Viewpoint, the Data Flows are also represented in a hierarchical structure. Parent Data Flows represents the communication between two elements in high level, where the Data Flows they contain are used by different Low Level Functions in the same Functional Area or High Level Function. In this level the communication between Terminators and the system is depicted only in the broadest terms, such as: “to Driver” or “from Driver”. Actually many different Low Level Functions interact with the Driver Terminator. In lower level representations of the Functional Viewpoint these data flows will be further detailed and Data Flow between Driver Terminator and each Low Level Function will be defined.

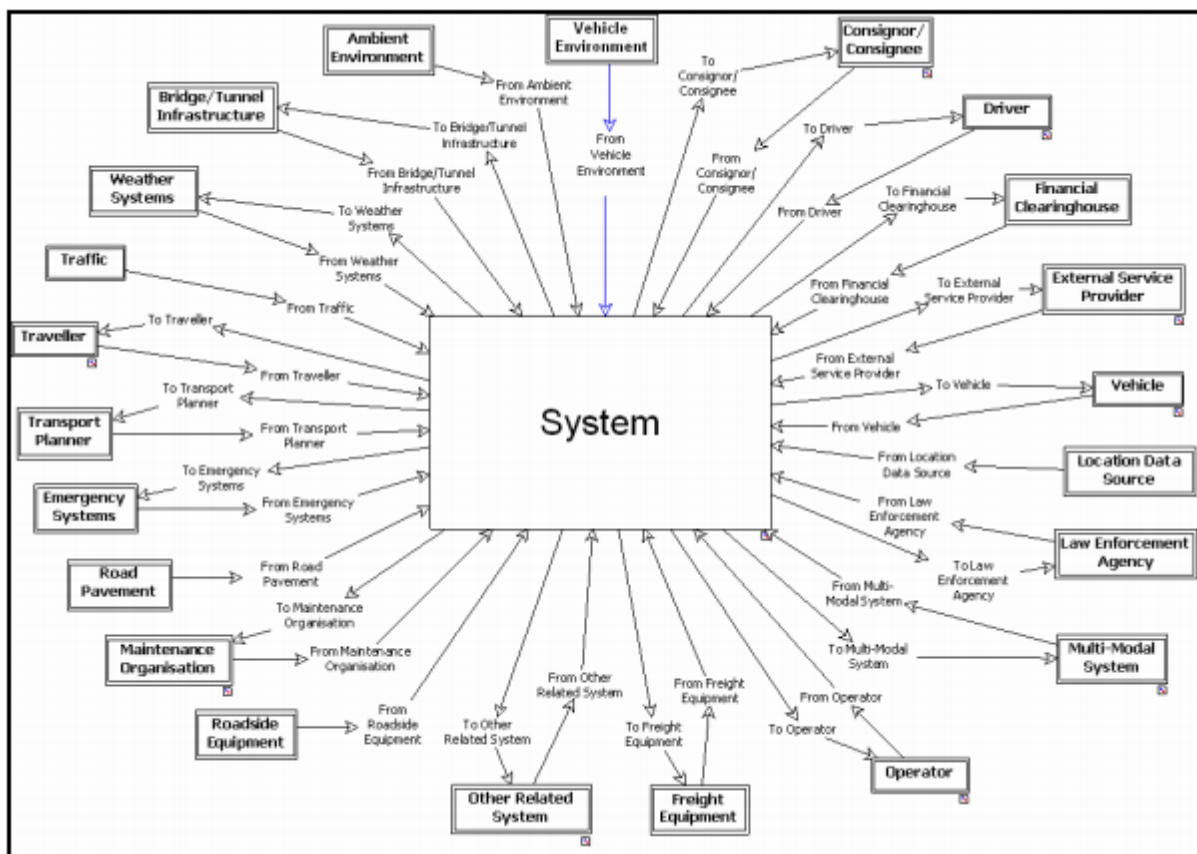


Figure 3 - FRAME Context Diagram (source: [3])

The highest level of functionality inside the system boundaries is shown with the Functional Area Diagram (DFD0) (see Figure 4). This diagram represents the entire scope of FRAME defined in 9 Functional Areas which provide the set of functionality specific to their areas of ITS. The diagram shows the interaction of all elements defined in FRAME with Data Flows either connecting two

Functional Areas or only entering to and leaving from one Functional Area, representing both the information exchange between Functional Areas and the interaction of the functions within those Functional Areas with the Terminators outside the systems' boundaries. This way of representing the interaction of the Terminators with the elements within the system boundaries is carried also to all lower level representations of the Functional Viewpoint since the Terminators are not represented in any of the diagrams inside the system boundaries.

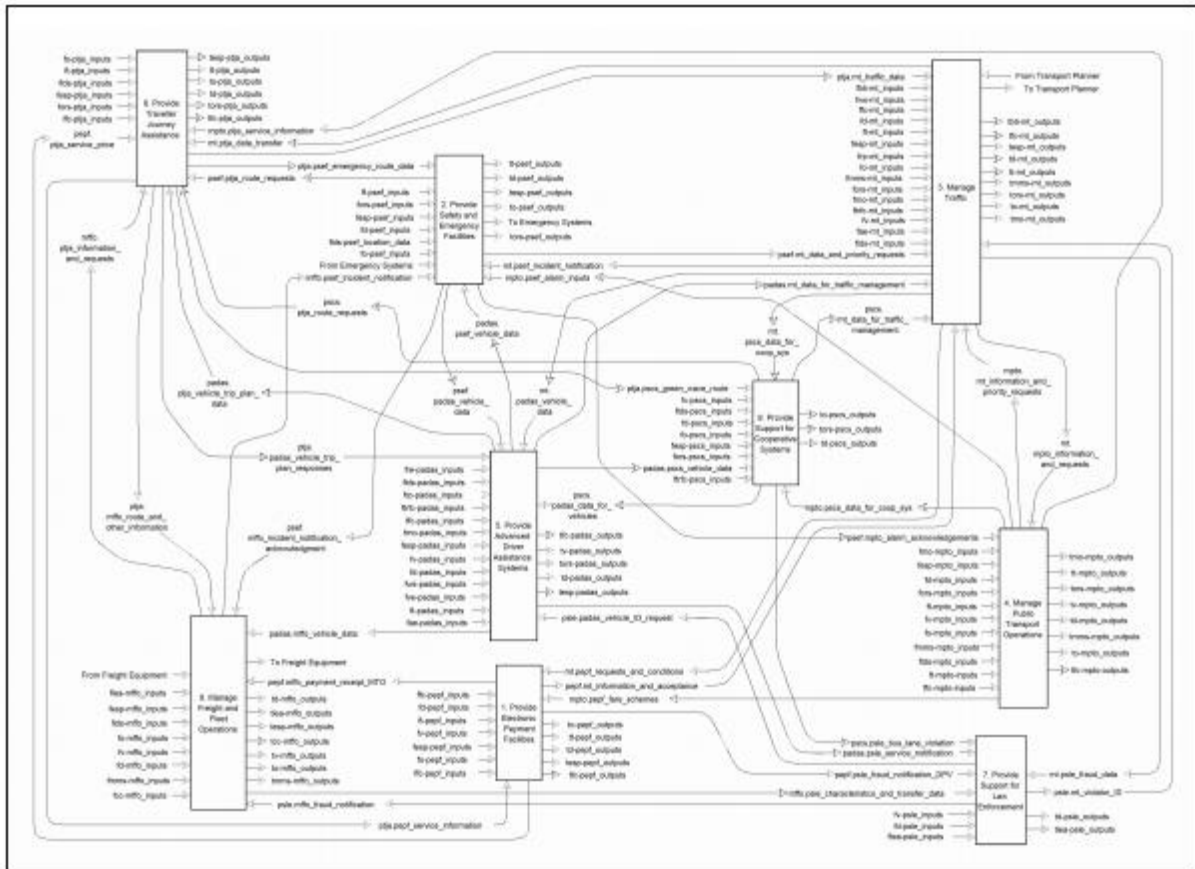


Figure 4 - The high level data flow diagram DFD0 (source: [3])

Each Functional Area contains the set of High Level Functions in their area of operations, and some of them may contain one or more Data Stores. Data Stores are elements of Functional Viewpoint that are used to store data that is used by two or more functions. Data Flow Diagrams are numbered according to the Functional Areas that they represent. DFD 7 below (see Figure 5) represents the Functional Area 7 - Provide Support for Law Enforcement. The interaction of High Level Functions between each other and Data Stores are shown with Data Flows connecting the elements. Interaction between High Level Functions and Terminators are also depicted, with Data Flows only going in to or out from the High Level Functions since Terminators is not represented inside system boundaries.

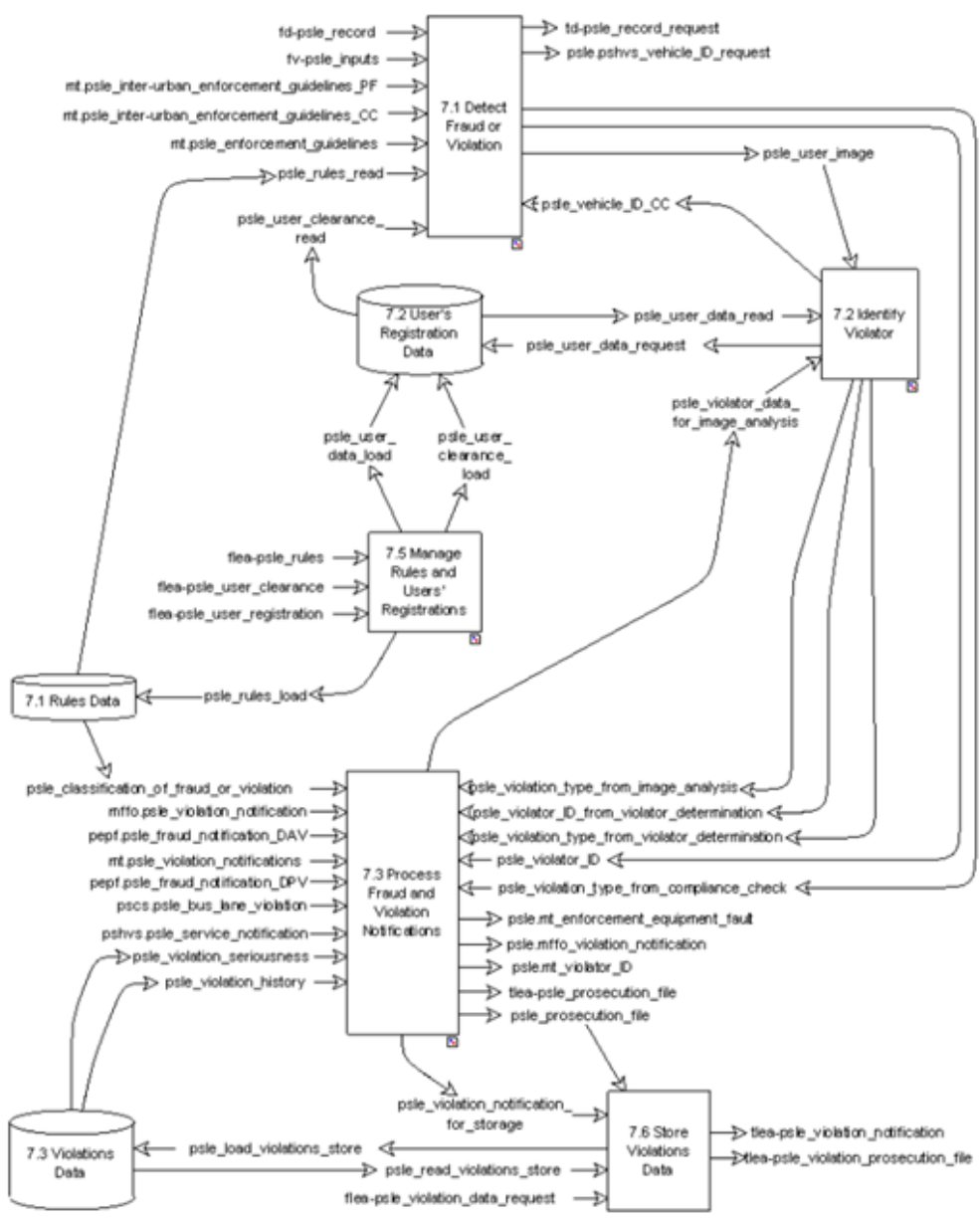


Figure 5 - DFD7 representing Functional Area 7 Provide Support for Law Enforcement (source: FRAME Browsing Tool)

Each High Level Function contains a set of Low Level Functions which perform the functionality and the Data Stores related to those lower level functions (see Figure 6). These Low Level Functions are the ultimate destinations or sources of the Data Flows pictured in the higher levels and they provide the FRAME its functionality. Each Low Level Functions specifies the facilities that the system(s) need to be able to provide and describes Functional Requirements to define what the system(s) are supposed to accomplish. Low Level Functions come together to deliver High Level Functions similar to the way that High Level Functions come together to deliver Functional Areas of ITS. Each High Level Function is described individually by data flow diagrams that are named and numbered with the High Level Function they represent. The interaction between the Low Level Functions and other elements is depicted in the similar manner that the higher level data flow diagram is organized, where the same level elements are connected via Data Flows and the connection to elements in higher levels are described with Data Flows that has one end free. In some cases throughout the Functional Viewpoint functions are organized in even lower levels

where some functions interact with low Level Functions of the High Level Function but contains even lower level functions themselves. The hierarchical organization is maintained in such cases, the Data Flow Diagrams represent only one level at a time and the interaction with elements of higher levels represented with one ended Data Flows.

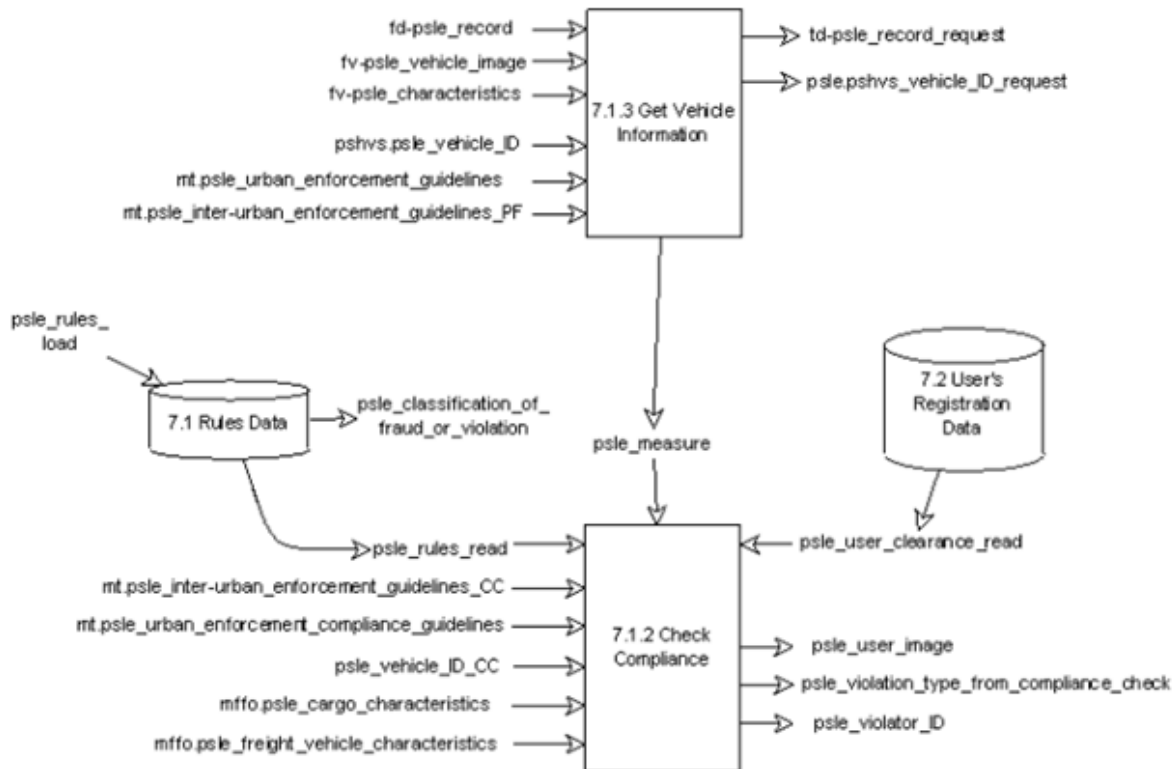


Figure 6 - High Level Function 7.1 Detect Fraud or Violation (source: FRAME Browsing Tool)

Systems described in the Functional Viewpoint of FRAME represent the systems that would fulfil all User Needs defined in the architecture. Each User Need is referenced to one or more Low Level Function that they require and thus the Functional Viewpoint is mapped to the User Needs. Low Level Functions in the FRAME Functional Viewpoint are defined by a description of facilities that the system(s) need to be able to provide, their Functional Requirements, the User Needs that they were referenced to and the Data Flows that uses them as the source or the destination that define the interaction of each Low Level Function with other elements of the Functional Viewpoint.

The FRAME continued to be developed over the years with further projects initiated to keep the architecture up to date, promote its use and support its users. The latest version extends the FRAME content to include results from various other EC funded projects on cooperative systems such as CVIS (Co-operative Vehicle Infrastructure Systems), COOPER (Co-operative Networks for Intelligent Road Safety) and SAFESPOT (Cooperative systems for road safety "Smart Vehicles on Smart Roads").

Covering each aspect of modern ITS, FRAME grew to be large in its size in years and it was found that it is a complex and exhausting task to go through and finally make use of the architecture. With the FRAME-S project (2001-2004) two computer-based tools were offered to assist architecture users. The Browsing Tool was created in form of HTML pages to be viewed using

Microsoft Internet Explorer and contains graphical representation of the Functional Viewpoint and the description of all of its Low Level Functions, Data Flows, Data Stores and Terminators along with the User Needs that the functions were referenced. The Browsing Tool provides an interactive interface through which the architecture could be navigated, enabling its user to move through its parts and follow through the relationships between its elements in various hierarchical levels and obtain the descriptions of each and every element within it.

A second tool, Selection Tool was created to support its user to select a consistent subset of the architecture and complete creating the Functional Viewpoint description of their system. Although it is not included in the boundaries of FRAME, The Selection Tool also supports the creation of subsystems and modules so that functions can be allocated according their physical locations and the system in subject can be defined by its Physical Viewpoint and the Organizational Viewpoint describing the roles and responsibilities of the stakeholders.

Using the FRAME architecture to create an architecture description is a process of creating subsets of the architecture by making selections out of it. Its ultimate product is a Functional Viewpoint, which displays all the Low Level Functions that were selected by its user.

First, the architecture user should capture the stakeholder aspirations. The “concerns” defined in ISO-IEC-IEEE 42010 is referred to as “stakeholder aspirations” in FRAME. Stakeholder aspirations are statements that answer the question “What does the people accumulated around the system, people who would own, use, operate, manufacture and supervise its use, expect the system to fulfil”. To find this out, the parties who decided to have the ITS should come together, debate and decide upon a comprehensive list of features and functionalities of the system that they desire.

The architecture user then should translate the Stakeholder Aspirations to User Needs by selecting the set of User Needs that would best describe the Stakeholder Aspirations. Since the User Needs are referenced to one or more function that compose the Functional Viewpoint, by mapping the stakeholder aspirations to the User Needs of the system in subject, the user of the Frame Architecture will be creating the Functional Viewpoint of primary functionalities of their desired system. User may continue developing their systems by adding functionalities beyond the primary functions offered with the selected User Needs.

The Functional Viewpoint created with FRAME may be used in further developing the systems, as a basis for creating Physical and Communication Viewpoints, designing and developing its components and parts, or comparing several alternative projects by cost and risk analysis and prepare yearly economic plans.

Chapter 1.3 - Architecture Reference for Cooperative and Intelligent Transportation (Arc-IT)

Architecture Reference for Cooperative and Intelligent Transportation (Arc-IT) provides a common framework for developing architectures for ITS. It was the first framework architecture developed in the field of ITS. It has been developed by United States Department of Transportation (US DOT) to support and manage transportation projects in 1996^[13] as a response to opportunities emerged due to advancements in the information processing and communication technologies. It was titled US National ITS Architecture when it was first published. US congress passed the Transportation Equity Act for the 21st Century (TEA 21) in 1997^[14], which requires to develop, implement and maintain a national architecture to promote the widespread use and evaluation of intelligent transportation system technology as a component of the surface transportation systems of the United States. In January 2001 Federal Highway Administration published the Final Rule on Architecture and Standards Conformity^[15] mandating that all ITS projects funded from the Highway Trust Fund be in conformance with the National ITS Architecture^[16]. By mandating a reference architecture US DOT aims a country wide integrated ITS.

National ITS Architecture was created to guide ITS planning and investment at both state and local level. By identifying interconnections and interdependencies between systems and its stakeholders, the common standards applicable for the systems the National Architecture aims to develop a blueprint for integration of the systems. The National ITS architecture was updated many times over the years to reflect the changes in technology and to make the architecture compatible with current advancements in ITS. The latest revision is a result of merging with Connected Vehicle Reference Implementation Architecture content to enable the architecture to include services supported by Connected Vehicle capabilities and to expand the stakeholder concerns addressed by the architecture. This since merger resulted with major changes in the content and the architecture structure, it was decided to change its name to Arc-IT as it is known today.

Following ISO/IEC/IEEE 42010, the architecture description of Arc-IT was organized into views each describe the system(s) from different perspectives. There were four views identified for Arc-IT, Enterprise, Functional, Physical, and Communication. Offering its user architecture descriptions of various views, Arc-IT serves beyond defining the functionality of the systems but portrays the full environment where the stakeholder concerns are satisfied.

Arc-IT was divided into Service Packages that describe architecture for a portion of ITS. With each Service Package a single service is described. The Service Packages includes definitions of all four views to describe the service that ITS provide, the concerns that are considered and how they are satisfied. In Arc-IT a wide array of ITS is described by a set of total 139 Service Packages that are distributed into 12 main service areas (see Figure 7). The Service Packages are entry points to Arc-IT. Architecture user is expected to select the set of Service Packages that would best describe the system(s) that would fulfil their needs. This paper will be differentiating two major architectures, the National ITS Architecture is the entirety of all Service Packages and the Regional Architecture is the architecture that the architecture user creates for their projects by making their selections out of the National ITS Architecture.

| | | |
|---------------|------|--|
| Public Safety | PS01 | Emergency Call-Taking and Dispatch |
| | PS02 | Routing Support for Emergency Responders |
| | PS03 | Emergency Vehicle Preemption |
| | PS04 | Mayday Notification |
| | PS05 | Vehicle Emergency Response |
| | PS06 | Incident Scene Pre-Arrival Staging Guidance for Emergency Responders |
| | PS07 | Incident Scene Safety Monitoring |
| | PS08 | Roadway Service Patrols |

Figure 7 - The Public Safety Service Area and some of its Service Packages (source: Arc-IT website)

Physical View elements that make up the architecture are depicted as Physical Objects and the interrelation of elements are visualized with Information Flows that connects each other. There are two types of Physical Objects: Subsystems and Terminators. Subsystems are parts of ITS which perform transportation related tasks and thus provide the ITS its functionality. They are assigned with Functional Objects, components of Physical View that clusters similar or complementary functions together. The Functional Objects are mapped to functional requirements to describe what is done by the Physical Objects. Terminators were not defined with any functionality nor were assigned with any Functional Objects since they represent the agents outside of the system boundary. They are either “sources” that provide the necessary information to or “sinks” that consume the information produced by the systems. Typically they are human operators or other systems but sometimes the ITS itself can be a Terminator for some Service Packages (E.g. Other Emergency Management Centres). Both types Subsystems and Terminators exchange information in order to provide ITS services thus are connected with other elements with Information Flows. The Physical Viewpoint in Arc-IT is defined in 5 classes and was color coded throughout the architecture to make it easier to analyze the diagrams.

- Centres, such as a Traffic Management Centre are in color cyan
- Field Equipments, such as a Traffic Signal Controller are in color orange
- Vehicles, including specialized vehicles such as Transit Vehicles are in color blue
- Traveller Devices, such as personal devices (smartphones) are in color yellow
- Support Systems; this class includes systems that provide non- operational use of ITS data (e.g. Archive Data Systems), and support to a variety of services (e.g. Map Update System). They are coded in color khaki green in the physical view.

Below is the highest representation of Arc-IT Physical View. All 40 subsystems defined in all 5 classes of ITS and the general communication links used to exchange information between these subsystems are depicted. It excludes anything outside the system boundaries, so the interaction with terminators and the environment of the system is not pictured. When it is observed at this level, the communication between subsystems is depicted in the broadest terms. Centres and Support Systems are using Wide Area Wireless Networks or Broadcasting when communicating with Vehicles and Traveller Devices whereas Field Equipment communicates with them using Short Range Wireless Communication. The Centres and Support Systems link to Field Equipment via centre to field communication channels. Centres and Support Systems employ centre to centre communication channels to communicate with each other. Field equipment interconnection is provided via field to field communication channels. Finally the vehicles establish communication through Short Range Wireless between themselves and the Traveller Devices. The type of communication channels required between different classes of Physical Objects lay the foundations of the Communication View. A similar diagram will be offered to architecture user as a product when they finish creating the Regional Architecture showing the Subsystems selected for the architecture and the interaction between them.

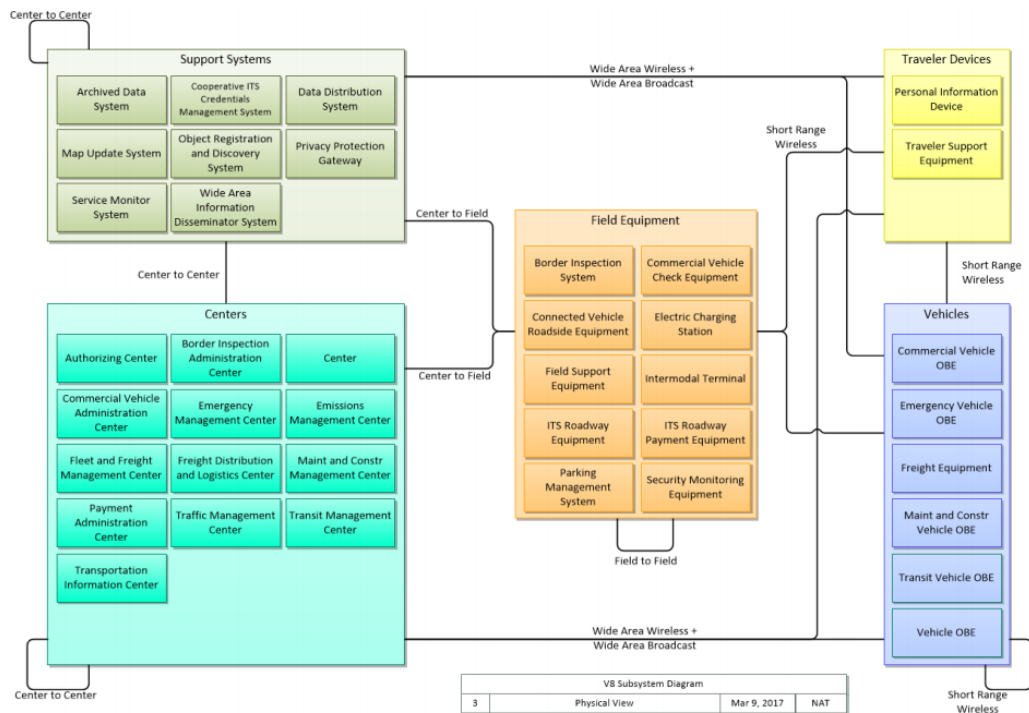


Figure 8 - The Arc-IT Subsystem Diagram (source: [13])

A Service Package Diagram that depicts the Physical View of the Mayday Notification Service Package is given as an example below (see Figure 9). There are two main interfaces defined with the Service Package, between the Emergency Management Centre and the Vehicle OBE and Personal Information Device. The green and red arrows are representing the Information Flow between these Subsystems. There are six more interfaces defined with the Service Package to represent the interaction of the system with its terminators in addition to the main interfaces. The black arrows are representing the Information Flow between the system and the Terminators. Information Flows will be used to define the Triples which are the basic element of the Communication View.

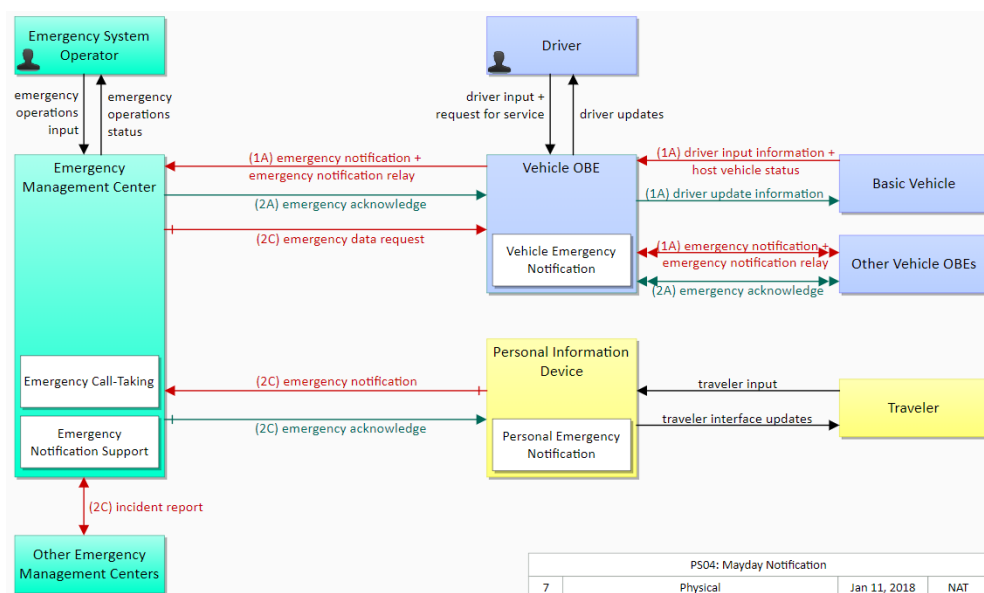


Figure 9 - The Mayday Notification Service Package Diagram (source: Arc-IT website)

Functional Objects are components of Physical View and intermediary to the Functional View in Arc-IT. The Functional Objects, represented by the white boxes above, are allocated to Physical Objects. Each Functional Object is created to respond a set of functional requirements which are originated to satisfy sets of needs (see Figure 10). In each Service Package the needs are listed and matched to the functional requirements and to the Functional Objects where the need is satisfied (or the process that satisfies the requirement is performed). Requirements analysis was done while setting up the architecture and requirements have already been associated with Functional Objects, which is stated as the reason why ARC-IT users are encouraged to jump start from the Physical View of the architecture. Functional requirements identify the processes and information exchanges necessary to satisfy needs. The functional requirements are traced to the needs defined for each Service Package. Since a Functional Object can be included in many Service Packages, the functional requirements are developed to cover all needs across all of the Service Packages of the architecture and only some portions of the requirements are matched with a single Service Package.

| Need | Functional Object | Requirement |
|--|--------------------------------|---|
| 01 Emergency Management needs to be able to determine that a crash or emergency situation has taken place, based on on-board sensor data that detect changes in velocity, vehicle orientation, and airbag status. | Emergency Notification Support | 01 The center shall be able to determine that a crash or emergency situation has taken place, based on on-board sensor data collected from the vehicle. |
| | | 02 The center shall monitor subscribed vehicle data, including changes in velocity, attitude/orientation, position, and air bag status to determine when an emergency situation (crash) has happened. |
| | | 10 The center shall request additional emergency details from or issue commands to the vehicle's security systems or vehicle driver if needed. |
| | | 11 The center shall maintain a log of all mayday signals received from vehicles. |
| | | 13 The center shall determine that a collision has occurred based on changes in vehicle sensor data. |

Figure 10 - Needs defined for the Mayday Notification Service Package and the Functional Requirements that satisfy them (source: Arc-IT website)

The functionality of the systems is further described as a set of processes which are called as P-specs in the Functional View (see Figure 11). Functional requirements are met by the P-specs. The processes in Arc-IT exchange information through data flows. Capabilities that are provided by each P-spec were described in detail and all Data Flows associated with each of them are listed. A process may be associated with more than one Functional Object in the Physical View. Data Flows define the specifications and the scope of the information that needs to be exchanged between processes. The Information Flows in the Physical View are defined by the Data Flows. Data Flows are organized in a hierarchical order where some Data Flows between P-specs may be parenting some sub Data Flows.

| Physical Object | Functional Object | PSpec Number | PSpec Name |
|-----------------------------|---------------------------------|--------------|---|
| Emergency Management Center | Emergency Call-Taking | 5.1.1.1 | Coordinate Emergency Inputs |
| | | 5.1.1.3 | Collect Incident And Event Data |
| | | 5.1.2 | Determine Coordinated Response Plan |
| | | 5.1.3 | Communicate Emergency Status |
| | | 5.1.4 | Manage Emergency Response |
| | Emergency Notification Support | 5.2 | Provide Operator Interface for Emergency Data |
| | | 5.1.2 | Determine Coordinated Response Plan |
| | | 5.1.3 | Communicate Emergency Status |
| | | 5.1.6 | Process Mayday Messages |
| | | 5.2 | Provide Operator Interface for Emergency Data |
| Personal Information Device | Personal Emergency Notification | 6.8.1.5 | Provide Traveler Emergency Message Interface |
| | | 6.8.2.1 | Build Traveler Personal Security Message |

Figure 11 - The Functional View description of Mayday Notification Service Package (source: Arc-IT website)

In the Enterprise View the stakeholders and their involvement to the system throughout the systems life cycle is described. The building blocks of Arc-IT's Enterprise View are Enterprise Objects, who are the organizations or the individuals which are identified as stakeholders of the system. In the Enterprise View relationships between these Enterprise Objects and their roles in building, operating and maintaining the system are described. The interaction between stakeholders, the services shared among each other and the information exchange are defined with the agreements in the architecture. The needs that serve as the rationale for the services described by the Service Packages are elements of the Enterprise View. Each need is matched with the functions that they require from the system while building the Service Packages and thus the Enterprise View is mapped into Functional Viewpoint. Each Enterprise Object has a kind of a relationship with the Physical Objects, which are defined as Roles in architecture, such as owns, operates, maintains etc. and some Responsibilities for them (see Figure 12). The Enterprise Viewpoint is mapped to the Physical Viewpoint via the Roles.

| Source | Destination | Role/Relationship |
|--------------------------|------------------------|-------------------------------|
| Basic Vehicle Maintainer | Basic Vehicle | <u>Maintains</u> |
| Basic Vehicle Manager | Basic Vehicle | <u>Manages</u> |
| Basic Vehicle Manager | Basic Vehicle Operator | <u>System Usage Agreement</u> |
| Basic Vehicle Manager | Driver | <u>System Usage Agreement</u> |
| Basic Vehicle Operator | Basic Vehicle | <u>Operates</u> |
| Basic Vehicle Owner | Basic Vehicle | <u>Owns</u> |

Figure 12 - Some of the Roles defined in Operations Stage of The Enterprise View of Mayday Notification Service Package (source: Arc-IT website)

The Communication Viewpoint describes the interaction between the elements and the processes and defines the standards and protocols necessary to provide interoperability between them. The information exchange between the elements of Functional View is provided with Data Flows (see Figure 13). The Information Flows in the Physical View carries the Data Flows. The Information Flows in Arc-IT are called as Triples. Triples define the Physical Object which is the source of the information, the Data Flow that is carried with the Information Flow and the destination Physical Object. Each Triple is described as a pair of communications stacks that lists the protocols employed in each layer of the communication. The primary element that make up the view are the communications diagrams (see Figure 14) that depict these communication stacks identifying the protocols regarding each layer of communication for each Information Flow.



| Source | Flow | Destination | Communications Diagram(s) |
|--|---------------------------------------|--|---|
| Alerting and Advisory System | alerts and advisories | Commercial Vehicle Administration Center |  |
| Alerting and Advisory System | alerts and advisories | Emergency Management Center |  |

Figure 13 - Some of the Triples defined in the Communication View of Arc-IT (source: Arc-IT website)

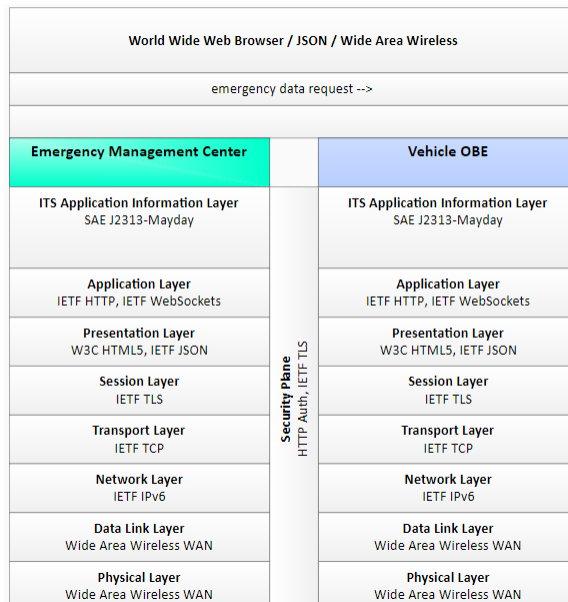


Figure 14 - The Communications Diagram of Emergency Management Centre - Vehicle OBE: Emergency Data Request Triplet from the Mayday Notifications Service Package (source: Arc-IT website)

Arc-IT content is reached through its web site. User is expected to study the Service Packages, the views that are describing the services and all other artefacts defined for the Service Packages from the architecture web site. The Arc-IT web site is developed in a tabular organization that the user can navigate through the Physical, Functional and Enterprise Views of the Service Packages, the Goals and Objectives they aim to satisfy, the Needs and Requirements that are defined for them, the Sources they are referenced to and the Security levels defined for the Service Packages.

Users are first directed explore the Physical Tab in a Service Package that contains the Physical View. It is considered to be the most natural way to start articulating about systems since the Physical View portrays the parts of the system that the user can touch, see and interact with. The Physical View consists of a Service Package Diagram that depicts the Physical Objects and the Information Flows between them and detailed descriptions of each Physical Object, Functional Object and Information Flow included in the Service Package.

Functional Tab contains the Functional View and consists of the Functional Objects allocated to Physical Objects included in the Service Package and the processes that the Functional Objects required to be performed in order to fulfil the services defined with the Service Package. Each process may be assigned to more than one Functional Object. User may find a detailed description of each process and the list of all Data Flows defined for the process.

The Enterprise Tab is divided into four stages of the system development life cycle: Development Stage, Installation Stage, Operations Stage and the Maintenance Stage. The roles and relationships of Stakeholders are outlined with roles or agreements defined in the architecture. The stakeholders were defined in a generic manner in the National ITS Architecture such as “Emergency Management Centre Manager”, “Vehicle OBE Supplier” or “Personal Information Device Maintainer” and the architecture user is expected to define who are those generic stakeholders correspond to for their Regional Architectures. All roles / relations are defined with a

stakeholder as the source, a Physical Object as the destination and a role such as Owns, Installs, Develops, Maintains etc. or a relationship such as The Installation Agreement, Application Interface Specification, Expectation of Data Provision or System Usage Agreement. These roles and agreements are already defined with the Arc-IT. The architecture user may define their own agreements or role and relationships according to their needs for their Regional Architecture.

In the Goals and Objectives Tab the architecture user should find the planning factors and goals, objective categories and the objectives and the performance measures for the Service Packages. The decision to invest on ITS are made to achieve desired outcomes with the project. A set of eight Planning Factors derived from 23 CFR part 450 - Planning assistance and standards for the Federal Highway Administration^[17] and a Goal was defined for each Planning Factor in Arc-IT. Each Goal than further was associated with some Objectives in few Objective Categories such as Emergency Management, Safety, Transit Operations Management etc. The Objectives in Arc-IT are provided with several Performance measures to assess compliance with the Objective such as "Per capita time to evacuate" or "Mean incident clearance time per incident". User is expected to either user existing Goals, Objectives and the Performance Measures from the architecture web site or to create their own according to the transportation plans for their projects.

The Needs and Requirements Tab contains the "stakeholder concerns" and the Functional Requirements. The Needs were defined for each Service Package in the creation of the architecture and Functional Requirements of the Functional Objects were matched with the Needs they aim to satisfy. By choosing the Service Package for their Regional Architectures the architecture user will be automatically including the set of Needs defined with the Service Packages they have selected.

There may be National or Regional Transport Development Plans, Traffic Management Policies, Government Incentives and Programs that are enforcing or supporting the ITS to be built or developed and that may be why the user is creating an architecture. The Source Tab is containing information of such sources for an ITS Architecture. The sources in Arc-IT are referencing such documents for US. Architecture user may insert their own sources while creating their own Regional Architectures.

Finally A set of security levels were defined for each Physical Object and Information Flow that was defined in Arc-IT. The systems that the architecture user will be creating with the architecture should at least meet or exceed the security levels that are assigned to each element of the Physical View of the Service Package that is under the Security Tab in Arc-IT website.

User is expected to select the Service Packages that would best describe the totality of the services that they are aspiring to obtain with their ITS investment and create a Regional Architecture. Then these architectures are used to create deployment sized projects and continue with next steps of systems engineering i.e. start defining product specifications etc.

There are two computer based products developed to enable architecture user to create the architectures for their projects. The Regional Architecture Development tool for Intelligent Transportation (Rad-IT) is where user creates their Regional Architecture which is a sub selection of the National ITS Architecture. The System Engineering Tool for Intelligent Transportation (Set-IT) is where they develop detailed architectures for the sub projects included in a Regional Architecture. User is able to define each element that makes up the Physical View with visual

charts and complete developing the Communications and the Enterprise Views of their projects with Set-IT. While the Set-It will not be covered with this paper, creating a Regional Architecture for an example Service Package will be detailed in Chapter 6.

Chapter 2 - Methodology

This paper analyses architecture frameworks created in field of Intelligent Transportation and tries to contrast the European ITS architecture (FRAME) with the American ITS architecture (Arc-IT). Ways to transfer the functionality from FRAME to another tool that would support the architectures to be represented better will be discussed.

In parts of the first chapter the concept of system architecting and architecture descriptions was introduced summarising the ISO-IEC-IEEE 42010 on System and Software Engineering - Architecture Descriptions. The architecture descriptions was proclaimed as the product of system architecting efforts and architecture frameworks were presented to be guidelines and best practicing while creating architecture descriptions in specific domains. Than the content of the European and the American ITS architectures was described in detail to study their methodologies in creating architecture descriptions, the conventions and principles they imply was discussed and the tools they provide were introduced.

In the practical analysis part, the paper will continue with introducing a small part of present day ITS service as an example, namely the eCall, to initiate discussion of architecture description methodologies employed by both FRAME and Arc-IT. A similar system's architecture description will be created both in the Selection Tool of FRAME and Rad-IT of Arc-IT to be able to present the architecture creation steps of both, in detail.

The fourth chapter introduces Enterprise Architect, a modern day architecture modelling tool that has widespread usage in many industries. The architecture description for eCall in FRAME Selection Tool will be modelled in this tool to add the visual representation that was missing in FRAME's architecture description. To use Enterprise Architect instead of the current Selection Tool will be proposed and a method to transfer FRAME content and methodology to Enterprise Architect as a Model Library will be developed. Lastly the Mayday Notification Service Package, the ITS service corresponds to eCall will also be modelled with Enterprise Architect. Since Arc-IT already allows means of visually representing the resulting Regional Architectures, transferring to another tool will not be proposed. The Mayday Notification Service Package will be modelled in Enterprise Architect only to be able to contrast the end product of both framework architectures under same conditions, to eliminate any differences that would arise because of using different tools in the comparison. In other words the architecture descriptions of both will be modelled in a third tool to not to be comparing apples with oranges. To further enable this, only the Functional Viewpoint of FRAME and the Functional View of Arc-IT will be modelled in the new tool and compared. Building other views on top of the functional will be discussed.

The paper will be concluded by pointing out the best practices in both FRAME and Arc-IT and by elaborating on the future work that might be needed to further develop the European ITS Architecture.

Chapter 3 – The Practical Analysis

Chapter 3.1 - The eCall Example

eCall is an in-vehicle emergency system that places an automated call to 112 in case of a severe crash or road accident. 112 is the single free of charge emergency number for all European Union member countries (in this case including also Iceland, Norway and Switzerland)^[18] and enables access to emergency services EU wide regardless the country of origin of the host vehicle. The system aims to reduce road accident related fatalities by reducing the emergency response time after a crash occurs, by establishing a voice call to nearest emergency response centre and communicating the precise vehicle location and other vital information about the incident (see Figure 15).

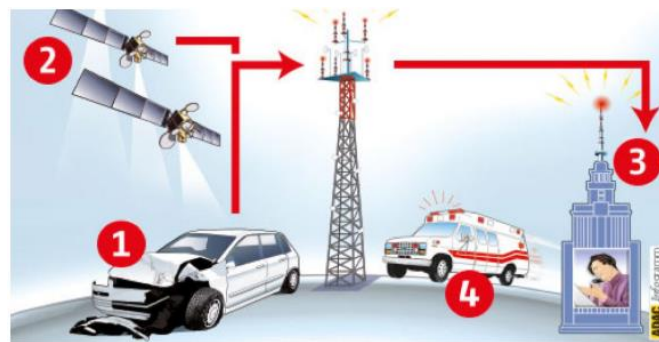


Figure 15 - A Simple Description of eCall (source: [20])

eCall first gained publicity in late 1990s^[19] as a potential civilian safety application for Galileo (The European Global Navigation Satellite System) and became mandatory for all new passenger cars and light duty vehicles (vehicle types M1 and N1) being sold in European Union after April 2018^[20]. While it is not mandatory for other regions in the world, similar services are being offered on subscription basis by third party service providers^[21] or as standard or optional features by various car manufacturers^[22].

The eCall can both be triggered by the driver by pressing a button present in the vehicle or automatically by the vehicle sensors in detection of a collision. When activated, the in-vehicle eCall device establishes a voice call to the nearest Public Safety Answering Point and at the same time transmits a mandated Minimum Set of Data (MSD) about the incident. The MSD was standardized with EN 15722 and contains message id, vehicle id (VIN), vehicle propulsion type, time stamp, vehicle location, direction of travel, type of the activation (manual or triggered), call type (emergency or test) and the position confidence (the position confidence would be low if the position is not within the limits of ± 150 m with 95% confidence, otherwise would be trusted)^[23]. Additional critical data to mandatory MSD (Such as number of closed seat belts) may be transmitted depending on the vehicle configuration. The eCall is monitoring the vehicle SRS (Supplemental Restraint System) such as such as airbag systems, air curtains, and seatbelt pretensioners, to detect collisions^[24].

eCall requires the calls to be placed by the on-board eCall module, recognized and processed by the mobile network and routed to specialized PSAPs where trained operators would coordinate the

necessary response and maintain contact with the occupants and the Traffic Management Centre until the incident is closed.

The eCall system was provided as an example to illustrate the usage of FRAME in deployment of ITS services with the FRAME NEXT project due to simplicity and small size^[25]. We will also be following that example in our research, creating the eCall architecture as it described in the example and compare the architecture creation methodology of FRAME and Arc-IT by creating architecture for a similar Service Package in Arc-IT.

We found that it is possible to cover the eCall capabilities with the Mayday Notification Service Package in Arc-IT, only with slight differences. The Arc-IT defines the Mayday Notification services to be accessible by traveller devices in addition to the vehicle on-board module. This is not the case for eCall where the notifications from travellers outside any vehicle are excluded. eCall users are able to report accidents that they are not involved using the panic button within their vehicles but travellers outside the vehicles are required to call e112 from their personal devices, which is another service maintained by EU initiative outside ITS scope.

To ensure that the service described in the eCall example is fully covered by the Mayday Notification Service Package we compared the functions provided by them and created the relationship matrix diagram below to demonstrate how they are related (see Figure 16). As described above, the only difference is found to be the functions related to the services provided to travellers outside the vehicle by the Mayday Notification Service Package.

| FRAME / Arc-IT | 5.1.1.1 | 5.1.1.3 | 5.1.2 | 5.1.3 | 5.1.4 | 5.2 | 5.1.6 | 6.8.1.5 | 6.8.2.1 | 6.8.2.2 | 3.1.3 | 3.3.1 | 3.3.2 | 6.7.1.2 | 6.7.2.1 | 6.7.2.2 |
|----------------|---------|---------|-------|-------|-------|-----|-------|---------|---------|---------|-------|-------|-------|---------|---------|---------|
| 5.11.7 | | | | | | | | | | | | | | | | |
| 5.12.7 | | | | | | | | | | | | | | | | |
| 2.1.2.1 | | | | | | | | | | | | | | | | |
| 2.1.2.3 | | | | | | | | | | | | | | | | |
| 2.1.2.4 | | | | | | | | | | | | | | | | |
| 2.1.2.5 | | | | | | | | | | | | | | | | |
| 2.1.5 | | | | | | | | | | | | | | | | |
| 2.1.9 | | | | | | | | | | | | | | | | |

Figure 16. eCall - Mayday Notification Service Package Relationship Matrix (source: author)

We will be using the eCall and Mayday Notification Service Package examples throughout the paper, to study how FRAME and Arc-IT define architectures for these services and demonstrate the similarities and differences between the two architecture frameworks.

Chapter 3.2 - Creating an Architecture Description of The eCall Example in FRAME

FRAME Selection Tool is a computer based product that was created to provide the architecture user with the ability to make selections to create the Functional Viewpoint description of their ITS projects. Once user created a Functional Viewpoint for their desired system(s) they can continue describing Physical and Organizational Viewpoints for their architectures. The User Needs along with the Viewpoints the user created make up the architecture for the particular system(s).

The architecture creation process is done in two phases, the first pass and the subsequent passes. In the first pass the architecture user will be selecting the User Needs that would best describe the system(s) of their interest and creating the part of the Functional Viewpoint that is directly related to those User Needs. In the subsequent passes the user may add or remove functionality or elements such as Data Stores or Terminators to fully describe the Functional Viewpoint of the system(s) that they have in mind.

The Selection Tool offers a simple interface that repeats itself in each step of architecture creation. The screen is divided into three main parts, the left half on the screen is where the available items are listed, and the selected items are moved to the right half of the screen by “Add” button. On the bottom the description and other information related to selected items are displayed. The user is expected to confirm their selection by hitting “ok” to proceed to the next step in architecture creation.

The architecture starts by offering the full list of User Needs to the architecture user (see Figure 17). The user may find the User Needs listed in a hierarchical order, divided into groups clustering the User Needs in similar areas of ITS. User is expected to go through the User Needs and make a selection of the ones that would best describe the system(s) of interest.

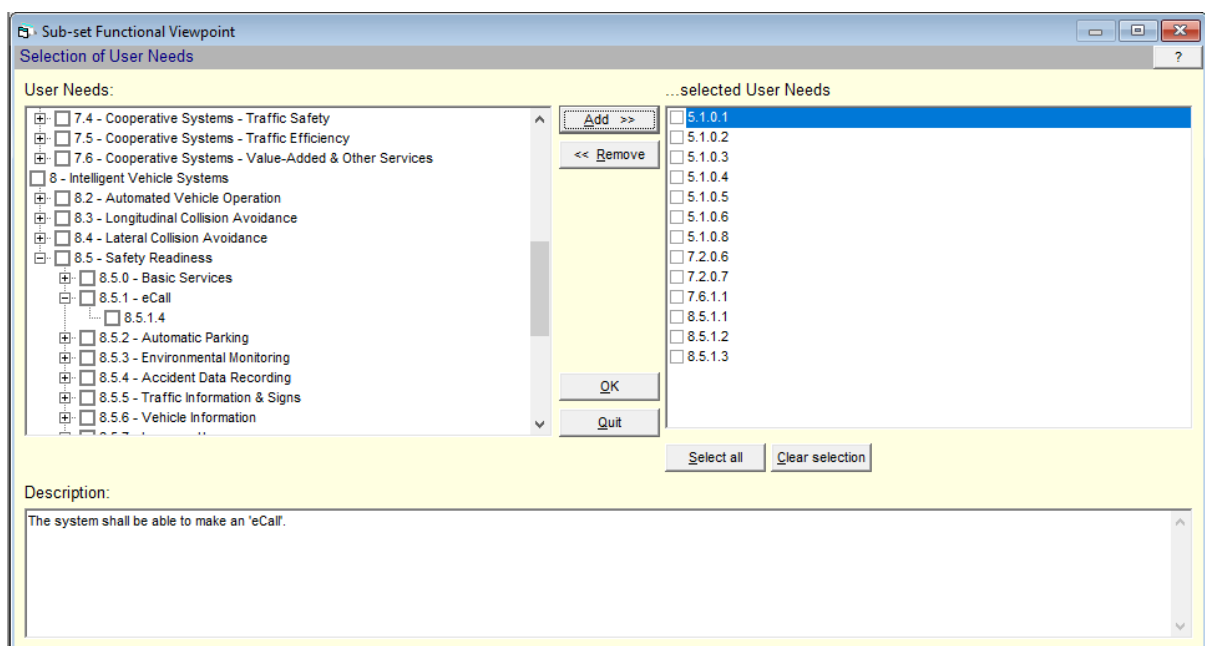


Figure 17 - Selecting User Needs (source: author)

In the next step the architecture user is offered the list of all Low Level Functions that are related to the User Needs that were selected in the first step (see Figure 18). Each function that composes the Functional Viewpoint is cross referenced to one or more User Needs (most commonly a Functions is mapped to many User Needs) so by choosing the necessary User Need the user is actually selecting a set of Functions that will be used to build up the Functional Viewpoint of the desired system's architecture.

As discussed earlier, the User Needs are used as an entry point for the Functional Viewpoint. The architecture user is expected to go through the Browsing Tool to understand what are implemented with the offered functions, how they are organized and how they will be interacting with each other. The entirety of the Functional Viewpoint is described via Data Flow Diagrams in the Browsing Tool with all of its low Level Functions, Data Stores and Terminators of the system(s) along with all Data Flows between these elements. The user is expected to identify and study the parts of the Functional Viewpoint that they would be including to their architecture so that they can make necessary selections in the next steps of the Selection Tool.

The user is expected to create a subset from the list of all functions that are offered by the selected User Needs and decide the primary functionality of their desired system(s). The selected functions in this step will determine which data flows will be offered in the next step. Functions added to the architecture in this step will be called as the primary functions throughout the rest of the paper.

The functions related to the selected User Needs are presented to user in this step with their descriptions and the User Needs related to them. The User Needs that were selected in the previous step are highlighted. User may go back to the previous step and change their selection of the User Needs if necessary. We continue this step by selecting the Low Level Functions that was mentioned in the eCall Example.

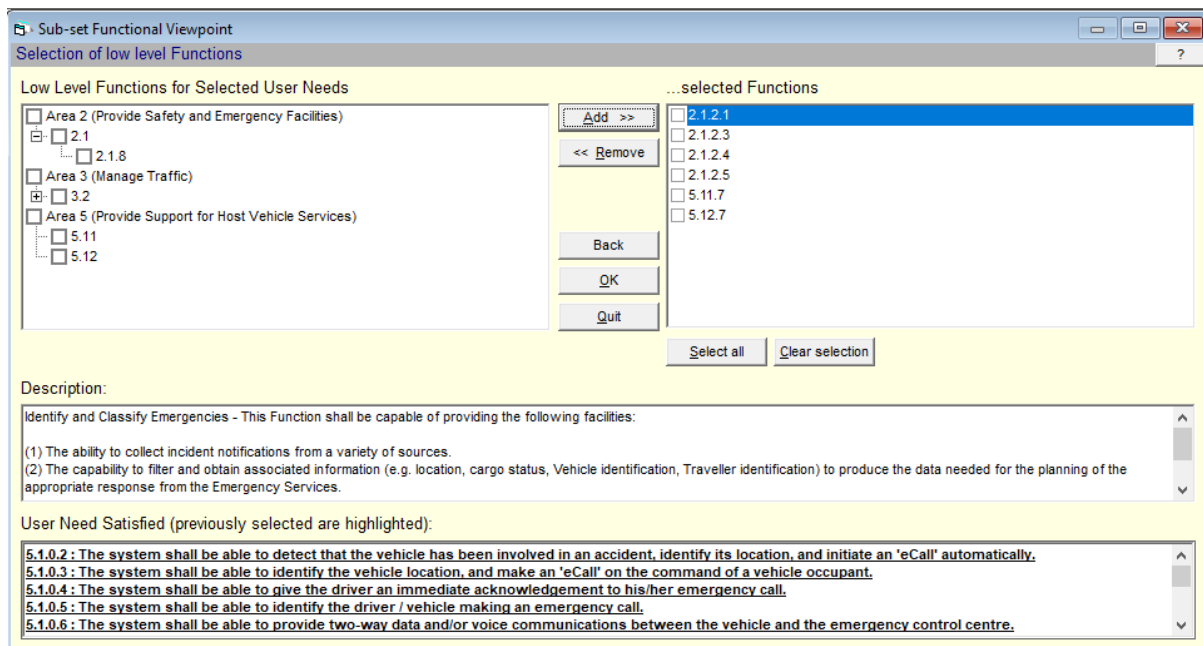


Figure 18 - Selecting Low Level Functions (source: author)

The functions 2.1.5 - Provide Access and Maintain Data for Emergency and 2.1.9 - Provide Emergency Interface that were described with the system in the eCall example was not offered

here since no User Need that would require those functions were selected in the beginning. This doesn't necessarily mean that those functions will not be included in the Architecture. Users still may add from the rest of the Functions that were not offered initially in the subsequent passes.

Upon selecting the primary functions that would be included in the architecture, the user is offered a list of all data flows that starts and ends with the primary functions and required to define how the primary functions interact with each other and also with the other elements of the Functional Viewpoint such as Data Stores and Terminators. By now the architecture user is expected to have a broad understanding of the Functional Viewpoint as it represented in the Browsing Tool since the Data Stores and Terminators that will be offered as available on the next steps will be based on the Data Flows that are selected in this step. If the user does not select any Data Flow that is ending or beginning with a Data Store or a Terminator, there will be no Data Stores or Terminators offered in the following steps. While it is possible to add those elements to the architecture in the subsequent passes, the user is expected to choose all the Data Flows that are describing the system(s) they have in mind in this step, for a meaningful first pass. The architecture user may also add the Data Flows between the primary functions and the functions that will be included in the architecture in the subsequent passes already in this step as long as they are confident that the function in the other end of the Data Flow will be required for their system(s). Selecting Data Flows related to elements that are not yet included in the architecture will result in errors at the end of the first pass. User may add the missing elements in the subsequent passes or they can go back to Data Flow selection screen and remove the unnecessary Data Flows to correct the error(s).

The Data Flows in FRAME Functional Viewpoint is defined specifically between certain elements, connecting a specified source element to a specified destination element and carry the designated information from one to another. Notice how the direction of the selected Data Flow is displayed in addition to its description on the related Selection Tool screen.

We continue this step by selecting the Data Flows between all elements that was defined with the system described in the eCall Example (see Figure 19). The selection includes Data Flows connecting eCall primary functions to each other, the Data Stores and Terminators in addition to some of the Data Flows connecting the functions that will be added in the subsequent passes (2.1.5 and 2.1.9). Note that the Data Flows connecting the Data Store D2.1 and the Terminator Emergency Operator is not available since none of the primary functions have communication with them defined in the Functional Viewpoint. We will be adding those elements and the additional Data Flows to and from those elements in the subsequent passes.

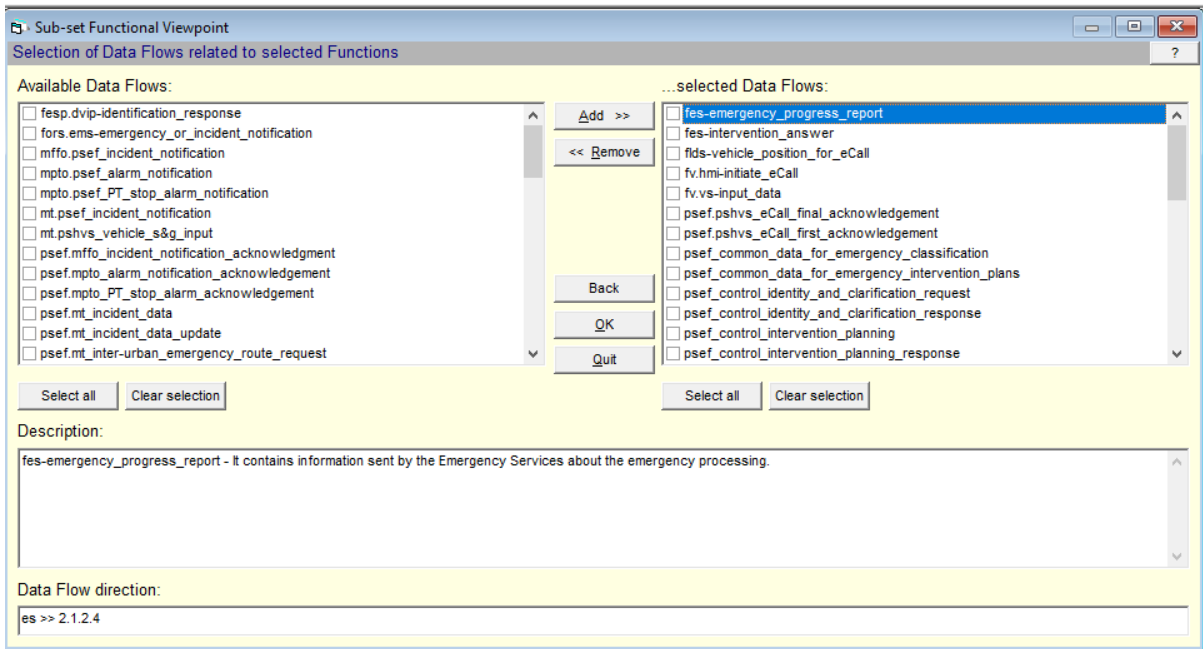


Figure 19 - Selecting Data Flows (source: author)

The Frame Selection Tool offers the Data Stores based on the Data Flows that were selected in the previous step. We continue by adding the Data Store to our eCall Architecture (see Figure 20).

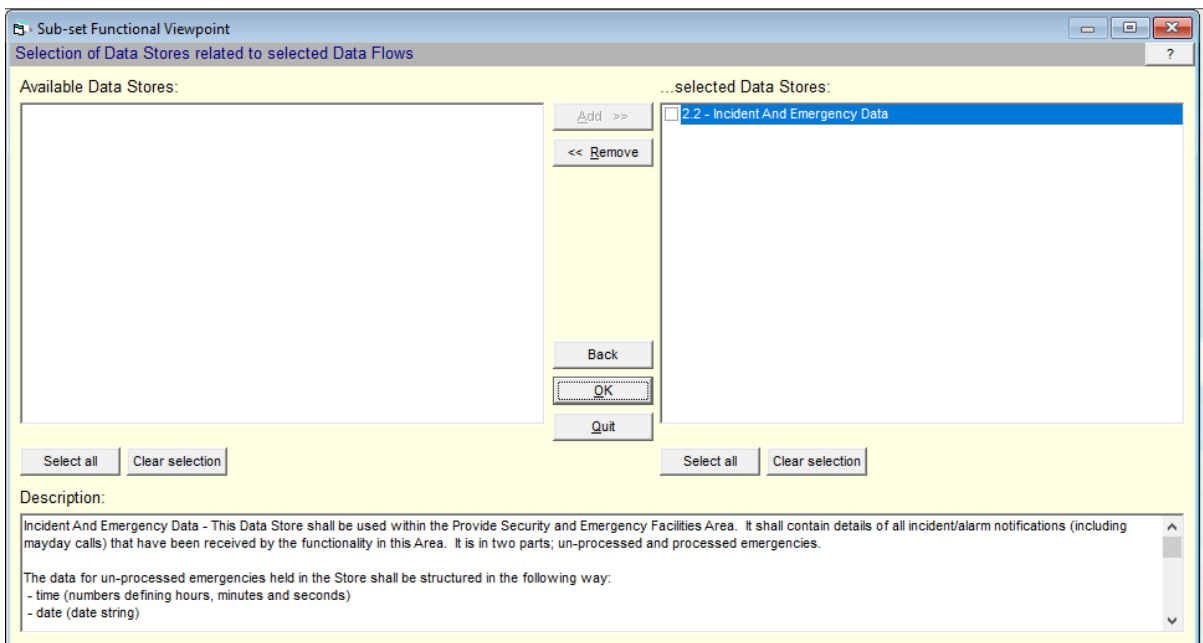


Figure 20 - Selecting Data Stores (source: author)

In the next step, the Selection tool offers all Data Flows related with the Data Stores that were selected in the previous step, the same way it offers the Data Flows for the Low Level Functions, including the flow connecting elements that are not yet included in the architecture. The Data Flows between the Data Stores and the primary functions were already added to the architecture in the Data Flow selection step and we see those data flows already on the right half on the screen when it is first opened. The user will be able to discover if more functions are related to the Data Stores in their architecture and add the Data Flows connecting additional functions if they decide

to expand their selection. Again, the Selection Tool will throw an error at the end of the first pass if the user adds any Data Flow that links functions that are not yet included in the architecture.

All Data Flows required for D2.2 Incident and Emergency Data was already added to the architecture in the data flow selection step to our architecture for the eCall example, we continue to next step without making any selection (see Figure 21).

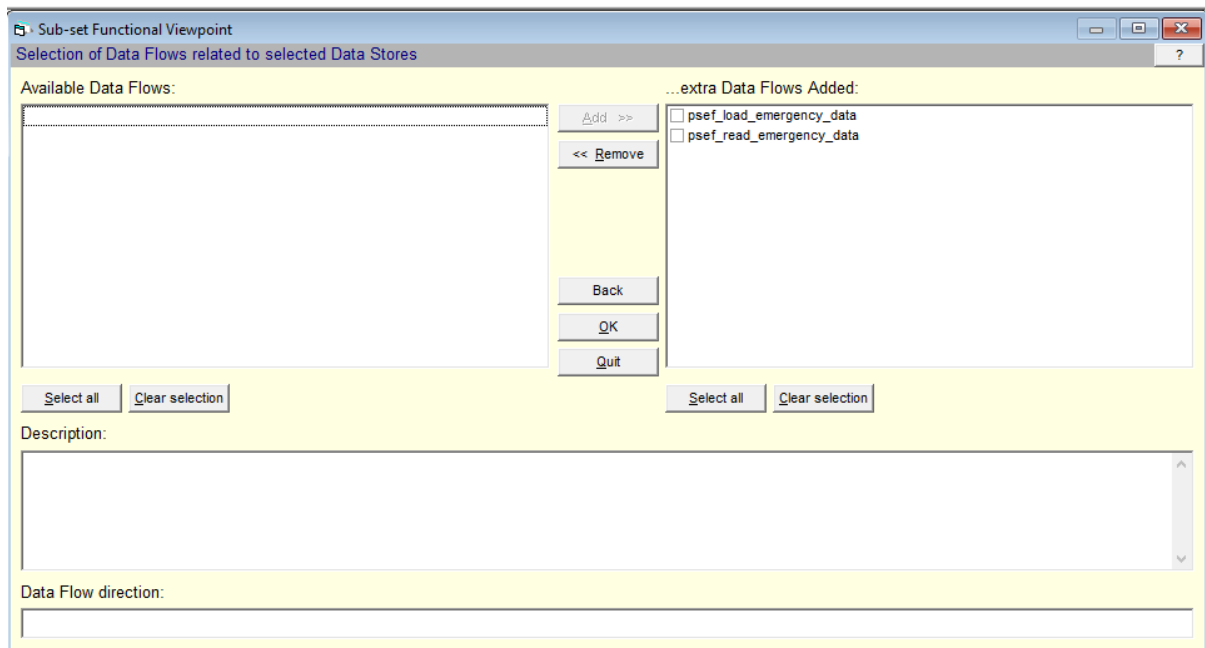


Figure 21 - Selecting additional Data Flows related to the selected Data Stores (source: author)

Since the available Terminators are offered based on the Data Flows selected previously there will not be any additional Terminator offered in this step (see Figure 22). The 4 out of 5 Terminators that were defined in the eCall example were connected to our system with the selection of Data Flows. The Emergency Operator will be added in the subsequent passes.

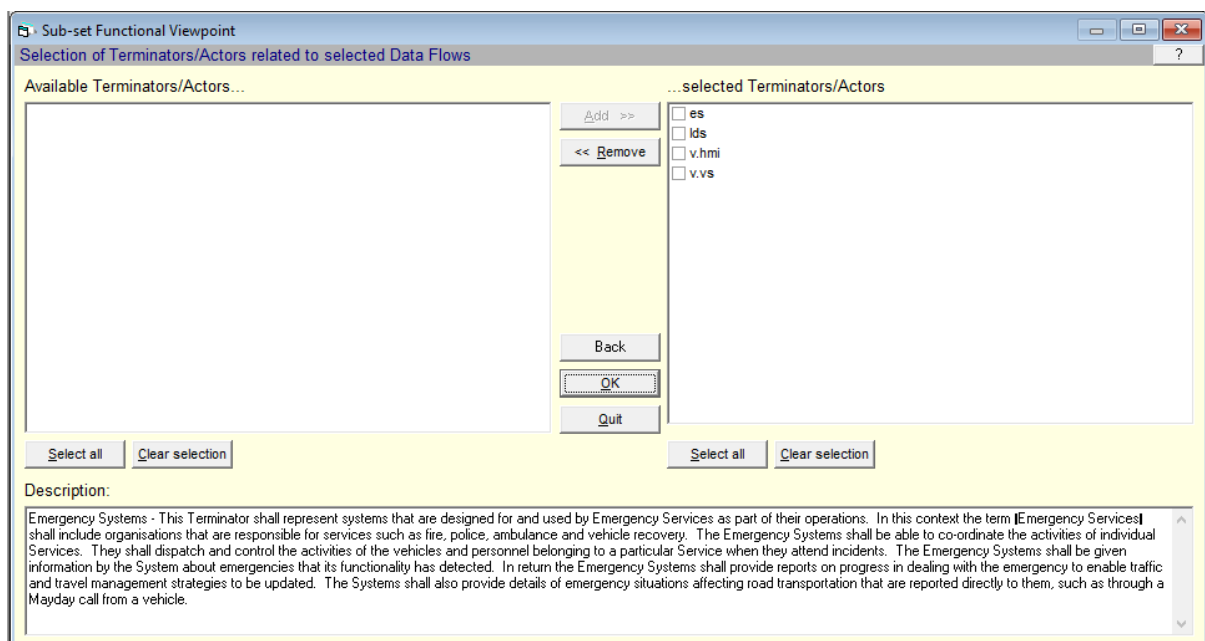


Figure 22 - Selecting Terminators (source: author)

The final screen of the first pass displays the logical errors found within the selection made so far. The possible errors are:

- Data Flows that has one end free.

These are the Data Flows that do not have a source or a destination terminator. Since the selection of data Flows is initiated by selection of functions and other elements in the Selection tool, it is impossible to include a Data Flow with both terminators missing. The Data Flows in FRAME Functional Viewpoint are defined individually to carry specific pieces of information between certain functions; the error message indicates which end of the Data Flow was found to be missing. The architecture user then add the specific element that was found to be missing for the Data Flows or remove the Data Flow that is causing the error from their architecture to correct this type of error.

The below screenshot is generated with the selection we made following the eCall example (see Figure 23). The errors thrown on final screen of the first pass selection are related to missing functions 2.1.5 and 2.1.9 which are needed to be included in the system described in the eCall example. We have selected Data Flows related to those functions deliberately after selecting the primary functions as it was mentioned above.

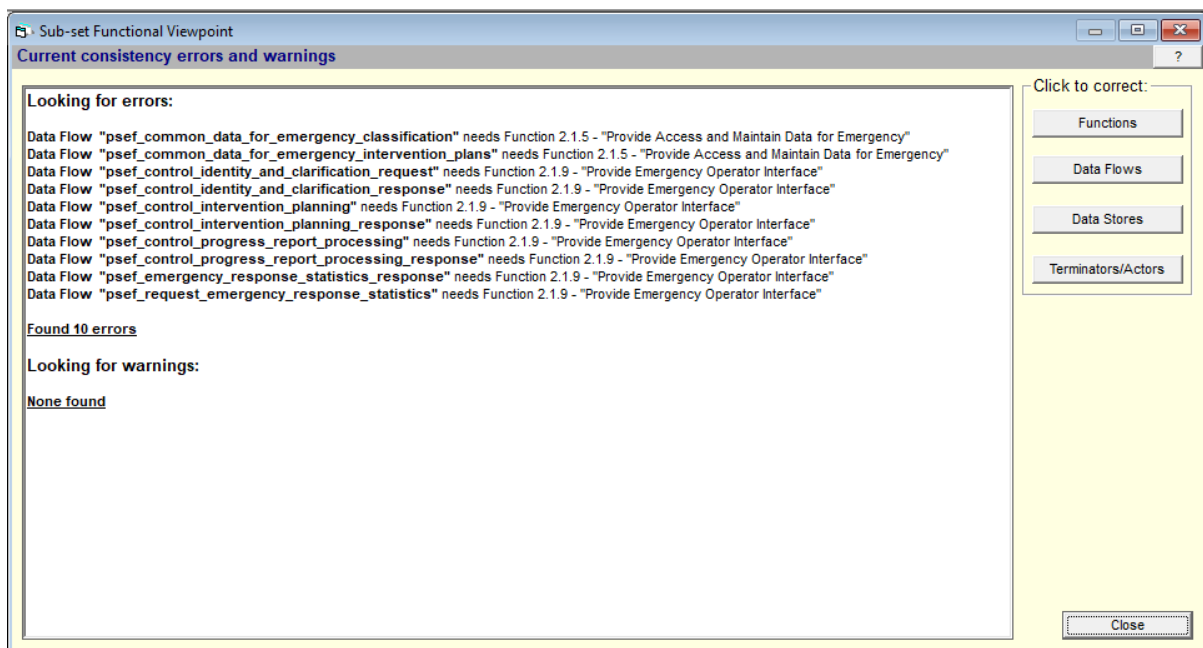


Figure 23 - Final screen of the first pass for ECall example (source: author)

- Elements that are not linked to any Data Flows.

These are elements added to the architecture but not connected, in other words not interacting with the system described in the architecture. Each function defined in FRAME Functional Viewpoint to interact with at least one other function, data store or terminator and these interactions most of the time are two way, where an element send some information and receive some information back as a response. Any element added to

project architecture should have at least one, in most of the cases more Data Flows defined to start or finish with that element. The error screen assist user to correct this type of error by suggesting possible Data Flows related with the particular element. Architecture user may follow the suggestions and connect the elements causing the error with Data Flows or remove the element form their Functional Viewpoint.

Below screenshot was created to display this types of error only and is not related to our eCall example (see Figure 24).

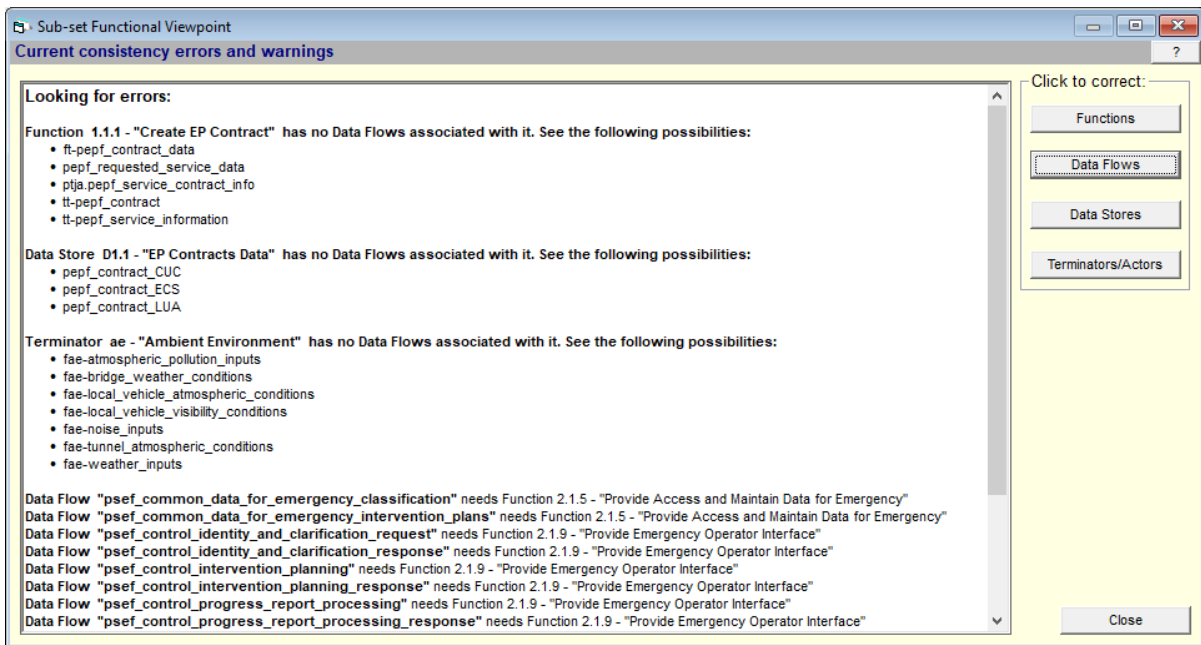


Figure 24 - Final screen of the first pass with additional errors (source: author)

There are also Warnings displayed in the final screen addition to the errors. The warnings occur when a function or a Data Store is connected with only one Data Flow. While it most of the time indicates a mistake in Data Flow selection since there expected to be at least two Data Flows related to each element representing the two way nature of the communication between elements, it may just be the case for that particular architecture that only one Data Flow is required to undertake the functions required by the system of interest. Warnings are not binding for the resulting Functional Viewpoint and resolving all warnings is not obligatory.

The creation of Functional Viewpoint ends when all desired elements are added in the architecture with no logical errors. Some warnings may remain.

The user is expected to correct the errors by adding or removing elements from their architecture. User also can add / remove items that were not offered by the selection of the User Needs to achieve the correct architecture describing the systems that they have interest with. This phase is called the subsequent passes.

The user is expected to continue the subsequent passes with the buttons displayed on the right end of the final screen. The user may start by adding / removing Low Level Functions, Data Flows, Data Stores or Terminators. The Selection Tool will follow through the architecture creation steps until the final screen as it was in the first pass no matter which step the user starts with. This is

because changes made in each step of the architecture may cause or require modifications to be made in the following steps.

We continue this step by adding the missing Functions from the eCall Example that were not offered by the User Needs selection (see Figure 25). Adding these functions to the architecture will also resolve the errors that were found in the final screen.

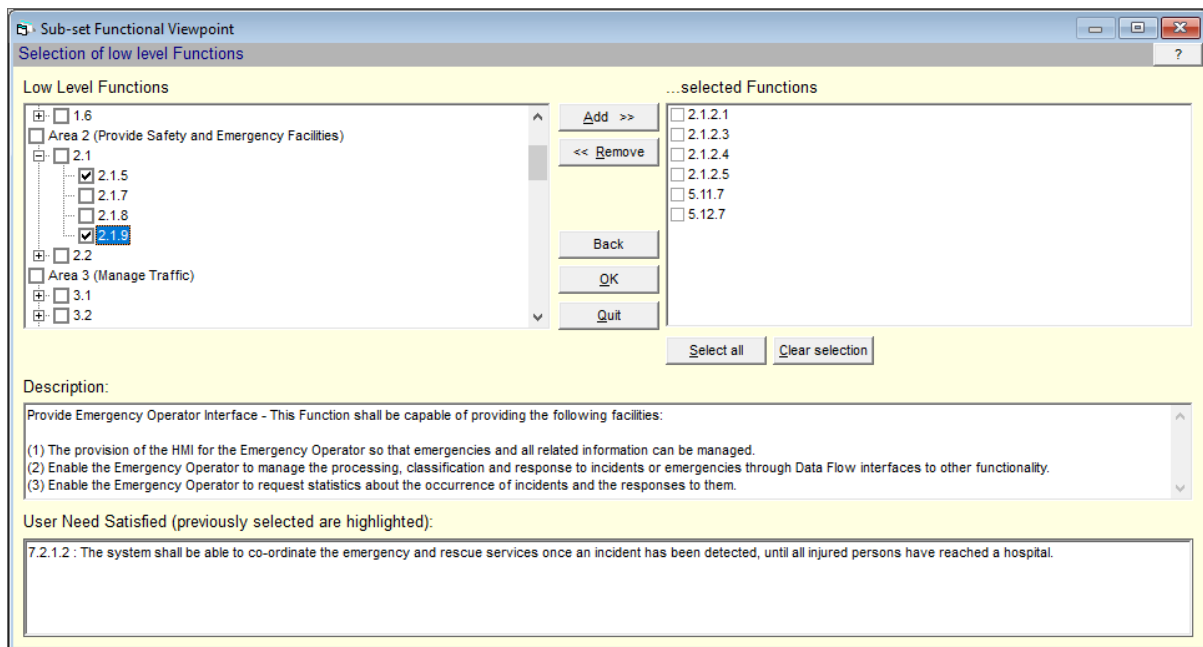


Figure 25 - Adding missing Low Level Functions to the architecture (source: author)

The next step requires attention to detail since the Data Flows from that were not selected with first pass will be listed again in this step along with the new Data Flows that would be offered with the functions that were added to the architecture in the last step (see Figure 26). User is expected to filter the redundant Data Flows select only the Data Flows necessary for their system(s).

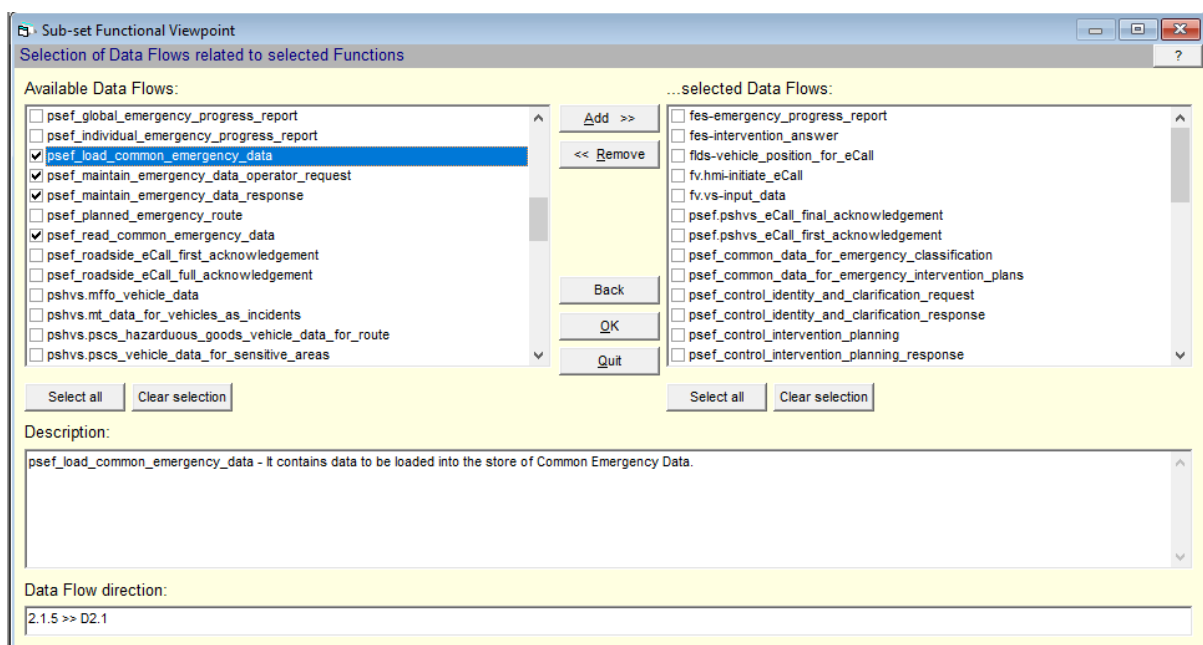


Figure 26 - Adding Data Flows missing between newly added elements (source: author)

The Selection Tool offers the list of all Data Stores that were defined in FRAME Functional Viewpoint (see Figure 27). The only Data Store related with the newly added functions will be selected to achieve the system that was described by the eCall Example, which is also the only Data Flow missing to create the architecture described in the eCall example.

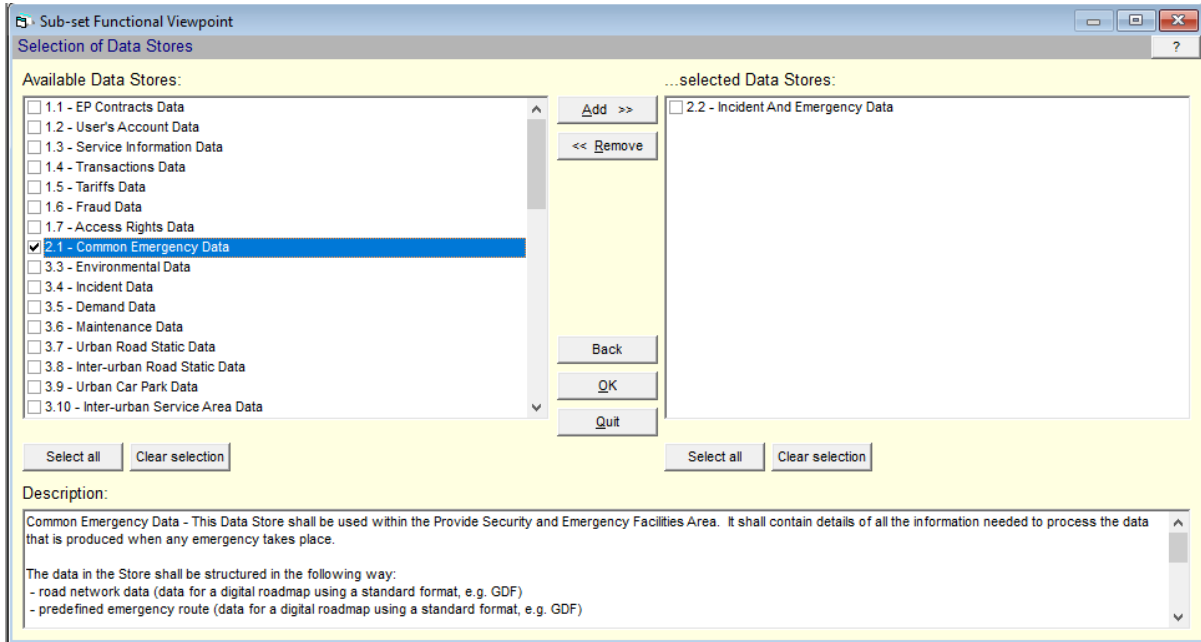


Figure 27 - Adding missing Data Stores (source: author)

The Selection Tool will proceed with offering the set of all Terminators available in the FRAME architecture. We will continue this step with including the Emergency Operator to complete the architecture described in the eCall example (see Figure 28).

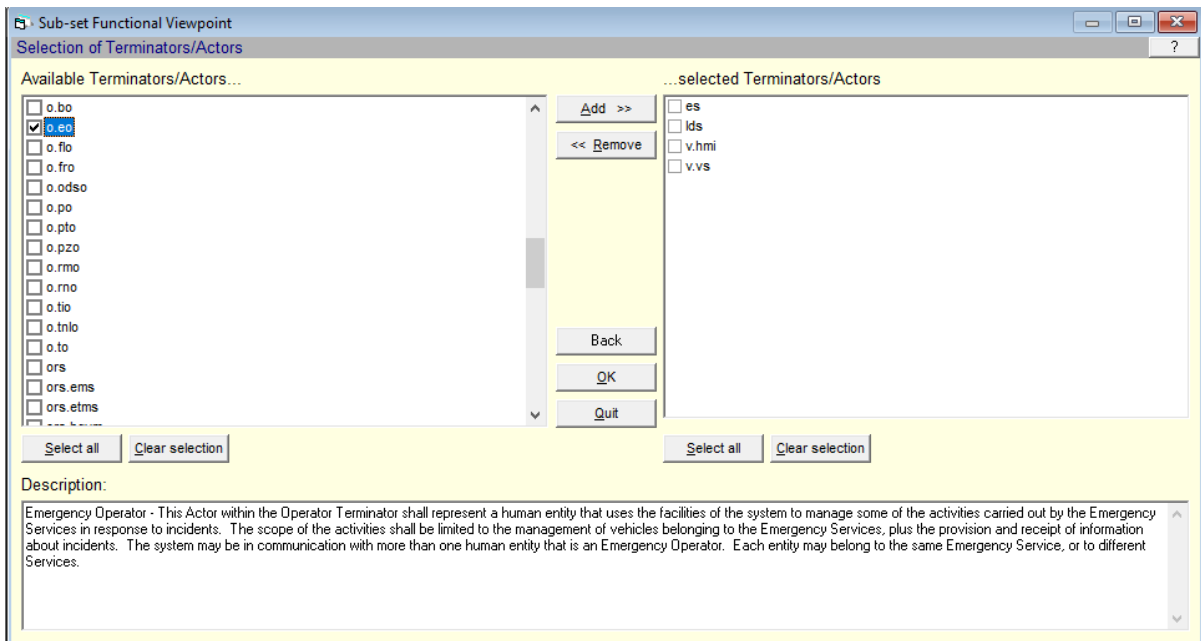


Figure 28 - Adding missing Terminators (source: author)

With the subsequent passes we can see that the resulting architecture doesn't have any errors (see Figure 29). The creation of the Functional Viewpoint ends when all desired elements are added to the architecture and there are no more errors.

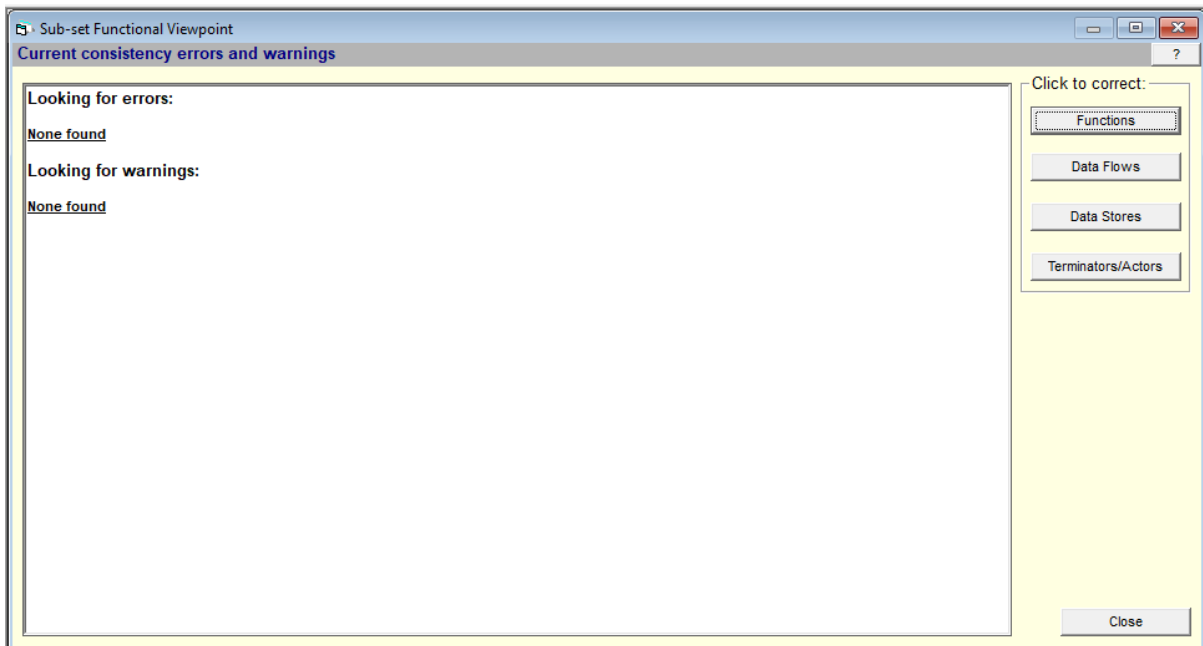


Figure 29 - Final Screen at the end of the subsequent passes (source: author)

The architecture user may continue the subsequent passes by adding and removing elements and reaching to the final screen as many times it is necessary for them to build their desired Functional Viewpoint. Since we have added all necessary elements mentioned in the eCall example and no logical errors was found for our architecture, and we can finish creation of our Functional Viewpoint.

The products of the Functional Viewpoint in Selection Tool is a handful of reports listing the elements selected by the architecture user and the unselected Data Flows that are related with selected functions and Data Stores (see Figure 30).

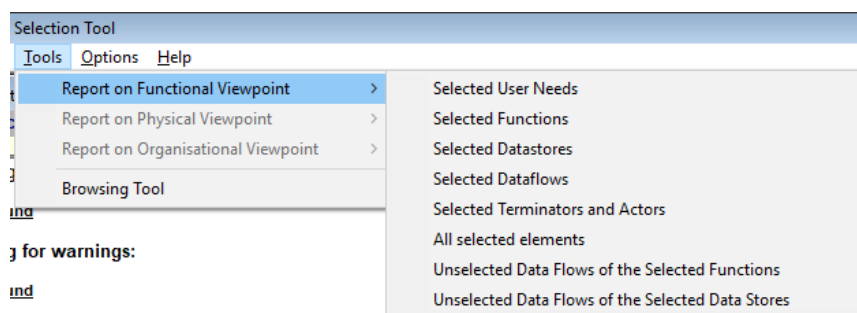


Figure 30 - The Functional Viewpoint artifacts that can be produced with FRAME (source: author)

The creation of Physical Viewpoint starts by defining Sub-systems and Locations (see Figure 31). The user is required to populate both fields themselves according the Physical Viewpoint they have in mind. The Locations created for one Sub-system will be added to Location drop down menu and may be used to assign other Sub-systems. Following the eCall example we start the

Physical Viewpoint creation by creating two sub-systems in two different locations: In-vehicle Component in the location Vehicle and PSAP in location Emergency Centre.

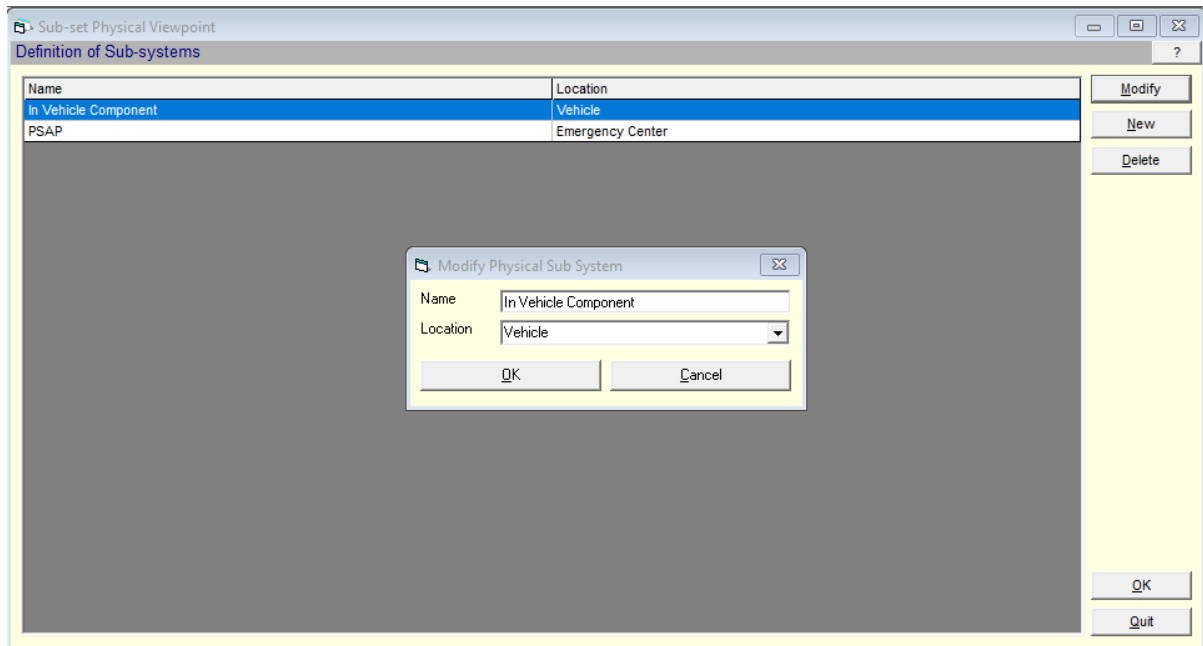


Figure 31 - The Sub-systems and Locations created for the eCall example (source: author)

The architecture user is expected to relate each function and Data Store from the Functional Viewpoint with the Sub-systems in the Physical Viewpoint and distribute the functions to physical locations. We continue this step by differentiating the Functional Viewpoint elements into two physical locations, the elements related to the vehicle goes to In-Vehicle Component and others go to the PSAP (see Figure 32).

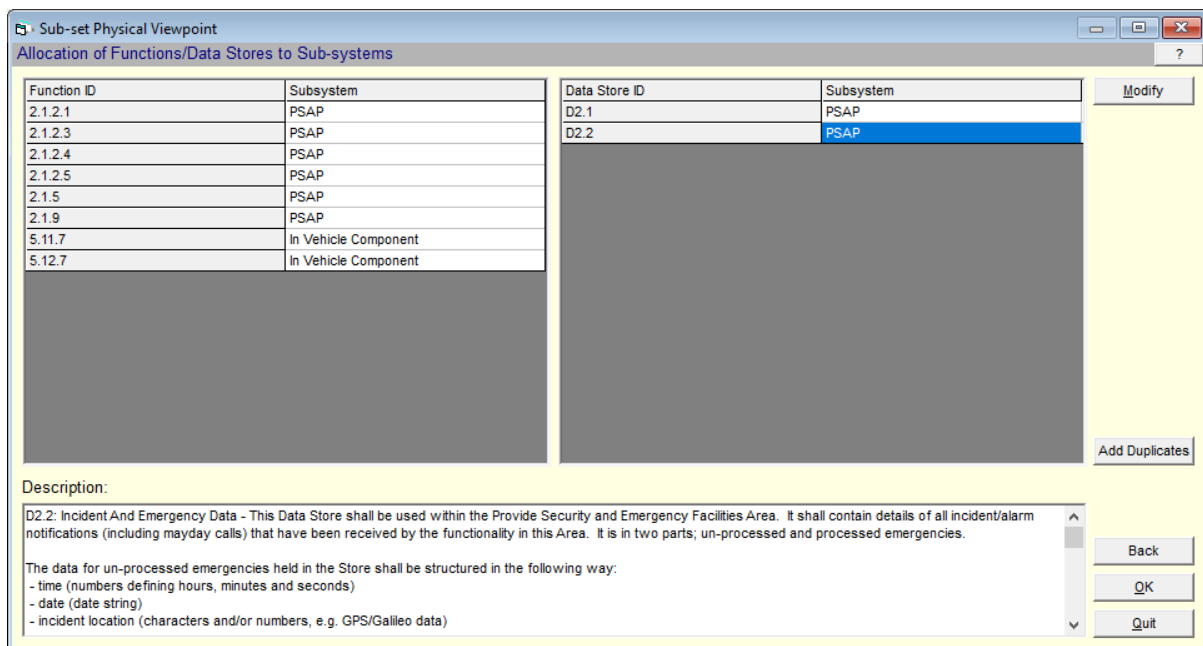


Figure 32 - Distributing the functions and Data Stores to Sub-systems (source: author)

In the next step the user may create Modules that would bundle similar functions together inside a Sub-system (see Figure 33). Given the simplicity of the system described in the eCall example we will not be specifying any Modules in our architecture. We have created the below imaginary modules for this step to demonstrate how an architecture where distributing functions between modules would be required.

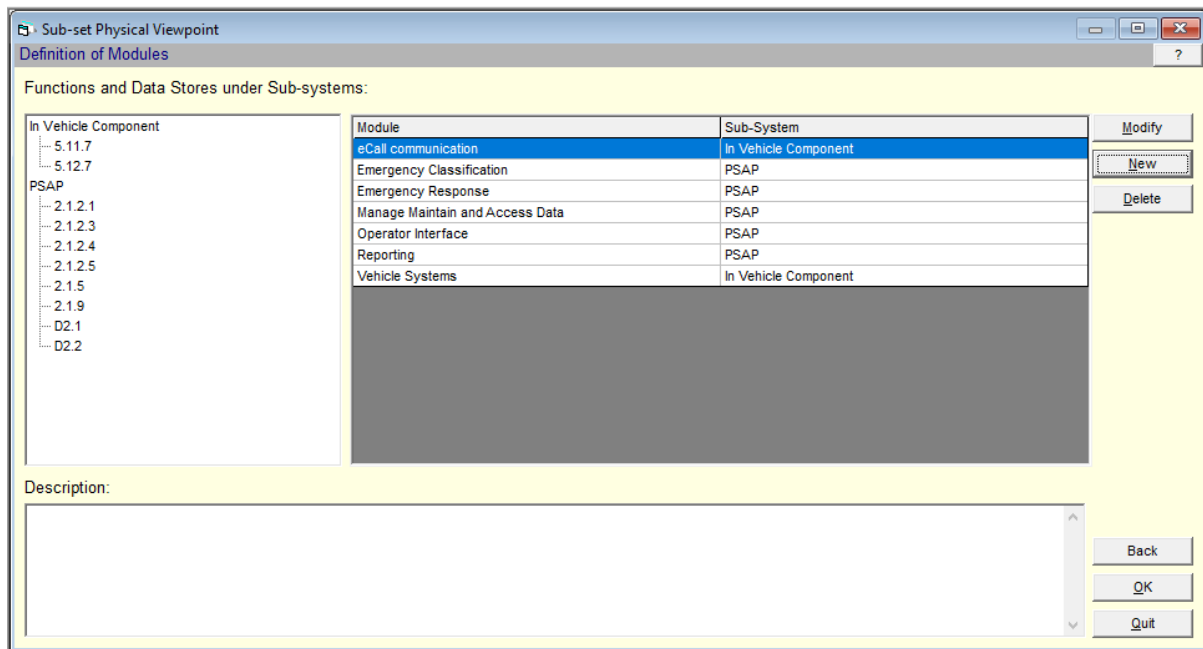


Figure 33 - Creating Modules for the Physical Viewpoint (source: author)

User is required to assign the Low Level Functions to Modules to define which functions will be residing in which module (see Figure 34).

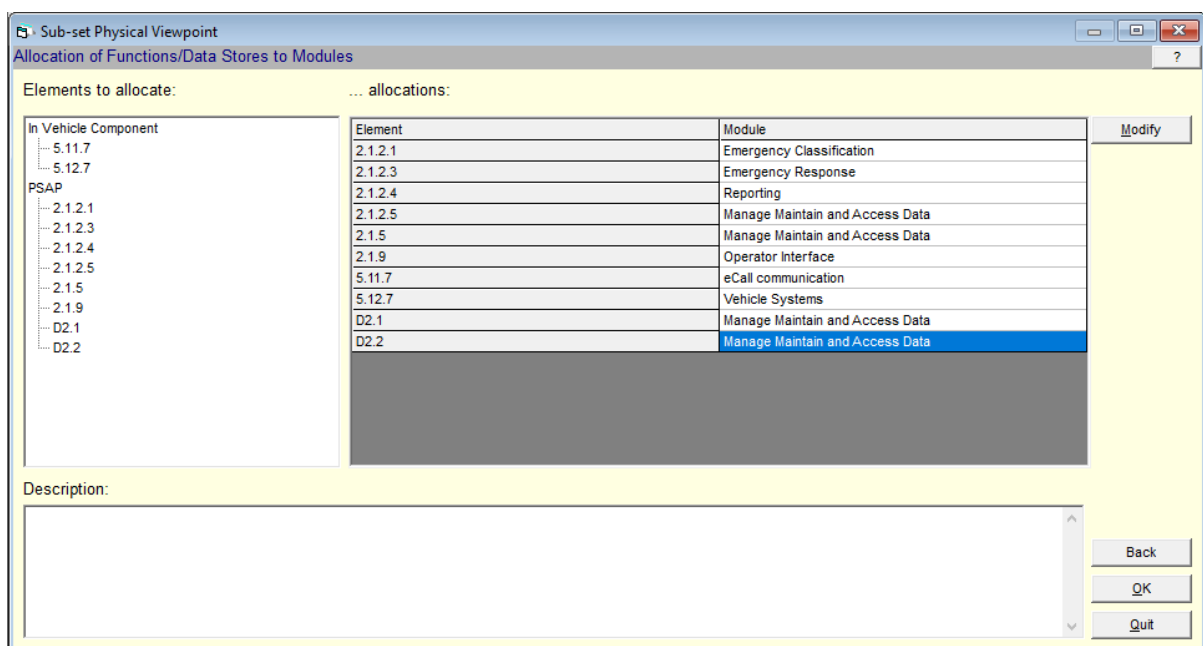


Figure 34 - Distributing the functions and Data Stores to Modules (source: author)

At the final screen of Physical Viewpoint creation the user is displayed the information flows that is required between the Modules and the terminators of the system described in the architecture (see Figure 35). Note that these flows include both the communication inside the sub-systems and between them.

| ID | Parent | Target |
|--------|--------------------------|---------------------------------|
| P01001 | Emergency Systems | Emergency Response |
| P01002 | Emergency Systems | Reporting |
| P01003 | Emergency Systems | Manage Maintain and Access Data |
| P01004 | Emergency Systems | PSAP |
| P01005 | Location Data Source | eCall communication |
| P01006 | Location Data Source | In Vehicle Component |
| P01007 | eCall communication | Vehicle Systems |
| P01008 | eCall communication | Emergency Classification |
| P01009 | eCall communication | PSAP |
| P01010 | eCall communication | Human Machine Interface |
| P01011 | Vehicle Systems | eCall communication |
| P01012 | Emergency Classification | eCall communication |
| P01013 | Emergency Classification | Emergency Response |
| P01014 | Emergency Classification | Manage Maintain and Access Data |
| P01015 | Emergency Classification | Operator Interface |
| P01016 | Emergency Classification | In Vehicle Component |
| P01017 | Emergency Response | Emergency Systems |

Description:

Back

Close

Figure 35 - Physical Data Flows for the Physical Viewpoint. (source: author)

Since there will be no modules defined for our eCall example, the below description of the Physical Viewpoint would be more accurate (see Figure 36).

| ID | Parent | Target |
|--------|-------------------------|-------------------------|
| P03001 | Emergency Systems | PSAP |
| P03002 | Location Data Source | In Vehicle Component |
| P03003 | Emergency Operator | PSAP |
| P03004 | In Vehicle Component | PSAP |
| P03005 | In Vehicle Component | Human Machine Interface |
| P03006 | In Vehicle Component | Vehicle Systems |
| P03007 | PSAP | Emergency Systems |
| P03008 | PSAP | Emergency Operator |
| P03009 | PSAP | In Vehicle Component |
| P03010 | Human Machine Interface | In Vehicle Component |
| P03011 | Vehicle Systems | In Vehicle Component |

Description:

This physical data flow includes the following functional data flows:
fes-emergency_progress_report
fes-emergency_services_information
fes-intervention_answer

Back

Close

Figure 36 - Physical Data Flows for the eCall example Physical Viewpoint (source: author)

After finishing creating the Physical Viewpoint the user can produce few reports related to it describing the allocation of the elements from the Functional Viewpoint to the Modules and Sub-system of the Physical Viewpoint (see Figure 37).

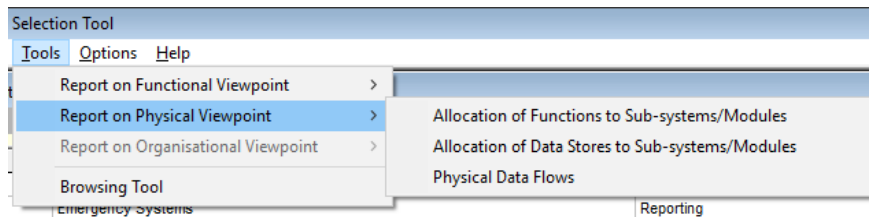


Figure 37 - The Physical Viewpoint artefacts that can be produced with FRAME (source: author)

Having the Physical Viewpoint created the architecture user then can start defining the Organizational Viewpoint for their architecture. User is not obliged to follow any order in creation of viewpoints after they have completed creation of the Functional Viewpoint for their systems but we merely followed how the architecture creation is presented throughout FRAME documentation.

User starts by creating definitions for the stakeholders of system(s) of interest (see Figure 38). The Low Level Functions and the Data Stores of the Functional Viewpoint will be allocated to stakeholders defined in this step while creating the Organizational Viewpoint, which is why we have defined only two stakeholders in this architecture. We find the way the Organizational Viewpoint was organized in Selecting Tool to be limited since no other role or responsibility could be assigned to stakeholders apart from being responsible of sending and receiving some Data Flows in the Functional Viewpoint and only one stakeholder can be assigned for each Low Level Function. As a result most of the stakeholders are excluded from the Organizational Viewpoint.

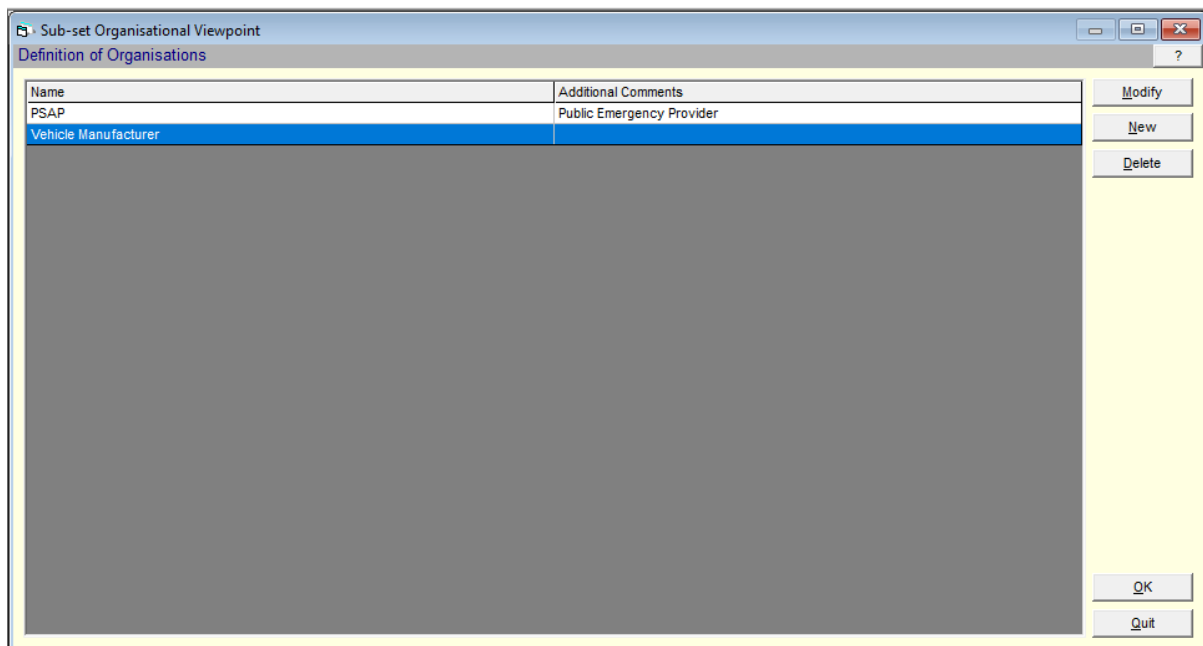


Figure 38 - Defining Stakeholders (source: author)

We continue allocating the Functional Viewpoint elements to stakeholders (see Figure 39). As mentioned above only one stakeholder can be defined for each function. Describing the

Organizational Viewpoint in this way comes short in cases where some functions require interaction between two stakeholders. Who will be responsible of the function in that case? Also it is not possible to specify the nature of the relationship between the stakeholders and functional Viewpoint elements are not defined. Do they build, own or operate the functions or Data Stores? What if in case one stakeholder maintains the Data Store and another uses it?

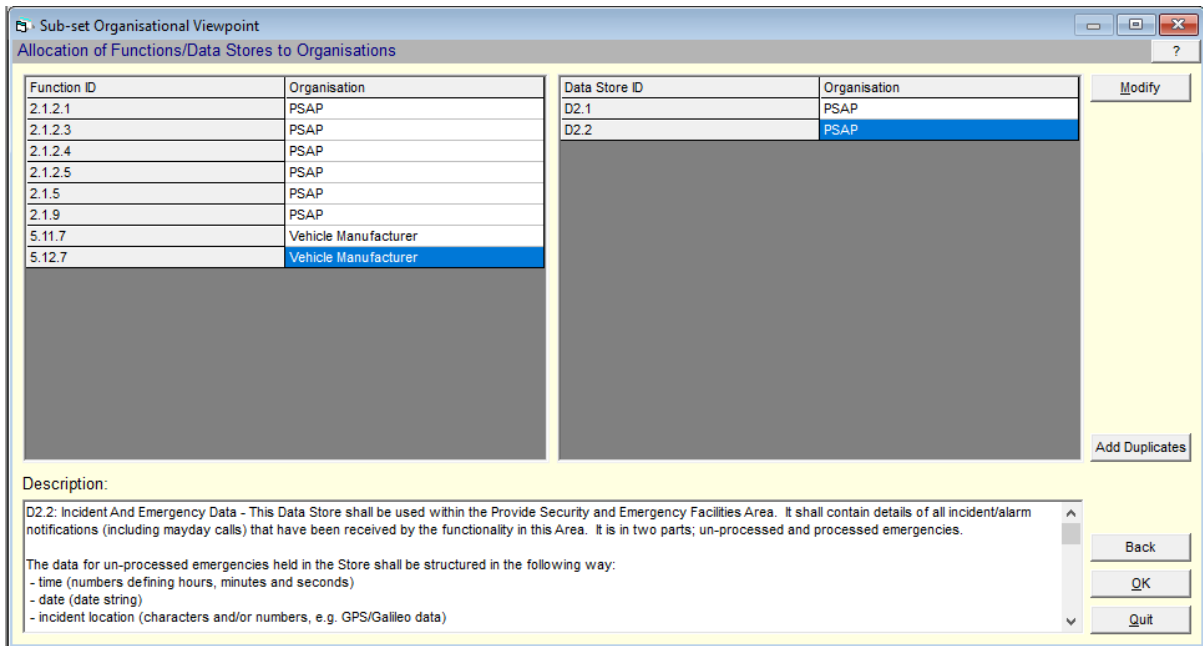


Figure 39 - Defining Stakeholder Responsibilities (source: author)

At the end a set of Organizational Data Flows is listed, showing which Data Flows must travel between the Low Level Functions and Data Stores related to defined stakeholders and the terminators of the system(s) (see Figure 40).

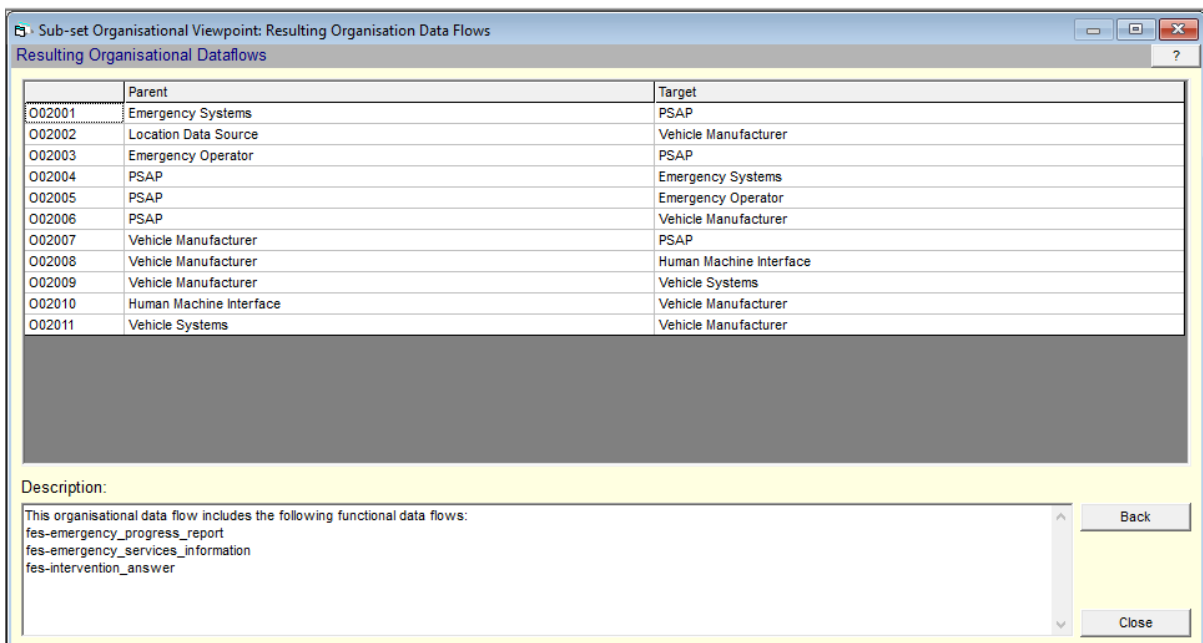


Figure 40 - Organizational Data Flows (source: author)

Similar to the Physical Viewpoints, few reports can be generated listing the allocation of Functional Viewpoint elements to stakeholders and the Organizational Data Flows (see Figure 41).

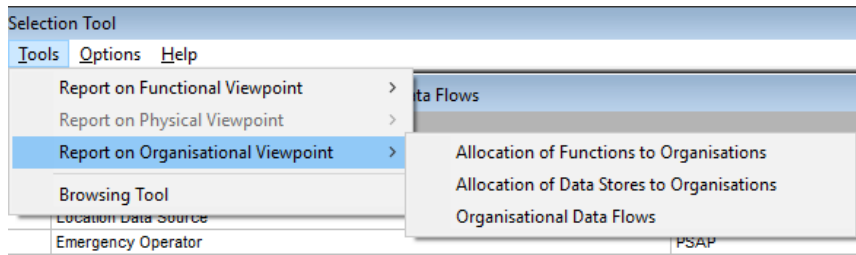


Figure 41 - The Organizational Viewpoint artifacts that can be produced with FRAME (source: author)

Chapter 3.3 - Creating an Architecture Description of The Mayday Notification Services in Arc-IT

Arc-IT offers the computer based Regional Architecture Development Tool for Intelligent Transportation, Rad-IT, for its user to create architecture description of their projects. The user selects the set of elements; the services provided by those elements and define the interaction between those elements to be able to provide the services desired by the stakeholders of the system(s). Rad-IT helps architecture user to create and define artefacts for Physical, Functional and Communication Views related to their architecture.

The first screen in Rad-IT is the “Start” screen where users expected to define basic information for their regional architecture (see Figure 42). As soon as the user hits the “New” button the fields on the right half of the screen becomes editable. The user is then expected to give a name, write a description, define a time frame, the geographical and service scope of their architecture. These fields are all text based and user is expected to populate the fields their own. The information entered by the architecture user will be used in later steps of architecture creation and will be the source for the documents and charts that the user can produce with Rad-IT.

The screenshot displays the 'Start' tab of the Rad-IT application. At the top, a navigation bar includes tabs for Start, Planning, Stakeholders, Inventory, Services, Needs, R & R, Functions, Interfaces, Standards, and Agreements. Below this, a header indicates the 'Current Region: Mayday Notification'. The interface is divided into several sections:

- Architectures:** A list on the left shows 'Regional' with 'Mayday Notification' selected, and 'Project' with 'E Call Project'. Buttons for 'Region to Project', 'Project to Region', 'New', and 'Delete' are present.
- Regional Architecture Attributes:** A form on the right with fields for:
 - Name:** Mayday Notification
 - Description:** This service corresponds to E Call in FRAME. This architecture will be created following the E Call architecture in FRAME that was created earlier in the paper to be able to compare architecture creation methodologies and the final products of Arc-IT and FRAME.
 - Timeframe:** not specified
 - Geographic Scope:** Capital City Prague
 - Service Scope:** Mayday Notification
 - Developer:** Mert Aksac
 - Maintainer:** (empty)
 - Version:** 1.0
 - Date/Time:** 05/12/2019 12:59:12
- Related:** A list on the left with 'New' and 'Delete' buttons.

At the bottom right, there are buttons for 'Change Log', 'Apply', and 'Cancel'.

Figure 42 - The Start Tab (source: author)

User needs to hit the “Apply” button and confirm the changes made for each step in Rad-IT before continuing to other fields or steps in the architecture creation.

Rad-IT offers the ability of creating projects under the regional architecture where user can divide their regional architecture into projects according to their needs. These projects may be used to cluster services that are similar or complementary to each other, services that are planned to be deployed in various phases or timeframes, services that will be supplied by a specific vendor or stakeholder or according to any other way that the architecture user would decide to organize their projects. The architecture elements for the Regional Architecture and any of its Projects are defined separately for each. Rad-IT keeps a list of all elements created throughout the Regional

Architecture so that these elements may be “Applied” to each other where necessary. We created a single project and named it “E Call Project” for our Regional Architecture in our example since the architecture will cover only one Service Package that will proposed in isolation of all other possible ITS.

One of the main functionality that is offered to architecture user in this step is Architecture Maintenance, where user can define the developers and maintainers of the architecture and assign to it a version name or number (see Figure 43). Arc-IT offers its user a Change Log, where the user can record and track the changes made to the selected architecture. Entries on the Change Log contain the timestamp, the name and the version of the architecture and the description of the changes made where all fields expected to be populated by the user.

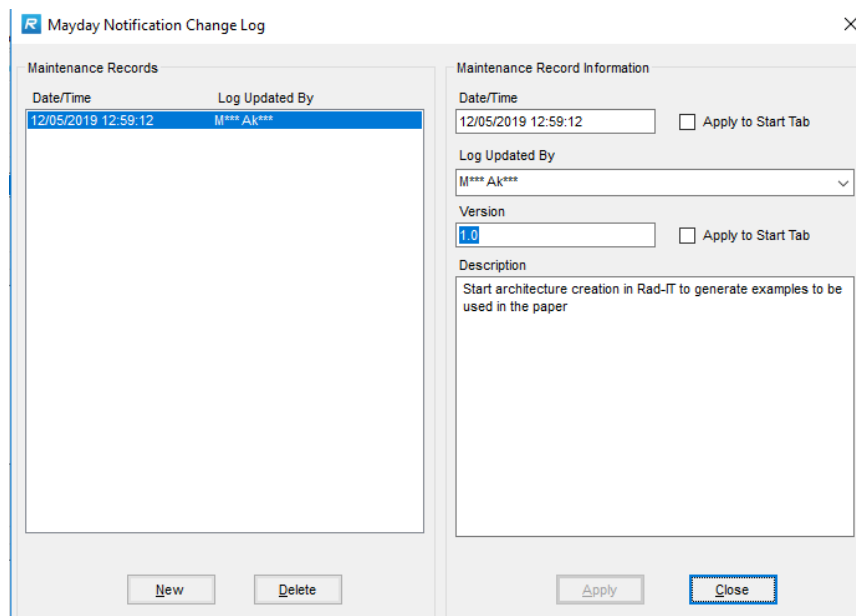


Figure 43 - The Change Log (source: author)

Rad-IT is a tabular based product where user can navigate the architecture creation steps through the tabs. User needs to create a new architecture or open an existing one to unlock the tabs. Once an architecture is created or selected the user is expected to follow the architecture creation steps in the order of the tabs since changes made in each step will be determining for the following steps. Nevertheless the user will be able to go to any tab they desire anytime during the architecture creation to skip or revisit necessary tabs.

The Planning Tab is where the user defines objectives and strategies for their Regional or Project Architecture (see Figure 44). These components are organized in two levels where and Objective parent the subcomponent Strategy. One or more strategy may be defined for each Objective. The user populates the goals for their architecture themselves by entering a name and a description for their objectives / strategies and can assign numbers to each to be able to sort them. A source can be defined for each objective such as National or Regional Traffic Management Policies, Government Incentives and Programs etc.

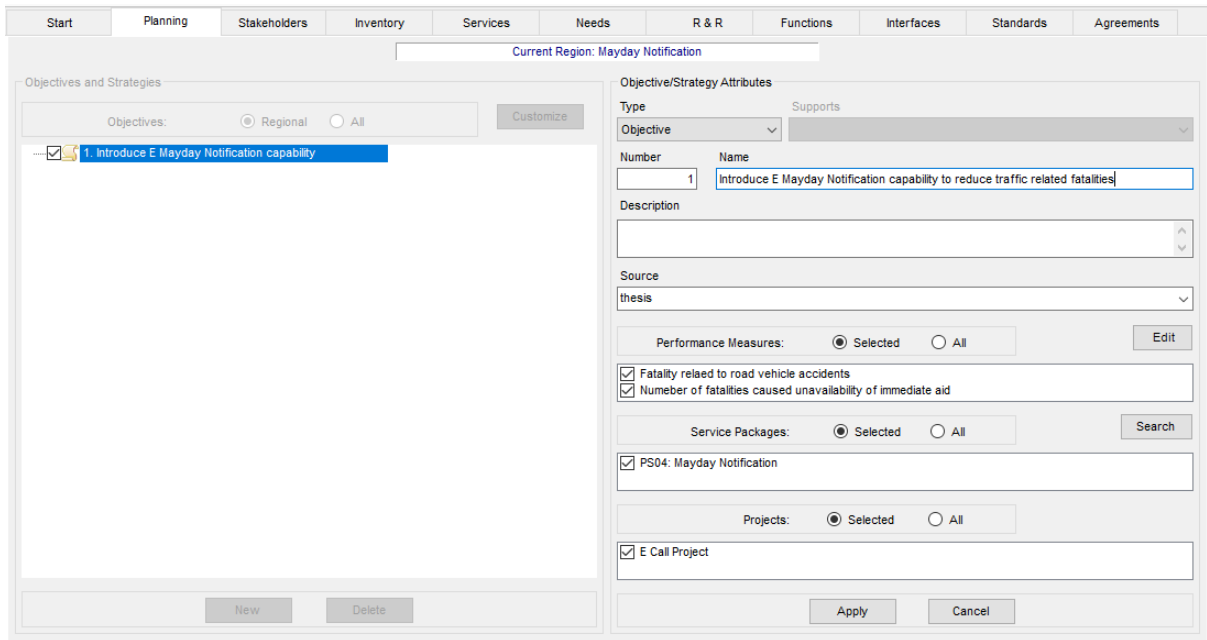


Figure 44 - The Planning Tab (source: author)

The user is expected to set Performance Measures for each objective / strategy they have added to their architectures (see Figure 45). The Performance Measures are expected to be populated by the user. The user can define categories for the criteria that they will be creating and assign numbers to be able sort them.

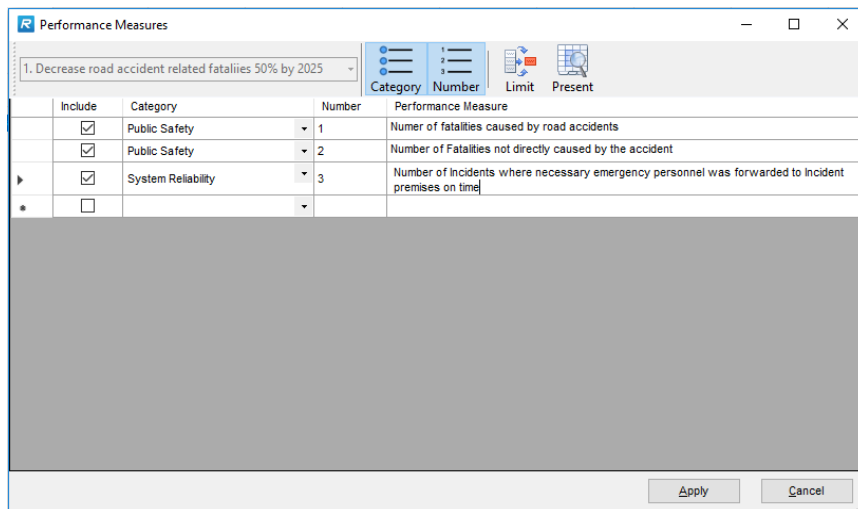


Figure 45 - Defining Performance Measures (source: author)

Lastly the user is expected to link the Service Packages that will fulfil the objective / strategies to define how the ITS architecture relates to the Transportation Plans of the region / project. The service packages selected to be associated with the objectives / strategies in this step will allow Rad-IT to narrow down the Service Packages that will be offered in the Services Tab (see Figure 46).

Next step in architecture creation is the Stakeholders Tab. The user can define individual stakeholders or associate them with Stakeholder Groups. The stakeholders defined in this step will

be used to associate them with inventory elements, agreements, needs and define the operational concept by identifying their roles and responsibilities in the architecture.

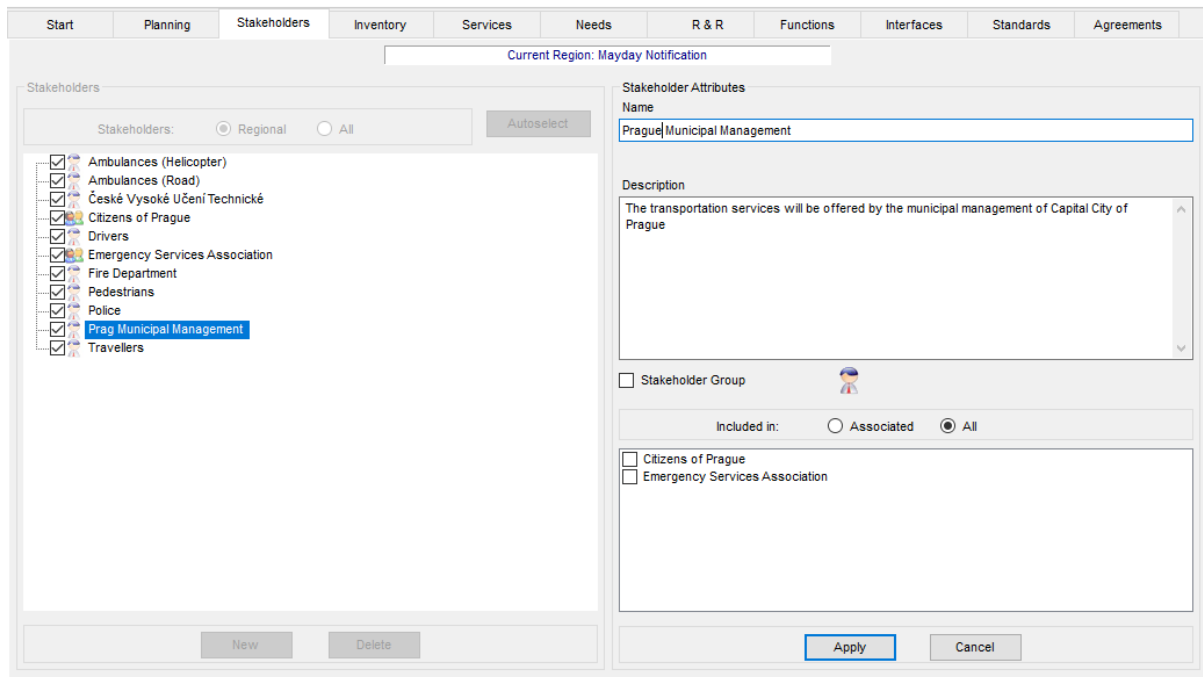


Figure 46 - The Stakeholders Tab (source: author)

In the next tab the architecture user has to define elements that will be used in their architecture. This step is the most crucial step in Arc-IT since user will be creating the Physical View of their architecture. Since Arc-IT distributes Functional Objects directly associated to the Physical Objects, the elements created in this step will be the foundation to describe the functionality of the system(s). Since the Communications View consist of the Profiles describing the Protocols and Standards of the necessary communication links between the Physical Objects, the elements created in this step will also be used as an entry to the Communications View. Finally the stakeholders defined as owners of the elements in this stage will be the foundation for agreements in the Organizational View of the architecture.

The user is expected to have a broad understanding of how the Physical View of the Service Packages are organised as they are described in the architecture website since they are required to create the inventory of the elements that will be necessary to completely describe the system(s) in their architecture. When creating an element the user is required to name the element and define its type and class. An element may be,

- a normal element which is a part of transportation systems,
- an instance of an element,
- a communication element,
- a shared element which is also part of a Related Architecture
- or a Human who is operator or user of the system(s).

Communication Elements provide a way to create an inventory for important communication elements that do not provide any transportation service directly. These elements will not have any interface in the physical view but the interfaces between other types of elements will be assigned

to them. A Human element is a direct terminator to the system(s) and they cannot be assigned any Functional Objects.

The elements are categorized in five Classes that were previously mentioned in the third chapter of this paper, which are Centre, Field Equipment, Vehicle, Traveller Device or Support System. The Physical Objects that will be suggested in the for the elements will be based upon their classes. User may continue defining the stakeholder that is associated with the element, define its status and populate a description for the element. An element’s status is defining its availability for the current architecture which can be existing, planned, future or not applicable.

Any element that is to carry out a transportation related function is defined as a Physical Object in Arc-IT. The architecture user should define which Physical Object the element is, to be able to assign necessary functionality to the system(s). The user is offered a list of Physical Objects of the selected type and class for the element but they are also able to select the Physical Object from a list of all Physical Objects that were defined in Arc-IT.

Following the Mayday Notification Service Package we have created an inventory of 9 elements; three Physical Objects, an Emergency Management Centre, a Vehicle OBE and a Personal information Device that will provide the system with the main functionality, 3 terminating subsystems which are Basic Vehicle, Other Emergency Centres and Other Vehicle OBEs along with 3 Human elements, the Emergency System Operator, the Driver and the Traveller (see Figure 47).

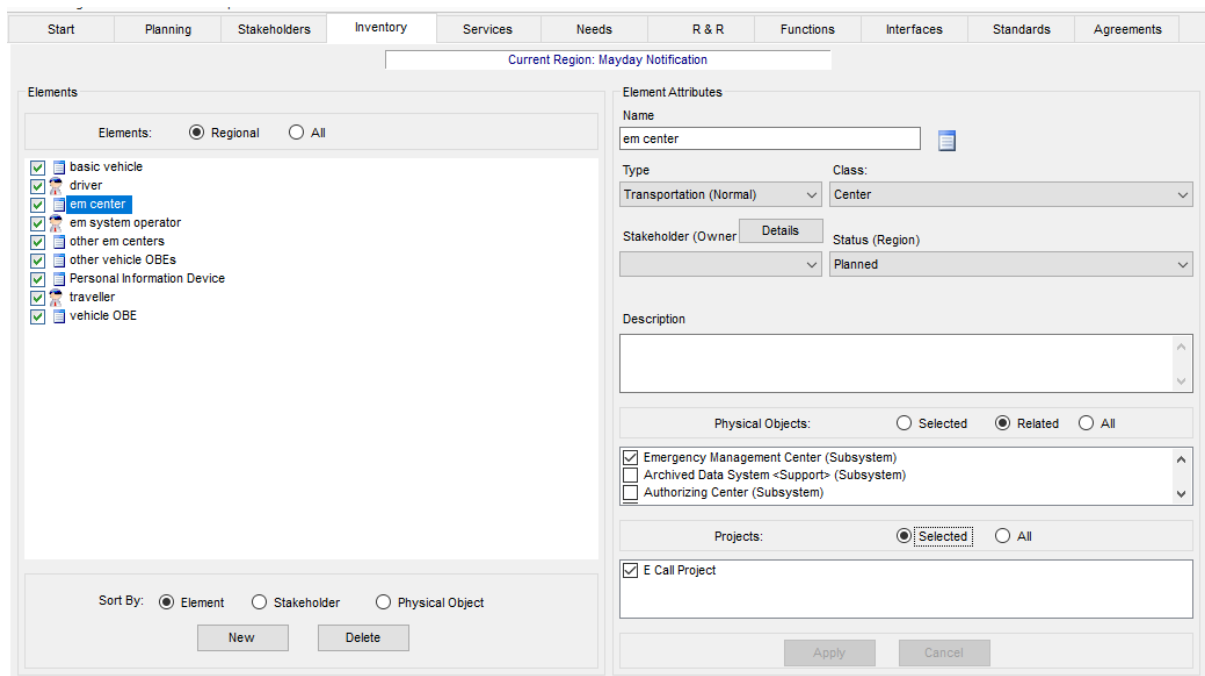


Figure 47 - The Inventory Tab (source: author)

The architecture user can further define roles of stakeholders for each element that was assigned as Physical Elements (see Figure 48). More than stakeholder can be assigned to a role for an element for different statuses; an object may be owned by one stakeholder in “Planning” status but may be owned by another in “Existing” status to be able to monitor how Roles and Responsibilities of stakeholders will be changing throughout the project life cycle.

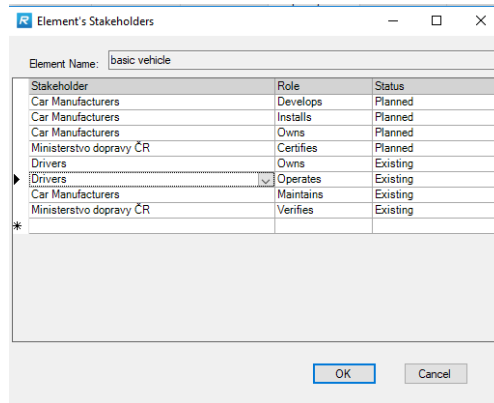


Figure 48 - Assigning Roles of Stakeholders for the elements (source: author)

The Services tab where user selects the Service Packages they would like to include to their architecture is the next step in architecture creation. User may browse the list of all service packages to go over the descriptions of them and evaluate one by one which ones to include along the way or the may user the “Autoselect” function (see Figure 49). This function helps user to reduce the list of all Service Packages into the list of Service Packages that was initially tied to an Objective / Strategy in the Planning tab.

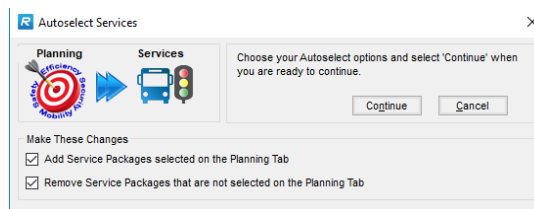


Figure 49 - Autoselecting Services (source: author)

The user continues with assigning elements that will be used in defining the Service Package and selecting the projects that the Service Package will be included. Note that these selections are for organizational purposes only and will not result with any implications for further steps in the architecture creation. The Functions that will be available to assign to the elements are related to which Physical Object they are and will not be changed or filtered in any way related to the Service Package that they are assigned in this stage.

In the next step architecture user shall chose the Needs that will be fulfilled by the system(s) described in the architecture (see Figure 51). User can define new Need Areas and add new Needs to be addressed in their architecture. The “Autoselect” function is the only option to choose among the Needs that are already defined with the architecture (see Figure 50).

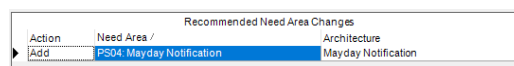


Figure 50 - Adding recommended Needs to architecture (source: author)

Architecture user is expected to go to the architecture website to study what Needs are already related with the functional requirements. The requirements analysis is already done by the Arc-IT while defining the functionality for the Service Packages. Each Need is referenced to some of the functional requirements that are satisfied by Functional Objects that were defined in the Service Package. Note that the Needs selected in this step are merely for documentation purposes only

and not definatory for the functionality of the system(s) to be defined with the architecture. It has no effect in the resulting architecture to exclude any needs or creating additional ones.

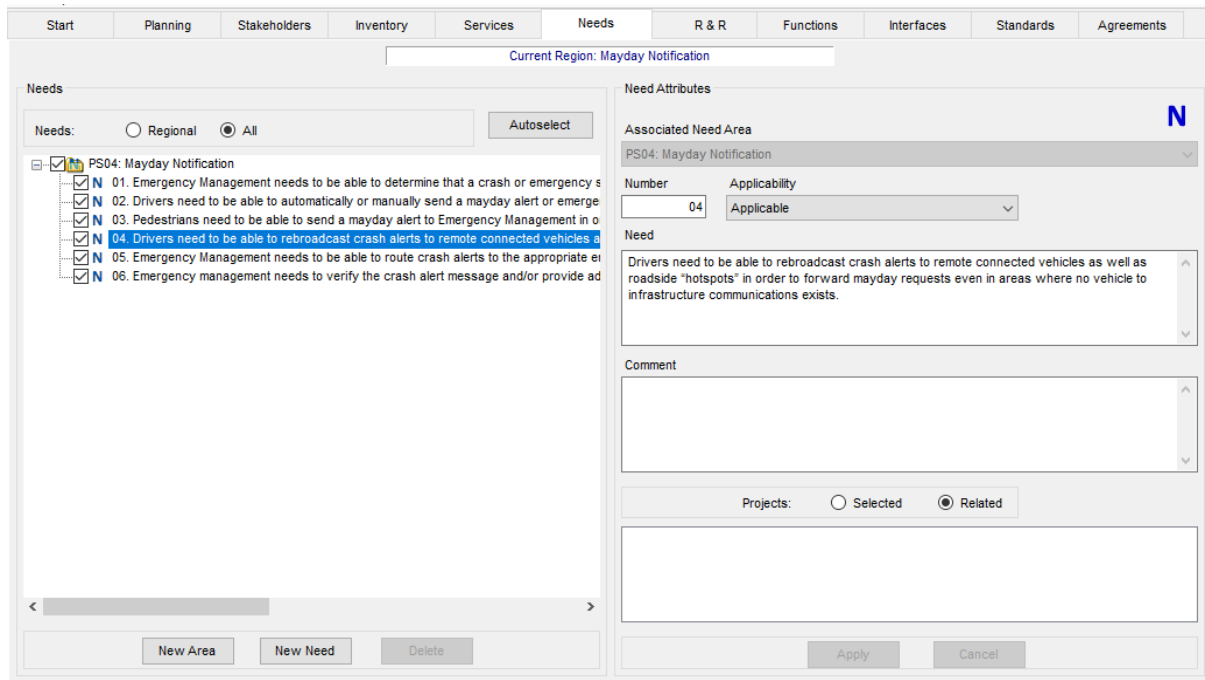


Figure 51 - Needs related to selected Service Package (source: author)

In the next step user shall define Roles and Responsibilities that would populate the Organizational View (see Figure 52). User is expected to to populate their own Role definitions and assign the stakeholders who will be Responsible for each Role.

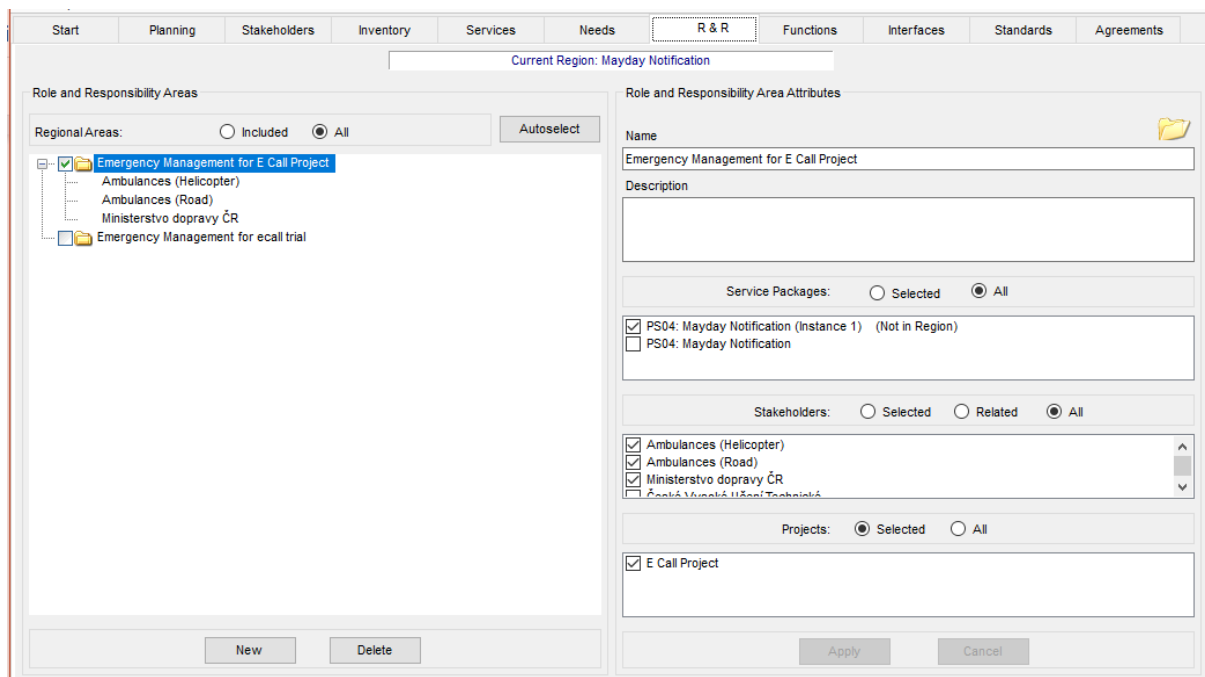


Figure 52 - The Roles and Responsibilities Tab (source: author)

The architecture creation continues with the Functions Tab where the user will be assigning Functional Objects to the elements (see Figure 53). Note that it is only possible to assign functions to elements which are classified as Physical Objects. The user can select the Functional Objects that are defined for the Service Packages that they have included to their architecture from the list or they can use the “Autoselect” function of Rad-IT. The user will be automatically offered the Functional Objects that are related with the Service Packages they have selected in the Services Tab.

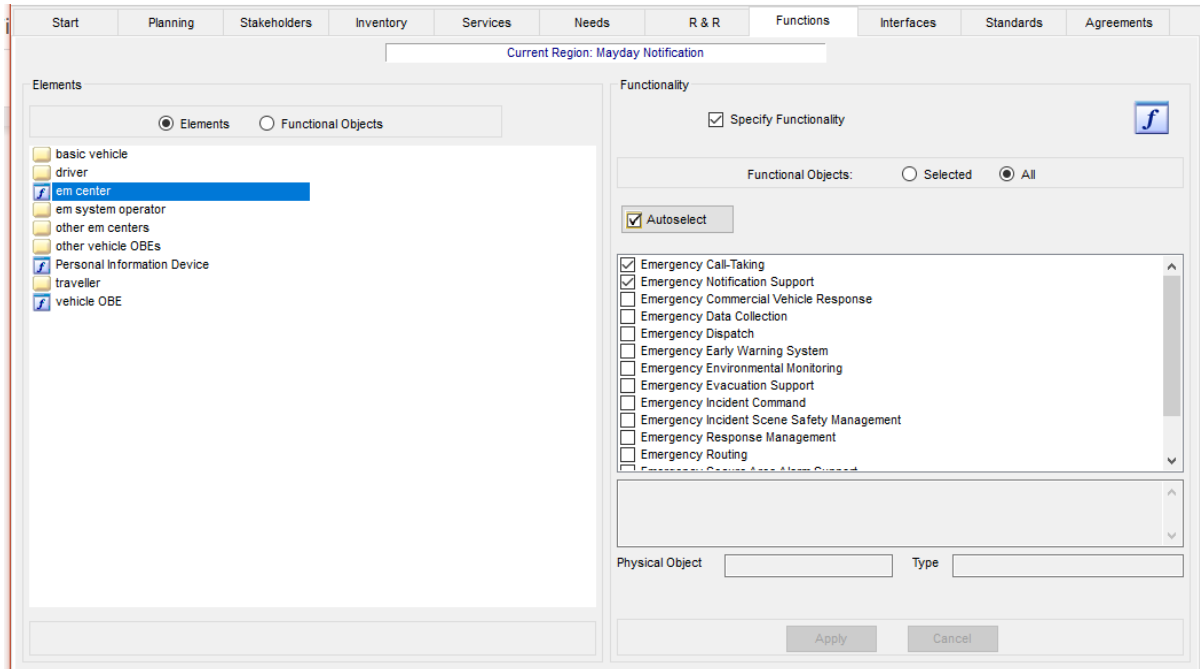


Figure 53 - The Roles and Responsibilities Tab

The next tab in Rad-IT is the Interfaces Tab where the interaction between Physical View elements is defined and also the Communications View is initiated. The architecture user shall define interfaces in two parts. In the Connect part the user will be defining the interconnects that need to be established between the Physical Objects. In the Flow part the user defines the Data Flow that connect Functional Objects. The Data Flows between Functional Objects residing in different Physical Objects will be passing through the interconnects.

We continue selecting all 8 interconnect that are necessary for the system described in Mayday Notification Service Package (see Figure 54).

| Start | Planning | Stakeholders | Inventory | Services | Needs | R & R | Functions | Interfaces | Standards | Agreements |
|--|-----------------------------|--------------|-----------|----------|---------|---------|-----------|------------|-----------|-------------------------------------|
| Mayday Notification: All Interconnects (8 Entries) | | | | | | | | | | |
| Element | Element | Element | Element | Element | Element | Element | Element | Element | Element | Include |
| basic vehicle | vehicle OBE | | | | | | | | | <input checked="" type="checkbox"/> |
| driver | vehicle OBE | | | | | | | | | <input checked="" type="checkbox"/> |
| em center | em system operator | | | | | | | | | <input checked="" type="checkbox"/> |
| em center | other em centers | | | | | | | | | <input checked="" type="checkbox"/> |
| em center | Personal Information Device | | | | | | | | | <input checked="" type="checkbox"/> |
| ▶ em center | vehicle OBE | | | | | | | | | <input checked="" type="checkbox"/> |
| other vehicle OBEs | vehicle OBE | | | | | | | | | <input type="checkbox"/> |
| Personal Information Device | traveller | | | | | | | | | <input checked="" type="checkbox"/> |

Figure 54 - Selecting Interconnects (source: author)

All possible information flows between Physical Objects are listed in the Flow part. The information flows that are related to interconnects that were included to architecture in the last step are already included here. We continue without making any selection in this part and confirm to include 24 Information Flows to our example architecture (see Figure 55).

| Start | Planning | Stakeholders | Inventory | Services | Needs | R & R | Functions | Interfaces | Standards | Agreements |
|--|------------------------------|-----------------------------|----------------|-------------------------------------|-------|-------|-----------|------------|-----------|------------|
| Mayday Notification: All Architecture Flows (24 Entries) | | | | | | | | | | |
| Source Element | Flow Name | Destination Element | Status | Include | | | | | | |
| em center | emergency operations status | em system operator | Planned | <input checked="" type="checkbox"/> | | | | | | |
| em center | incident report | other em centers | Planned | <input checked="" type="checkbox"/> | | | | | | |
| em center | emergency acknowledge | Personal Information Device | Planned | <input checked="" type="checkbox"/> | | | | | | |
| ▶ em center | emergency acknowledge | vehicle OBE | Planned | <input checked="" type="checkbox"/> | | | | | | |
| em center | emergency data request | vehicle OBE | Planned | <input checked="" type="checkbox"/> | | | | | | |
| em system operator | emergency operations input | em center | Planned | <input checked="" type="checkbox"/> | | | | | | |
| other em centers | incident report | em center | Planned | <input checked="" type="checkbox"/> | | | | | | |
| Personal Information Device | emergency notification | em center | Planned | <input checked="" type="checkbox"/> | | | | | | |
| Personal Information Device | traveler interface updates | traveller | Planned | <input checked="" type="checkbox"/> | | | | | | |
| traveller | traveler input | Personal Information Device | Planned | <input checked="" type="checkbox"/> | | | | | | |
| vehicle OBE | driver update information | basic vehicle | Planned | <input checked="" type="checkbox"/> | | | | | | |
| vehicle OBE | driver updates | driver | Planned | <input checked="" type="checkbox"/> | | | | | | |
| vehicle OBE | emergency notification relay | em center | Planned | <input checked="" type="checkbox"/> | | | | | | |
| vehicle OBE | emergency notification | em center | Planned | <input checked="" type="checkbox"/> | | | | | | |
| other vehicle OBEs | emergency notification relay | vehicle OBE | Not Applicable | <input type="checkbox"/> | | | | | | |
| other vehicle OBEs | emergency acknowledge | vehicle OBE | Not Applicable | <input type="checkbox"/> | | | | | | |
| other vehicle OBEs | emergency notification | vehicle OBE | Not Applicable | <input type="checkbox"/> | | | | | | |
| vehicle OBE | emergency notification relay | other vehicle OBEs | Not Applicable | <input type="checkbox"/> | | | | | | |
| vehicle OBE | emergency notification | other vehicle OBEs | Not Applicable | <input type="checkbox"/> | | | | | | |
| vehicle OBE | emergency acknowledge | other vehicle OBEs | Not Applicable | <input type="checkbox"/> | | | | | | |

Figure 55 - Selecting Information Flows (source: author)

As mentioned in the introduction, one of the main purposes for developing and using ITS architectures is to consider common standards on similar interfaces across the architecture. The Standards in the Standards tab address the interfaces between ITS elements and provides descriptions for the Communications View of the architecture (see Figure 56). Most of the

Information Flows defined in Arc-IT is already cross referenced to the standards related to specific type of communication. Upon determining the information flows between elements the user is presented the applicable standards to the information flows they have included in their architecture. The standards are defined for information flows instead of the interfaces. Common standards are grouped into profiles to be able present them in a simple way. Architecture user is provided with a report that they can generate that is listing applicable standards for each information flow to further investigate how the Communication View is organized. Standards are identified for almost all of the information flows with few exceptions. Rad-IT provides the capability to define new standards, or copy or modify existing ones to architecture user.

| Start | Planning | Stakeholders | Inventory | Services | Needs | R & R | Functions | Interfaces | Standards | Agreements |
|---|-------------------------------------|-----------------------------|--|--------------------------------|--------------------------|-------------------------------------|-----------|------------|-----------|------------|
| Mayday Notification Standards (10 Entries) | | | | | | | | | | |
| | Group | Group/Doc ID | Title | SDO | User Defined | Include | | | | |
| ▶ | <input checked="" type="checkbox"/> | DSRC-WSMP | Vehicle-to-Vehicle/Infrastructure using WSMP | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | RSEGateway-VehicleDestin... | Vehicle Communications via RSEs, Vehicle Destination | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | RSEGateway-VehicleSource | Vehicle Communications via RSEs, Vehicle Source | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | Vehicle-On-Board | Vehicle-On-Board | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | WAW-ASN1 | Wide Area Wireless using ASN.1 as encoding method | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | WAW-WWWBrowser-JSON | Wide Area Wireless using JSON as encoding method | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input checked="" type="checkbox"/> | WAW-XML | Wide Area Wireless using XML as encoding method | Profile | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input type="checkbox"/> | SAE J2313 | On-Board Land Vehicle Mayday Reporting Interface | Society of Automotive Engin... | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input type="checkbox"/> | SAE J2735 | Dedicated Short Range Communications (DSRC) Message Set Dictionary | Society of Automotive Engin... | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |
| | <input type="checkbox"/> | SAE J3067 | Candidate Improvements to Dedicated Short Range Communications (DSRC) Message Set Dictionary [S... | Society of Automotive Engin... | <input type="checkbox"/> | <input checked="" type="checkbox"/> | | | | |

Figure 56 - The Standards Tab (source: author)

The final step in creating Regional Architecture with Arc-IT is the Agreements Tab where the architecture user to further develop the Organizational View. Similar to the interfaces of a system, the agreements are interfaces between agencies and organizations who are stakeholders of the system(s). Agreements define how a stakeholder interacts with other stakeholders, what service is provided or acquired, which information need to be exchanged and how this exchange will be done, and the roles and responsibilities of the stakeholders. The expected terms of agreement are security, budget, scope, boundaries, standards, information exchange formats. Arc-IT provides many different types of agreements such as formal agreements, financial agreements, operational agreements and many other types that differ in their scope or complexity. A Lead stakeholder can be assigned to agreements wherever necessary and these may be from all list of stakeholders even they are not part of the agreement. Architecture user may populate their own agreements in addition to agreements that would be suggested by Rad-IT. If architecture user utilizes the Autoselect function Rad-IT provides agreement suggestions based on the information flows created between Physical Objects (see Figure 57). Each Physical Object was assigned a stakeholder as their owner in the Inventory Tab. Based on this assignment Rad-IT suggests that an agreement is needed when Physical Objects owned by different stakeholders exchange information.

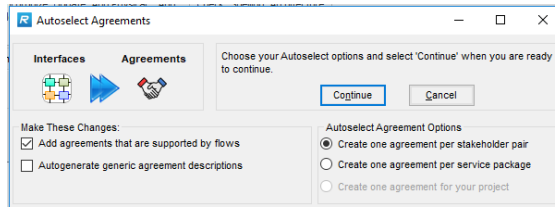


Figure 57 - Autoselecting agreements (source: author)

We proceed with adding agreements based on the information flows in the architecture. There were three Physical Objects defined in our example architecture, an emergency management centre owned by Emergency Services Association stakeholder, Vehicle OBE owned by the drivers stakeholder and Personal Information Device owned by the travelers stakeholder. Two information provision agreements were offered by Rad-IT based on the information flow between these Physical Objects, one between Drivers and Emergency Services Association and one between Travellers and Emergency Services Association (see Figure 58).

| Recommended Agreement Additions | | |
|---------------------------------|---|---------------------------------|
| # | Title | Type |
| 1 | Emergency Services Association Drivers Information Provision Agreement | Information Provision Agreement |
| 2 | Emergency Services Association Travellers Information Provision Agreement | Information Provision Agreement |

Figure 58 - Recommended agreements based on Information Flows (source: author)

Rad-IT recommends a generic “blanket” agreement assigned to the service package where architecture user populate agreements that bind all stakeholders defined for the service package in the Services Tab, such as operating procedures, non-disclosure agreements etc. (see Figure 59).

| Recommended Agreement Additions | | | |
|---------------------------------|---------------------------------------|-----------------------------------|----------------------------------|
| # | Title | Type | Description |
| 3 | Mavdav Notification Blanket Agreement | Service Package Blanket Agreement | This is a blanket agreement asso |

Figure 59 - Recommended agreements based on Service Packages (source: author)

A total of 3 agreements are offered by Rad-IT for our example architecture (see Figure 60). Architecture user may continue defining agreements, such as the user agreement with the vehicle owner or the agreements with GSM operator for our example to complete defining the Organizational View.

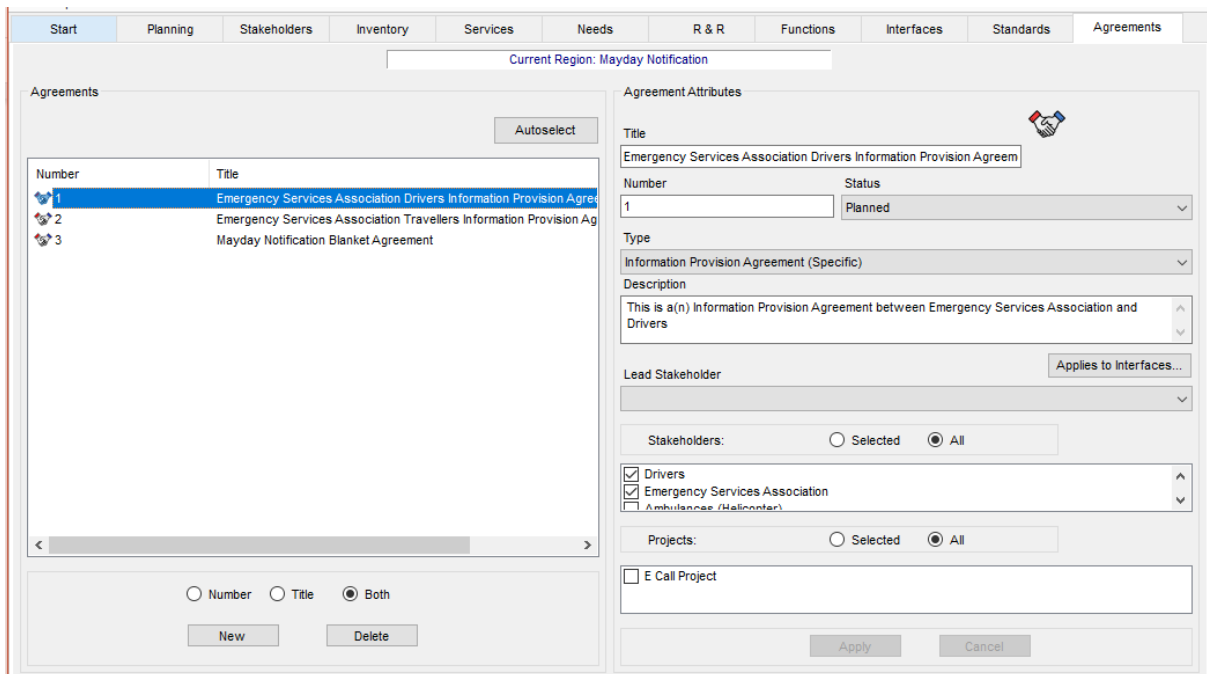


Figure 60 - Agreements Tab (source: author)

There are many artefacts offered with creation of a Regional architecture in Rad-IT that can be accessed from the Output ribbon at the top of the screen (see Figure 61). These artefacts are few diagrams representing Physical and Communication Views, a variety of detailed tables, a document describes the regional architecture carrying the information specified and populated by the architecture user from all eleven tabs of Rad-IT to a Microsoft Word template document and a Web Page created in htm or html format that conveys all information of the architecture tabs in a format that user can easily navigate between parts of the architecture and access to the tables linked to the website.

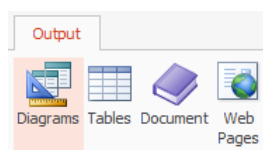


Figure 61 - The artefacts that can be produced with Rad-IT (source: author)

The first diagram offered to the architecture user is the Subsystem diagram where the Physical Objects in the architecture and the Communication links between them are displayed according to their classes. Below is the Subsystem diagram generated for our example architecture (see Figure 62).

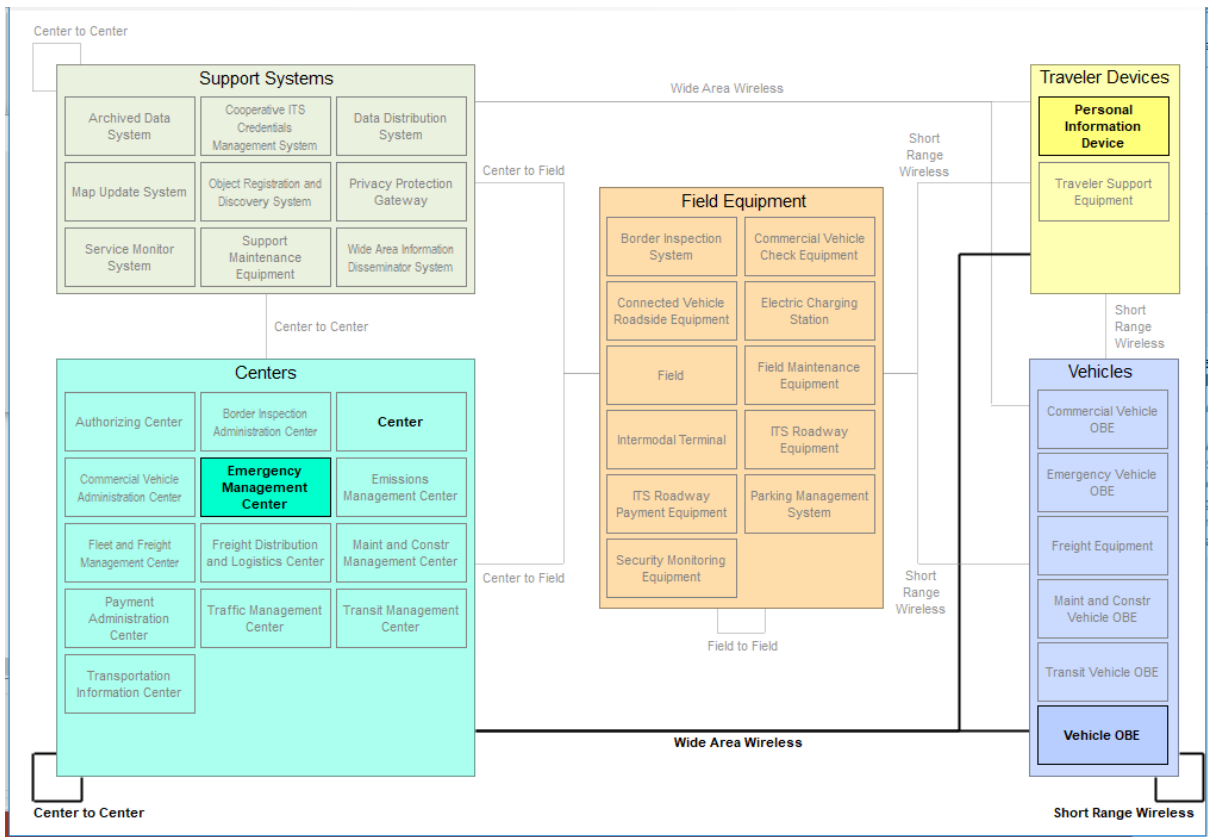


Figure 62 - The Subsystem Diagram (source: author)

The Interconnect diagram is a high level representation of the information flow between Physical Objects. Below is the interconnect diagram produced for our example architecture (see Figure 63).

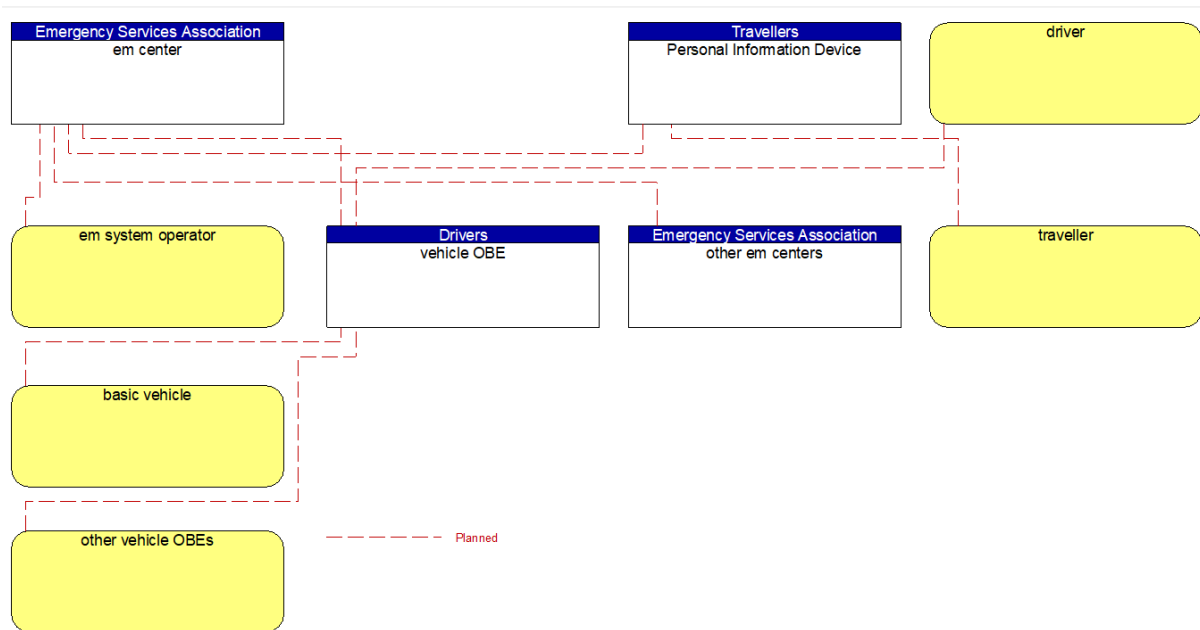


Figure 63 - The Interconnect Diagram (source: author)

The final diagram that the architecture user is provided is the Flow diagram where all Information Flows between the architecture elements are represented. Below is the Flow diagram for our example architecture (see Figure 64).

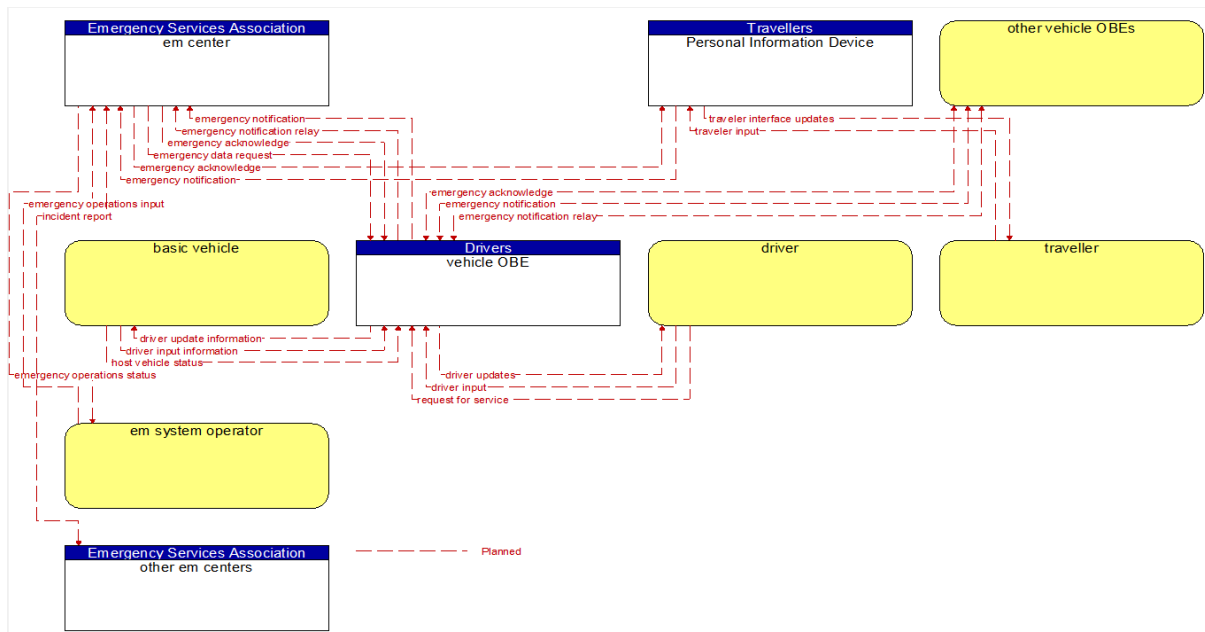


Figure 64 - The Flow Diagram (source: author)

The architecture user is offered a list of tables where they can produce tabulation related to general and administrative information related to architecture, specific information related to stakeholders, the interaction between architecture elements and many more information defined throughout the architecture tabs along with tables created to check the logical consistency of the architecture (see Figure 65). The tables can be produced in Word, Excel or .csv formats.

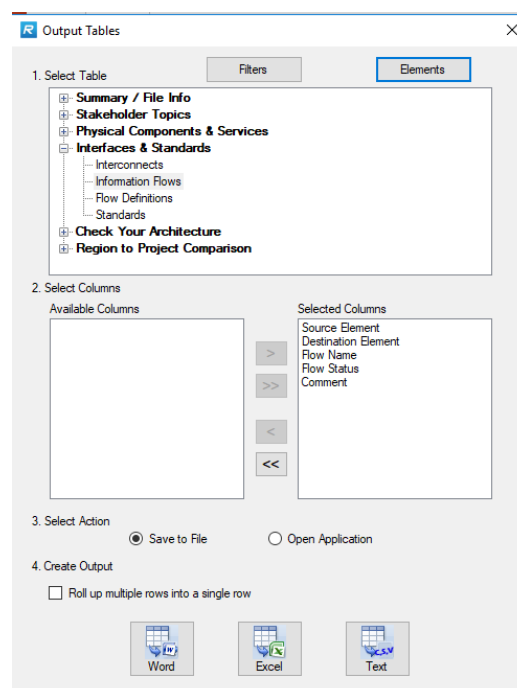


Figure 65 - The Table outputs of Rad-IT (source: author)

In addition to the Diagrams and Tables, Rad-IT offers its user the ability to generate a document for their architectures (see Figure 66). This document carries all information selected or populated throughout the tabs of Rad-IT to a template Microsoft Word document contains basic definition and explanation of the topics and fields in the architecture along with the information populated by the user and provides ability for user to access all information in one place. The document then can be further modified to contain logos, images and additional text that the user decides to include to their architecture. All available output tables that the user can generate in previous section are generated and included automatically in the document for further convenience of the architecture user.

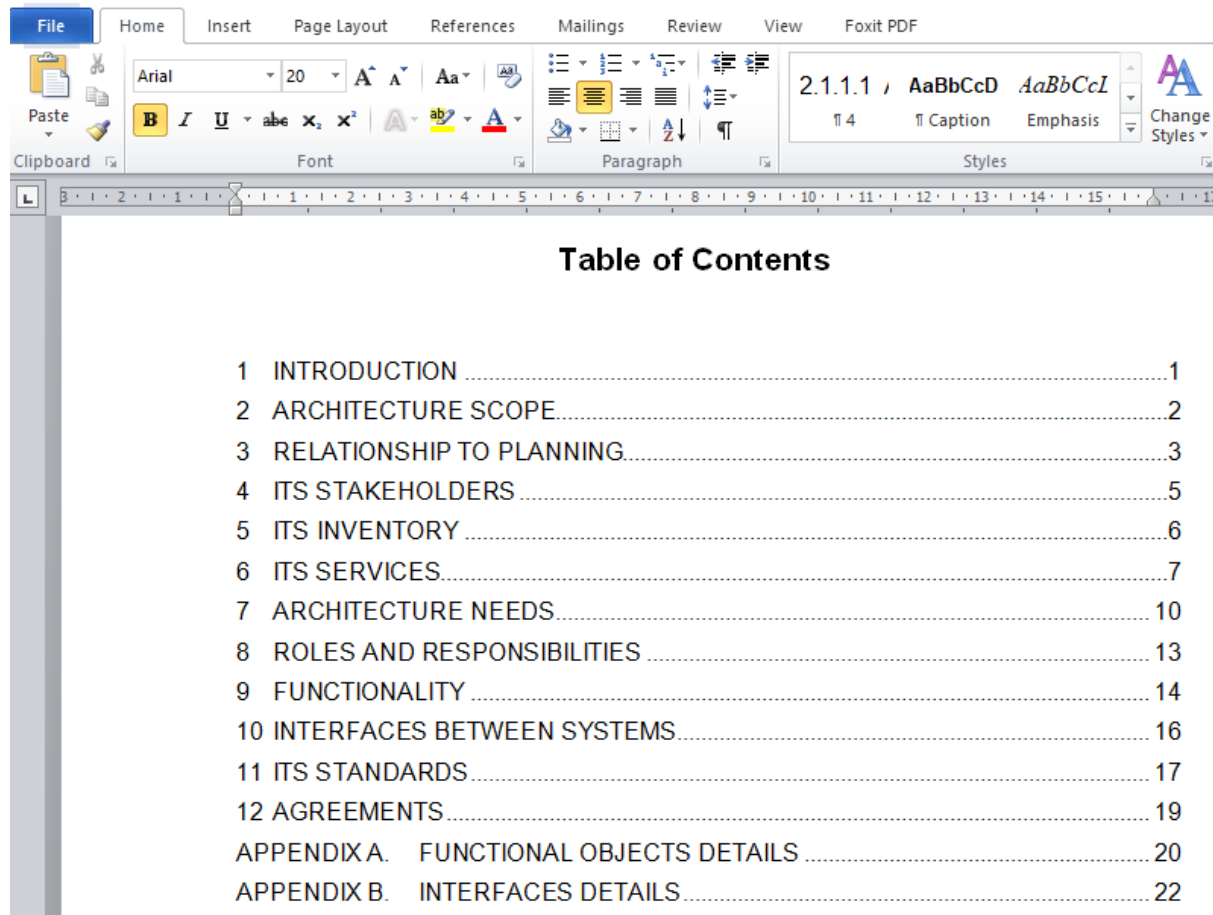


Figure 66 - The Table of Contents of the output document (source: author)

Finally the user can export the architecture to .htm or .html format that can be uploaded to internet as a Web page (see Figure 67). The full content of the architecture document that was explained previously is carried to this format and organized into sections that are linked to each other to simplify the presentation and navigation through the architecture. Below is the first screen of the Web page output for our example architecture.

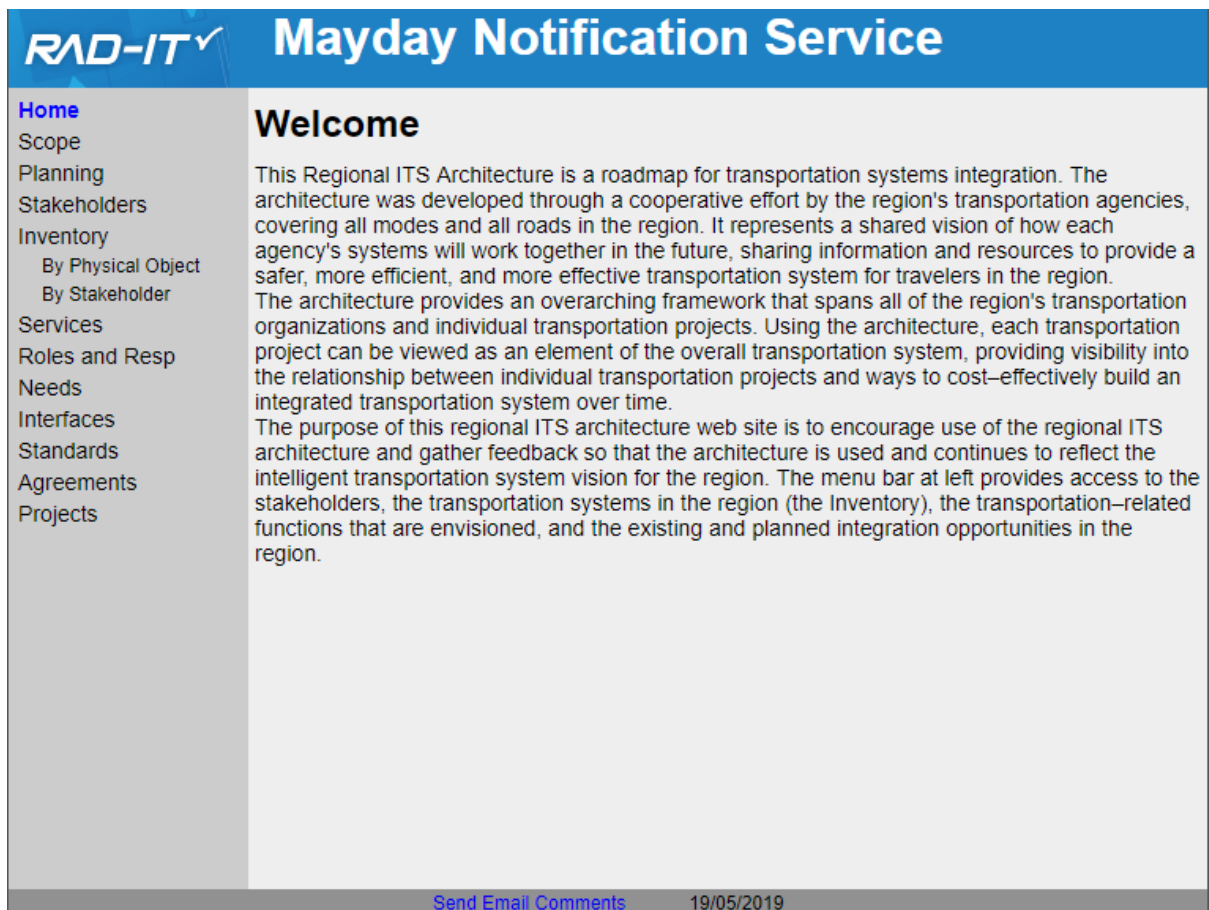


Figure 67 - The Web Page output of Arc-IT (source: author)

The user is expected to continue developing lower level Project Architectures for the projects defined in their Regional Architectures with the computer based System Engineering Tool for Intelligent Transportation offered with Arc-IT, which will not be covered with this paper.

Chapter 4 - A New Tool for Modelling the Architectures

Chapter 4.1 - Introducing Enterprise Architect

Enterprise Architect is a repository based modelling software which enables user to benefit from a wide range of state of the art modelling standards such as UML, BPMN, SysML and more. Enterprise Architect supports architecture frameworks such as TOGAF, Archimate and others. In addition to building models the software aims to assist its user in requirement analysis, simulation, testing and documenting their architectures. With the repository based approach the tool offers more than visual representation, the user is able to generate multiple views for their models using the same repository elements and any change done for an element in a single representation can be reflected in all models that the element is being used. Supporting many high end features for architecture modelling the tool is widely used in many industries. Since it is also the preferred tool for use by ISO TC204 and CEN TC278^[26] for the development of ITS related standards, Enterprise Architect was selected to be the tool in our proposal for improving FRAME. The Architectures for both the eCall and the Mayday Notification Service Package will be visualized and a Model Library based version of the Selection Tool will be created using Enterprise Architect software in the coming sections of this paper.

Chapter 4.2 - Describing eCall on Enterprise Architect

We decided to describe the eCall architecture that we have created with the FRAME Selection Tool in Enterprise Architect. We will be using Data Flow Diagrams to visualize our models. The High and Low Level Functions will be represented by Processes, Terminators with External type elements, Data Stores will be Data Stores and the interaction of the elements will be represented with Data Flows. The Processes are represented as a circle, the External elements are rectangular, the Data Stores are two parallel lines with some space between them and the Data Flows are represented as arrows going from one element to another and the direction of the flow is emphasized with the arrow heads on the Data Flows.

We started describing the system in the highest level and created the Context Diagram to display the system's interaction with the Terminators, the External elements which reside outside of the system boundaries (see Figure 68). In this level of representation we see the whole system depicted as a single process, the circle written "ECall" on it. The chain symbol on the process notifies that the process is further linked to another Data Flow Diagram, which would be representing inside the system boundaries in this case. By allowing to nest the diagrams in such way Enterprise Architect supports the system depicted in an hierarchical structure, where several levels of representation of the processes is possible. Data Flows that connects the system with the External elements are represented in detail, depicting each piece of information that is exchanged rather than clustering them in to interfaces, such as "to Emergency Systems" or "from Emergency Systems". This is because the Enterprise Architect does not support the Data Flows to be hierarchically nested into parent and sub Data Flows. Please note that the Data Flows shown to be connecting a Terminator and the system in this level of representation does not contain the information of which Low Level Function is related with the Data Flow. The Data Flows are represented to be going in and out of the system for illustration purposes only. These Data Flows represent the interfaces that need to exist between the system and its Terminators and each Data Flow need to pass through these interfaces. The true nature of the Data Flows that displays the exact Source or Destination Processes should be represented with lower level data flow diagrams.

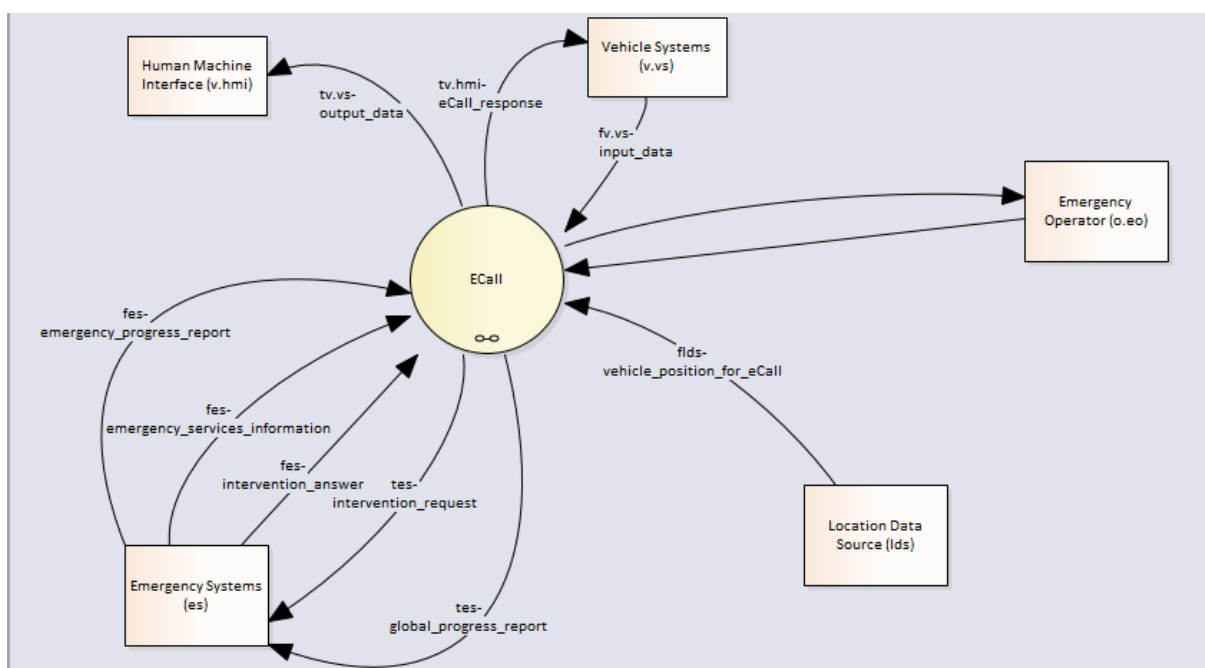


Figure 68 - eCall Example Context Diagram

We proceed to represent inside of the system boundaries with a lower level data flow diagram (see Figure 69). This diagram corresponds to the DFD0 for the FRAME Functional Viewpoint which displays the interaction between the Functional Areas defined for the system. First the diagram was created and then it was linked to the eCall Process that was represented in the Context Diagram. This linkage was represented with a chain symbol on the eCall Process and the lower level data flow diagram is accessible by double clicking on the parent Process in Enterprise Architect. The Functional areas are represented as Processes and similar linkages were established for the Functional Areas that link them with lower level data flow diagrams that display the system inside each Functional Area. Just as it is in the Context Diagram, the Data Flows between Functional Areas are illustrated to represent the interfaces that must be built between the Functional Areas but do not express the correct Source or Destination Low Level Functions. The relationship of Functional Areas and the Terminators are excluded from this diagram since the elements outside system boundaries are not represented in this level.

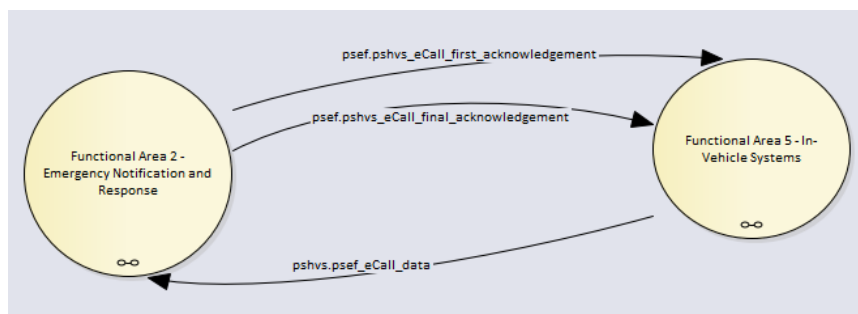


Figure 69 - DFD0 for eCall Example (source: author)

Inside the Functional Area 2 - Emergency Notification and Response is defined a single High Level Function depicted as a Process. The Process is represented on its own, without any interaction since there is no other High Level Function was selected for eCall and the functionality that interacts with other parts of the system is either in the lower level or the higher level representations of the system (see Figure 70).

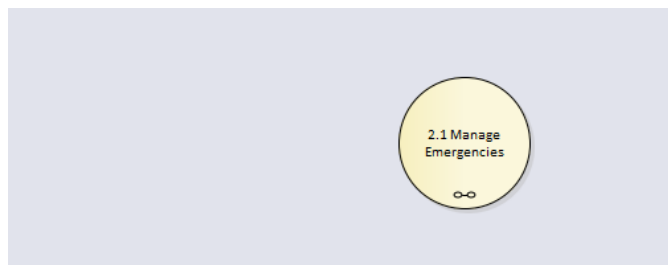


Figure 70 - DFD2 Emergency Notification and Response for eCall Example (source: author)

The High Level Function 2.1 - Manage Emergencies consists of a High Level Function, two Low Level Functions and a Data Store (see Figure 71). The Data Flows between the Low Level Functions and the Data Store represents the actual Data Flows with their correct source and destination elements. The High Level Function 2.1.2 - Manage Emergency Intervention is associated with further Low Level Functions and the relationships with higher level elements in the architecture are not displayed on this diagram.

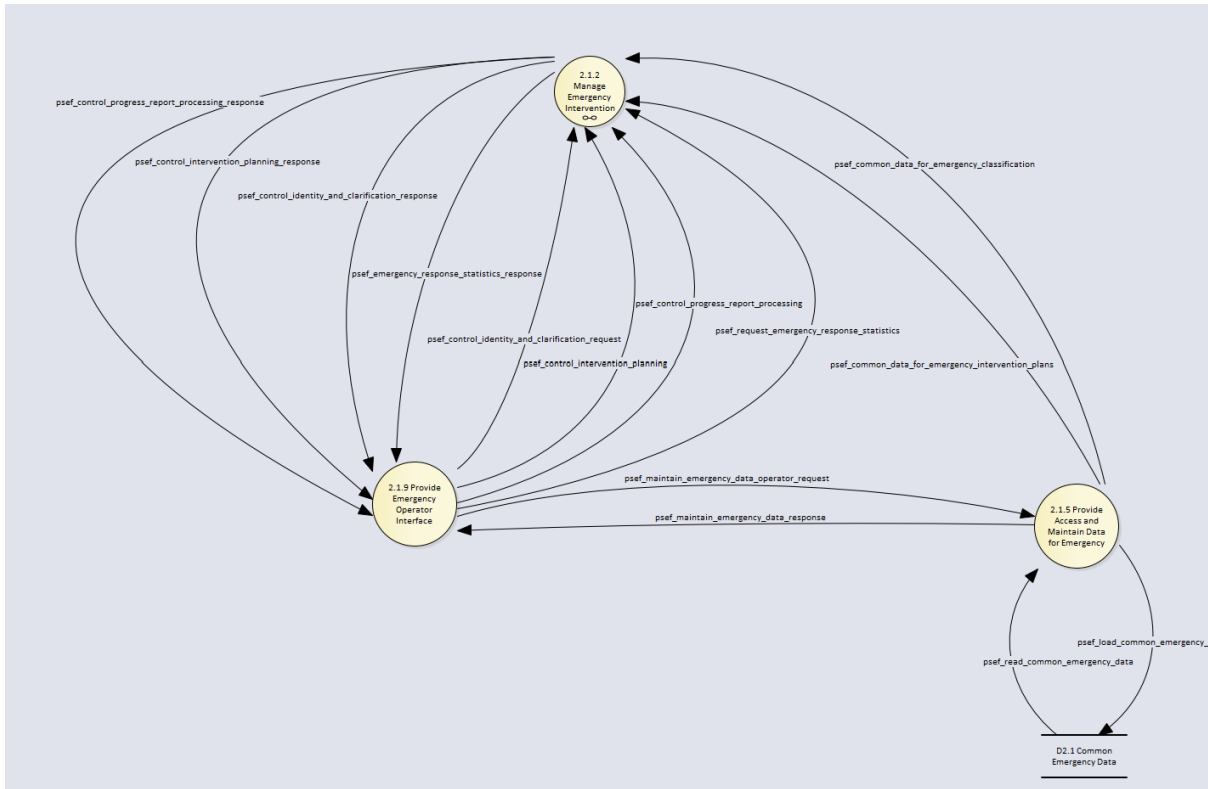


Figure 71 - The Data Flow Diagram representing the High Level Function 2.1 (source: author)

There are four Low Level Functions and a Data Store selected for High Level Function 2.1.2 Manage Emergency Intervention (see Figure 72). All of the Data Flows represented in the diagram below are the actual Data Flows defined in the architecture since this is the lowest level diagram related with the Process and the all Processes that are displayed are Low Level Functions. This diagram excludes the interaction with the rest of the Low Level Functions inside the Functional Area, with the Low Level Functions of the Functional Area 5 - In-Vehicle Systems or the Terminators of ECall since these elements are not represented in this level.

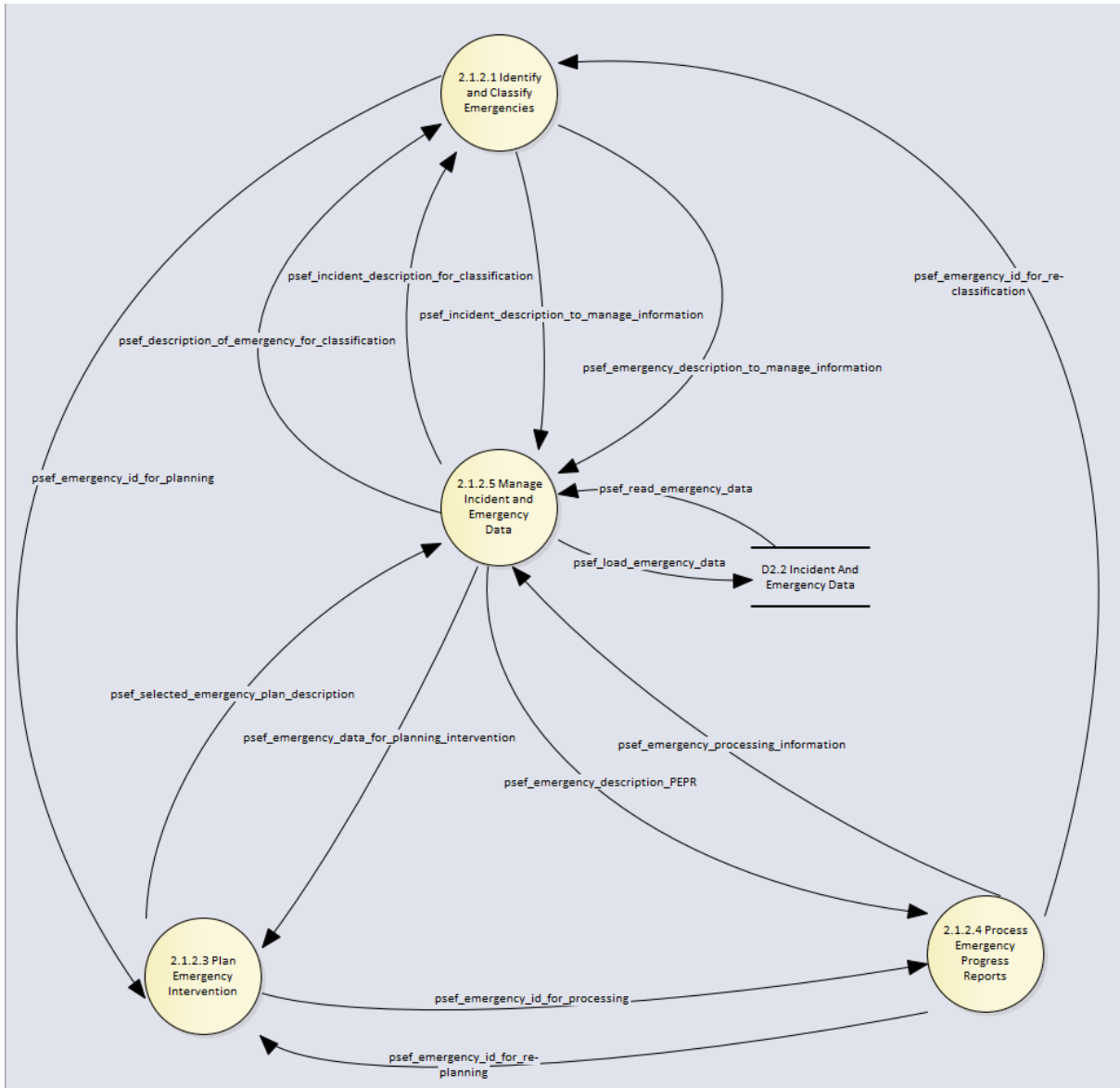


Figure 72 - Data Flow Diagram representing High Level Function 2.1.2 Manage Emergency Intervention (source: author)

We have summarized the Functional Area 5 - In-Vehicle Systems of the eCall example with the below chart (see Figure 73). The first square represents the High Level Functions inside the Functional Area and the interaction between them. Only one Low Level Function for each High Level Function was selected in the eCall example. All of the diagrams below exclude the relationship of the Processes in this Functional Area with the Processes in Functional Area 2 - Emergency Notification and Response and the Terminators of the system following the method hierarchical representation mentioned above.

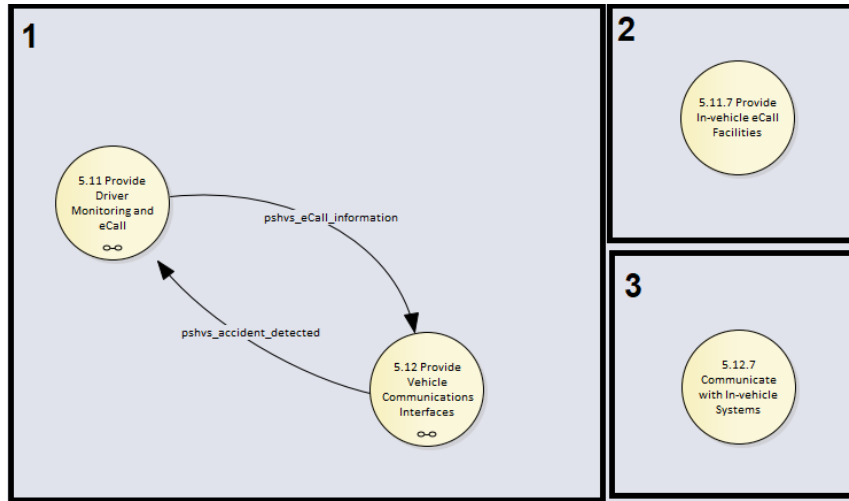


Figure 73 - The lower level Data Flow Diagrams in Functional Area 5 - In-Vehicle Systems of eCall Example (source: author)

Chapter 4.3 - Proposal: Modelling FRAME on Enterprise Architect

FRAME offers its user the ability to create an architecture for their projects with the Selection Tool and the user ends up having lists of elements they have included in their architecture at the end of the selection process, but not a visual representation of their architectures. Although the user has the complete lists of elements, this output is not enough to “clearly communicate” about the system(s) as it is aimed with having an architecture in the first place. The architecture user needs to transfer these otherwise complex lists to another tool and visually model the systems to be able to create a harmonious understanding of the systems by all parties that are involved with the system(s). We found that by using several functionality that was built in with the Enterprise architect it is actually possible to mimic the Selection Tool and the architecture creation process of FRAME without losing any details related to architecture in addition to access modern architecture products such as Data Flow Diagrams represent user’s selection and many more features offered available by the Enterprise architect. In this part we will model the part of FRAME that was related with the eCall example and propose a Library of FRAME User Needs and functions to be created in Enterprise Architect. We will demonstrate how the architecture user may benefit from such an architecture creation method via the eCall example. We will show how this methodology could be extended to cover all FRAME User Needs and Functional Viewpoint. Finally we will comment on how the Physical Viewpoint representation may be added to this method of creating ITS architectures.

To begin modelling the FRAME we first created a Data Flow Diagram that represents the Functional Viewpoint of the eCall example in its lowest level (see Figure 74). We have transferred all Low Level Functions, Data Stores and Terminators that was selected to Enterprise Architect and connected them with Data Flows that uses the lowest level source or destination elements, disabling the hierarchical representation of the Functional Viewpoint. This allowed us to relate the elements with each other correctly where each Data Flow represents the actual interaction that will take place between elements.

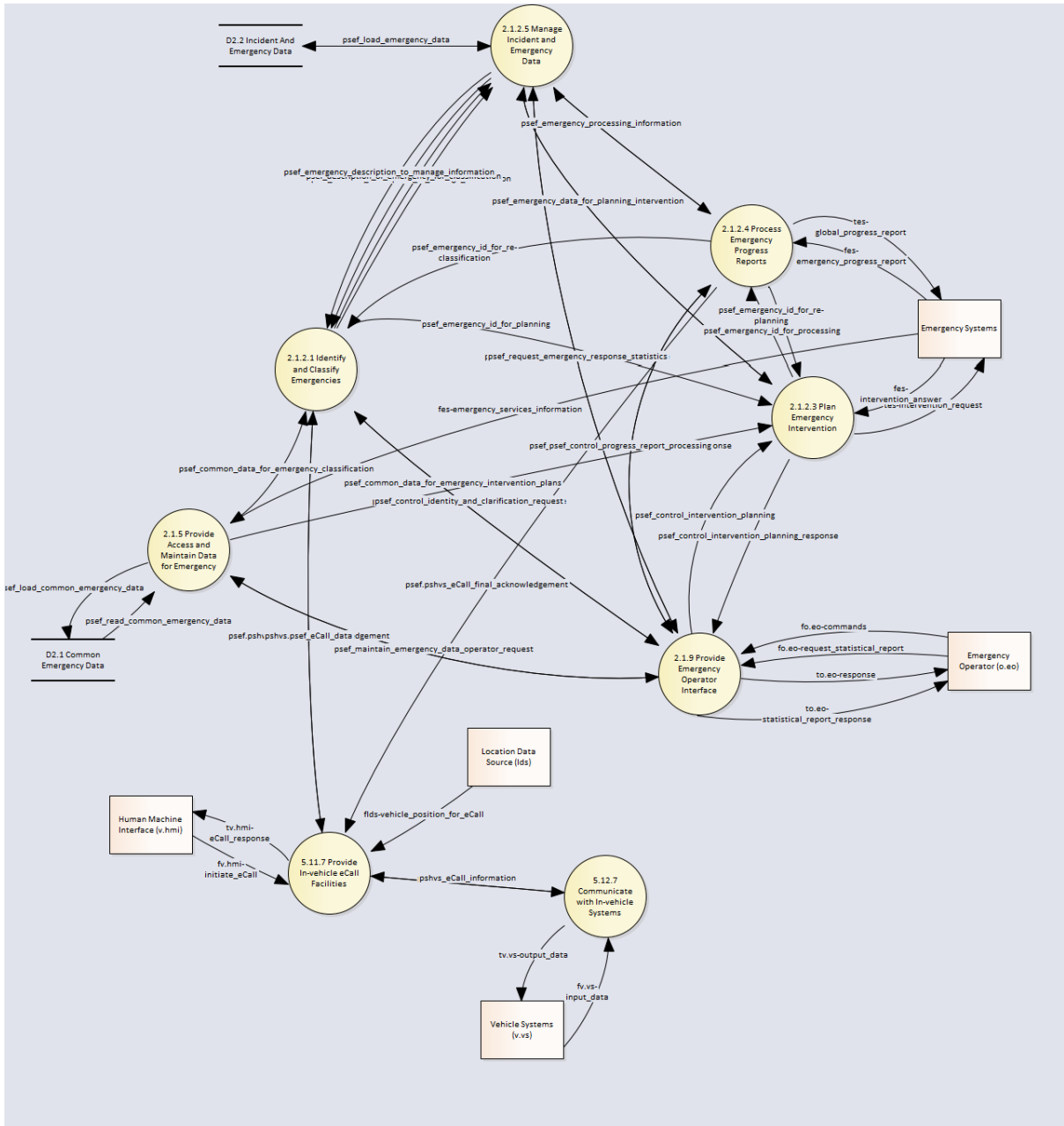


Figure 74 - Lowest Level Data Flow Diagram for eCall example. (source: author)

Connecting elements in Enterprise Architect defines “Relationships” between them that are coded to the connected elements (see Figure 75). Each Data Flow in the lowest level Data Flow Diagram above defines a relationship between the elements that they connect. These relationships may be traced back in any other diagram that would include the particular element since they are attached to the element as soon as they are established. These relationships will be the basis for automating the creation of the Functional View, which will be explained later in this section. The figure below displays the list of elements that have relationship with the Low Level Function 2.1.2.3 - Plan Emergency Intervention.

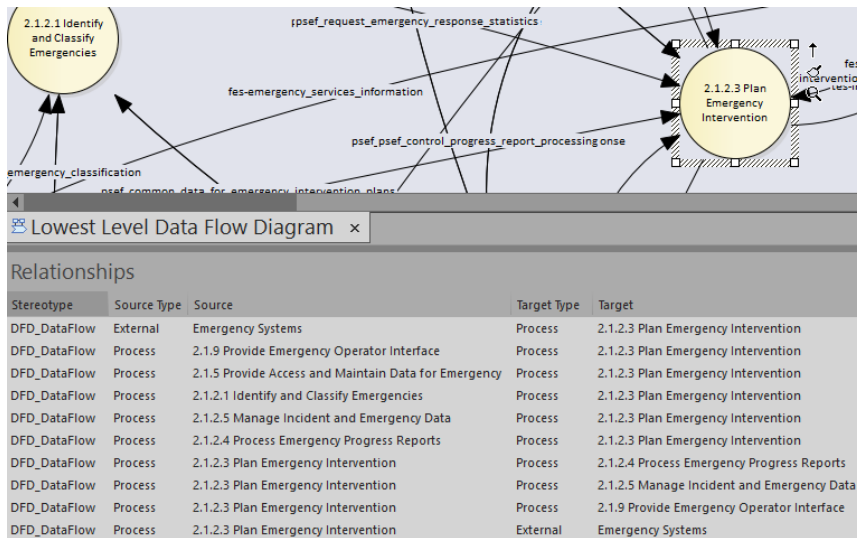


Figure 75 - List of elements that have relationship defined with the selected Low Level Function 2.1.2.3 - Plan Emergency Intervention (source: author)

We have created a library of elements that compose the Functional Viewpoint of the eCall example by listing the Low Level Functions, Data Stores and Terminators that were used in the lowest level Data Flow Diagram for the example (see Figure 76). This library will be used as a basis for the Subsequent Passes while creating a project architecture, which will explained later in this section.

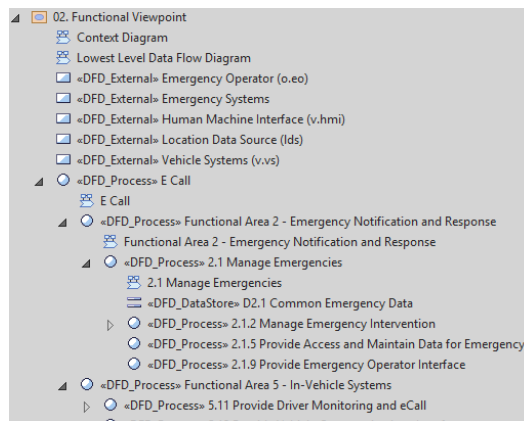


Figure 76 - The Functional Viewpoint library of eCall example (source: author)

Having the Functional Viewpoint library created we proceeded to create the list of User Needs that was selected in the eCall example. We used the Requirement Specification View Diagram in the Enterprise Architect that allows a text based editor to generate the requirements for an architecture. We have transferred the User Needs along with their descriptions to this diagram (see Figure 77).

| Item | Stereotype | Status | Difficulty | Priority |
|---|-----------------|----------|------------|----------|
| <input checked="" type="checkbox"/> 5.1.0.1 The system shall be able to make an 'eCall'. | UserRequirement | Proposed | Medium | Medium |
| <input checked="" type="checkbox"/> 5.1.0.2 The system shall be able to detect that the vehicle has been involved in an accident, identify its location, and initiate 'eCall' automatically. | UserRequirement | Proposed | Medium | Medium |
| <input checked="" type="checkbox"/> 5.1.0.3 The system shall enable the driver, or any other vehicle occupant, to make an 'eCall', and to receive confirmation that the call has been acknowledged, from outside the vehicle, i.e. at the roadside. | UserRequirement | Proposed | Medium | Medium |
| <input checked="" type="checkbox"/> 5.1.0.4 The system shall be able to give the driver an immediate acknowledgement to his/her emergency call, i.e. to indicate that assistance is on the way | UserRequirement | Proposed | Medium | Medium |
| <input checked="" type="checkbox"/> 5.1.0.5 The system shall be able to identify the driver / vehicle making an emergency call. | UserRequirement | Proposed | Medium | Medium |

Figure 77 - The Requirement Specification View Diagram for eCall example (source: author)

Each entry in the Specification View Diagram creates a requirements element in the Enterprise Architect Model Library. We have hierarchically organized these elements as it is in the User Needs of FRAME (see Figure 78). The library of User Needs will be used as an entry point in the First Pass of architecture creation, just as it was in the original Selection Tool of FRAME.

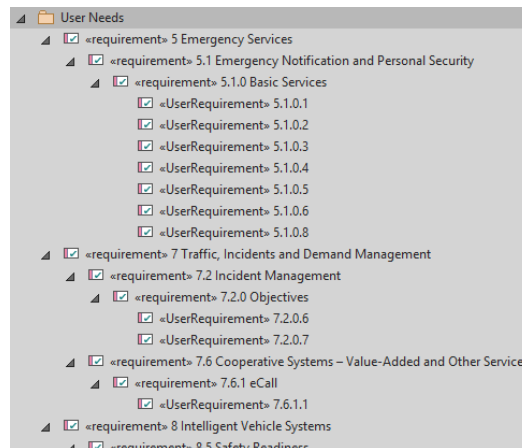


Figure 78 - The User Needs Library created for eCall example (source: author)

The User Needs that was included in the eCall example than were referenced to the Low Level Functions that they require, as it is in FRAME (see Figure 79). Relationships are defined between User Needs and Low Level Functions are the basis for the First Pass in FRAME, this will allow user to select the primary functionality for their architecture by selecting the User Needs.

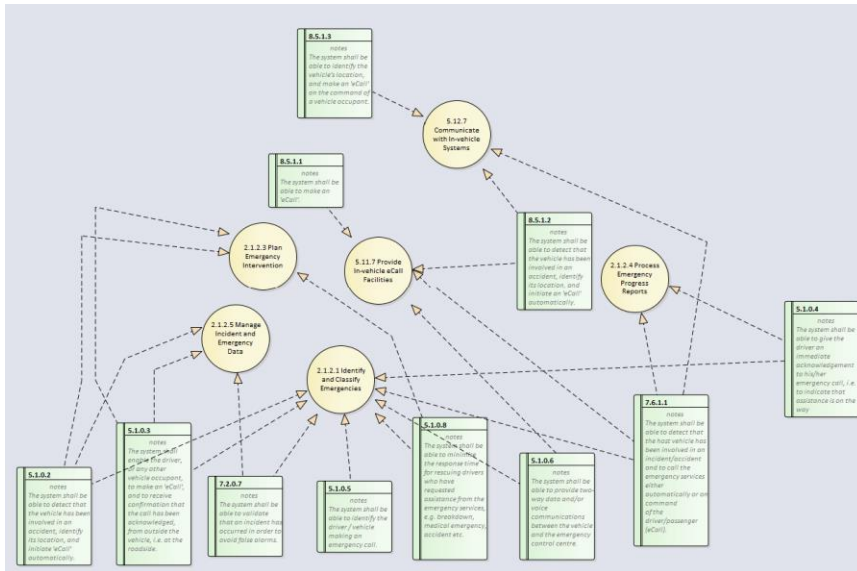


Figure 79 - Mapping User Needs to Low Level Functions (source: author)

The Functions that are mapped to User Needs can be traced back similar to the Data Flow relations between the Functional Viewpoint elements. Below is the list of User Needs those have relationship defined with Low Level Function 2.1.2.1 Identify and Classify Emergencies (see Figure 80).

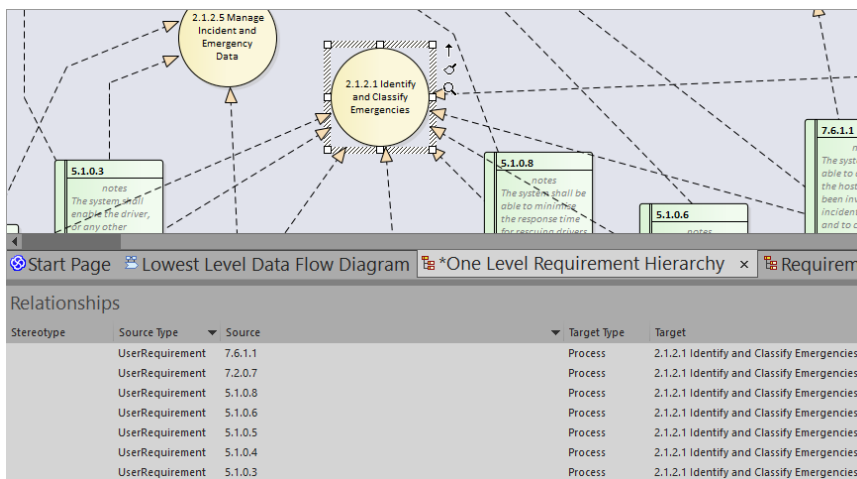


Figure 80 - User Need relationships with the Low Level Function 2.1.2.1 Identify and Classify Emergencies (source: author)

With creating the User Needs and Functional Viewpoint libraries we have finalized transferring FRAME elements to Enterprise Architect. We have defined relationships between each element of the Functional Viewpoint of the eCall example and also between the User Needs and the Functional Viewpoint. Now we will explain how to use this libraries and the relationships to automate the architecture creation and offer our method to replace the Selection Tool of FRAME.

We have added a blank Data Flow Diagram in our Model Library in Enterprise Architect and named it "Create Your Architecture". This blank diagram will be used by the architecture user to create their own architectures out of FRAME. User is expected to select the User Needs from the Model Library and "drag & drop" them to this blank diagram. When an element is inserted to a

diagram on Enterprise Architect, the elements that have a relationship defined with that can be imported to the diagram by using the “Insert Related Elements” functionality (see Figure 81). This feature of Enterprise Architect is the main factor of the architecture modelling method that we propose in this paper.

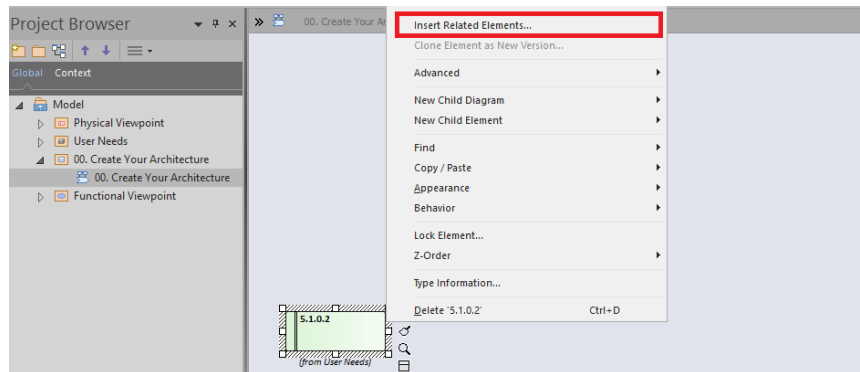


Figure 81 - Importing related elements to a diagram on Enterprise Architect (source: author)

When architecture user selects to import the elements in relationship with the User Need added to the diagram, “Insert Related Element” window pops up as it is displayed below (see Figure 82). In this window user can filter the type of elements that they would like to add to their architecture by using “Connector Type” and “Element Type” fields. The Low Level Function are “Realization” of the User Needs and the Functional Viewpoint elements those are connected via “DataFlow” connectors. After adding the desired User Need to their architecture user is expected to import the Low Level Functions related to the particular User Need.

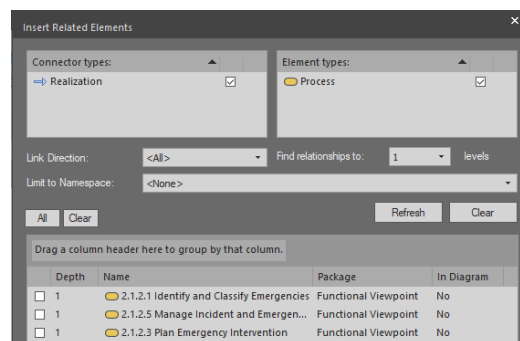
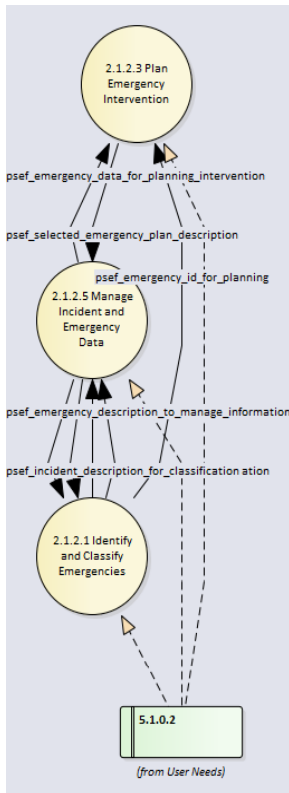
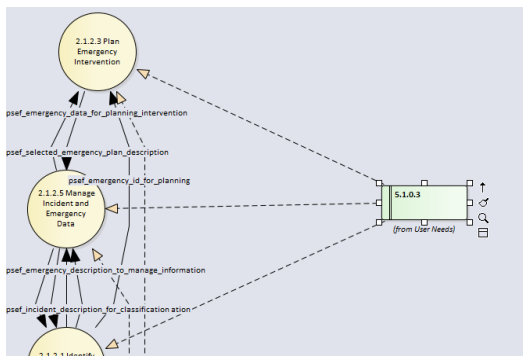


Figure 82 - Importing Low Level Functions that are related to the User Need 5.0.1.2 (source: author)



As soon as the architecture user selects the elements to be added to the diagram Enterprise Architect creates an automatic layout for the items that exist in the diagram as it is shown in the Figure 10 on the left (see Figure 83). It usually rearranges how the elements are distributed in the diagram to achieve the least complex and easiest to follow representation of what is already in the diagram. In our example the User Need 5.1.0.2 was related to three Low level Functions. We have selected all to be included in the diagram. Notice how the dashed arrows are pointing from the User Need towards the Low Level Functions. This is how a Realization relationship is depicted in the Enterprise Architect. The Low Level Functions added to the diagram carried all Data Flows between them since they were defined with DataFlow type of connectors in the first step, when we were creating the lowest level Data Flow Diagram. Since all relationships of elements are traceable in every diagram they are imported, Enterprise Architect automatically displays the relationships that exist between the elements that are already in the diagram. This feature helps to automate the architecture creation process since any element that is added to a diagram would automatically establish relations with the elements that exist in the diagram.

Figure 83 - The Data Flow Diagram after inserting the related elements with the User Need 5.1.0.2 (source: author)



We continue adding the User Needs that are necessary for the eCall example. Notice how adding User Need 5.1.0.3 automatically establishes the connections with the Low Level Functions that already exist in the diagram (see Figure 84). Since there are no other elements related with the specific User Need, the architecture user is not required to use Insert Related Elements feature, user is expected to continue adding User Needs.

Figure 84 - The Data Flow Diagram after inserting the related elements with the User Need 5.1.0.3 (source: author)

The architecture user should repeat the above steps for all User Needs they desire to add to their architecture. The Low Level Functions that already exist in the diagram will be automatically matched with the related User Needs. In case there are new Low Level Functions that are related with a User Need user is expected to add them using Insert Related Elements feature. The Low Level Functions that are added with new User Needs will automatically establish the Data Flows between the other Low Level Functions that already exist in the diagram. The Enterprise Architect carries those Data Flows from the Functional Viewpoint of the architecture that was created in the first step, the creation of the lowest level Data Flow Diagram.

The user will be creating the primary functionality of their system(s) by repeating above steps for each User Need that they desire to include with their architecture. Following the steps of Selection

Tool, now the user should investigate each Low Level Function included in their architecture and add any Data Store that is found to be related with the primary Low Level Function in the architecture. The architecture user can easily filter the Data Stores by the Element Type filter and add only the missing Data Store as it is displayed in the figure below (see Figure 85).

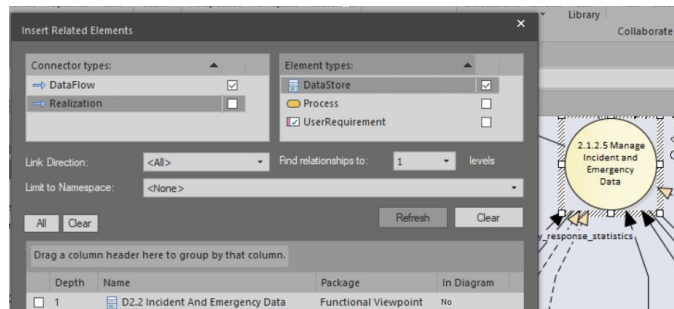


Figure 85 - Adding Data Stores to the architecture (source: author)

The architecture user is then expected to add the Terminators to their architectures. They can do so by filtering the External type of elements in addition to Data Stores with the last step. Since all Terminators that are related to the eCall primary functions already exist in the diagram we do not add any Terminators in this step.

By adding the Low Level Functions related to User Needs and the Data Stores and Terminators related to these functions the First Pass of the architecture creation is completed. The architecture user may continue with the Subsequent Passes by importing the elements from the Functional View set in the Model Library that was created with the first step and / or using the Insert Related Elements feature as all of the elements in the Functional View is already connected to one or more element by their nature.

We continue the Subsequent Pass by adding the two Low Level Functions that were not related with the User Needs by using the Insert Related Elements functionality (see Figure 86 & 87).

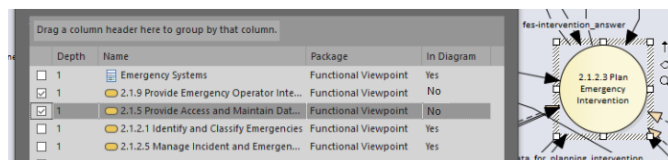


Figure 86 - Adding missing Low Level Functions to the eCall example architecture
We add the Data Store related with the secondary functions. (source: author)

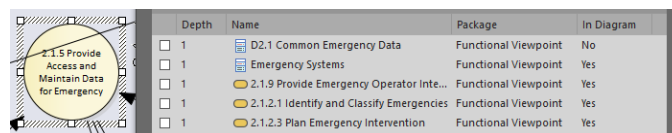


Figure 87 - Adding Data Stores related with the secondary functions of eCall example (source: author)

To complete creating the architecture for eCall example we add the missing Terminator (see Figure 88). This terminator was not added in the First Pass since it has no relation defined with the primary functions of the eCall example.

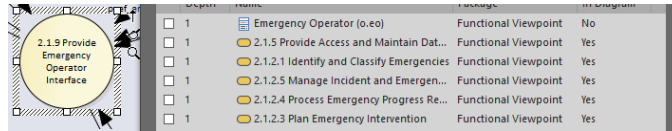


Figure 88 - Adding the missing Emergency Operator to the eCall example architecture (source: author)

After inserting each element that is necessary to completely describe the desired system(s), the Enterprise Architect automatically lays out the elements to the once “blank” diagram for the user. The Data Flow Diagram Below represents the resulting architecture that was created for the eCall example (see Figure 89).

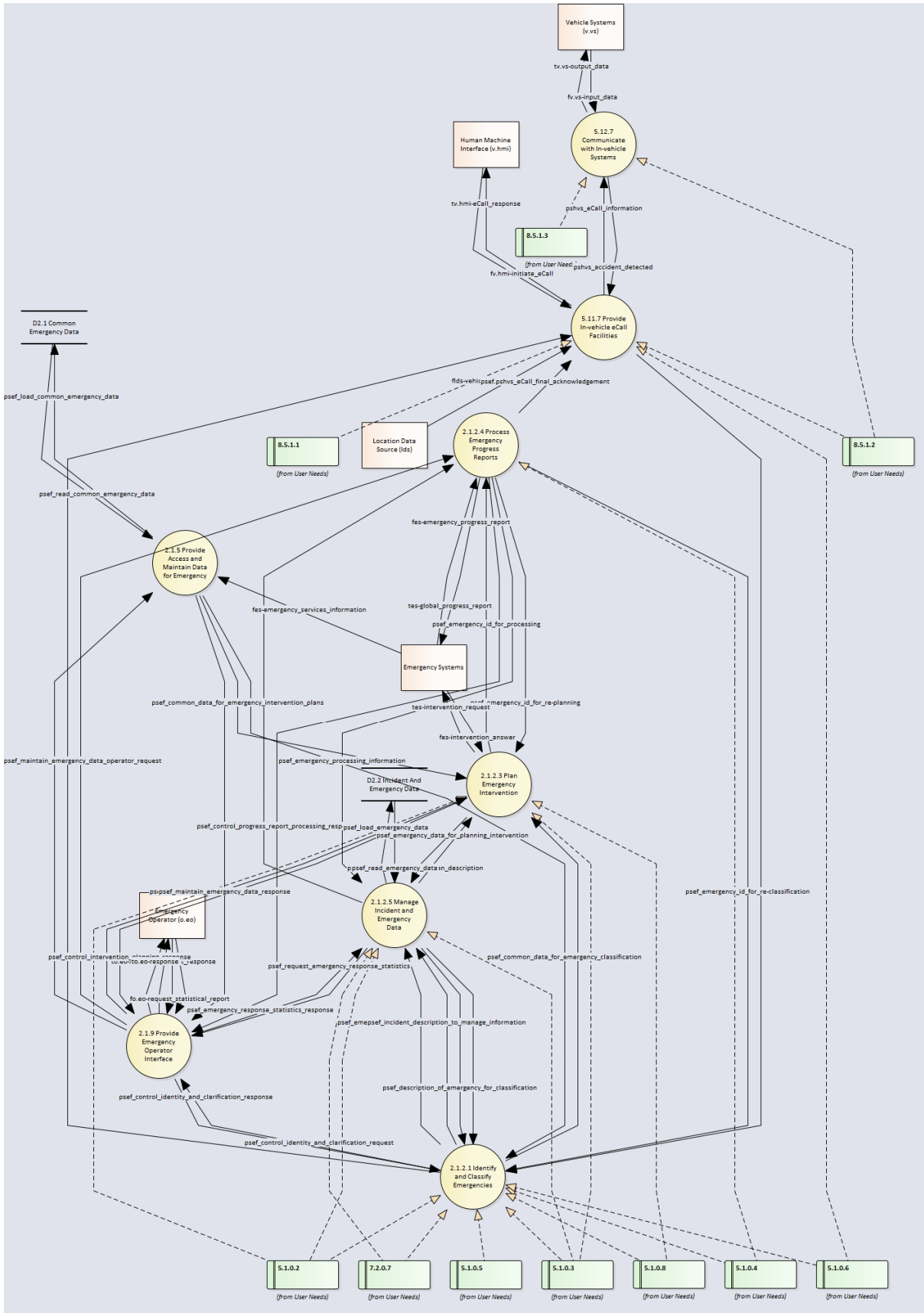


Figure 89 - The Data Flow Diagram showing the resulting architecture

Here we saw an architecture created for a small portion of ITS. If the complete FRAME Functional Viewpoint would be created with all Low Level Functions, Data Stores and the Terminators defined in the architecture and cross referenced to the list of all User Needs in FRAME this example can be extended to cover all FRAME, thus will be able to replace the Selection Tool entirely. Once such a Model Library that contains all User Needs and Functional Viewpoint is created, It would be sufficient for the user to have access to a copy of Enterprise Architect and download the FRAME Model Library from the FRAME website.

Chapter 4.4 - Experiences on describing The Mayday Notification Services in Enterprise Architect

We have transferred the Mayday Notification Service Package architecture that we have obtained by Rad-IT to Enterprise Architect to be able to have an understanding of how a project architecture could be made use of in case of a deployment scenario, in other words how the architecture user should be using the architecture description when they are actually aiming to build a system in real world. Our premise in doing so was that the architecture user would have to have a solid understanding of the resulting architecture, its elements and the relationships between those elements to be able to actually build the system(s) and trying to model the system(s) outside Arc-IT would require the same understanding. By doing so we would have a solid base to be able to compare the usability of architecture descriptions produced by FRAME and Arc-IT for a similar system.

Arc-IT builds the architecture for a system based on Service Packages, for each Service Package the Physical Objects that should exist in the system are defined with the Functional Objects that eventually will be conveying the required functionality for the system(s). To be able to compare the two ITS Architectures we tried to capture the Functional View of Arc-IT since the ultimate product for FRAME is the Functional Viewpoint.

We found out that the Functional View on Arc-IT was not supported as much as the other views in the architecture. While it is possible to trace interaction between Physical Objects to fulfil the functionality that is required for the systems, the interaction of the individual functions is not so visible. In other words the interaction of functions within a Physical Object, which gives the system its functionality, is not known to the architecture user. To elaborate on this issue we have analysed the Pspec 5.1.1.1 - Coordinate Emergency Input that is supposed to be included with the Emergency Call Taking Functional Object defined for the Mayday Notification Service Package (see Figure 90).

| Physical Object | Functional Object | PSpec Number | PSpec Name |
|-----------------------------|-----------------------|--------------|---|
| Emergency Management Center | Emergency Call-Taking | 5.1.1.1 | Coordinate Emergency Inputs |
| | | 5.1.1.3 | Collect Incident And Event Data |
| | | 5.1.2 | Determine Coordinated Response Plan |
| | | 5.1.3 | Communicate Emergency Status |
| | | 5.1.4 | Manage Emergency Response |
| | | 5.2 | Provide Operator Interface for Emergency Data |

Figure 90 - Some of the Pspecs that were defined for the Mayday Notification Service Package (source: Arc-IT website)

The Pspec 5.1.1.1 - Coordinate Emergency Inputs were defined with a total eight Data Flows (see Figure 91). When we investigate these Data Flows we found that the half of them were connecting with other Pspec that are included with the Mayday Notification Service Package such as 5.2 - Provide Operator Interface for Emergency Data, 5.1.2 - Determine Coordinated Response Plan and 5.1.1.3 - Collect Incident and Event Data. The remaining four Data Flows are connecting Pspec that do not meant to be included with the Service Package such as 5.1.1.2 - Identify Commercial Vehicle Emergencies or 5.1.1.4.2 - Manage Secure Area Surveillance. This distinction is not made in the Functional View of Arc-IT and the user is expected to go through the list of all Data Flows defined for each Pspec and figure which Data Flows will be required for the systems

they desire to build. Another large problem regarding this issue is that even a Data Flow connects two Pspec that exist in the same architecture, it is not clear that if the Functional Objects need to communicate the specific Data Flow for the functionality required for the Service Package in question. Since same Functional Objects may be employed for several different Service Packages, architecture user should differentiate if any of the Data Flows are related for their architecture themselves with no guidance from the architecture.

This process is associated with the following data flows:

- [collected_incident_data](#)
- [emergency_verification_from_operator](#)
- [incident_and_event_data](#)
- [incident_cvo_data](#)
- [incident_sensor_data](#)
- [incident_surveillance_data](#)
- [threat_detected](#)
- [verified_emergency](#)

Figure 91 - The list of Data Flows that was defined for Pspec 5.1.1.1 - Coordinate Emergency Inputs (source: Arc-IT website)

We have given the Pspec 5.1.1.1 - Coordinate Emergency Inputs as an example since it is associated with only two Functional Objects and has relatively small amount of Data Flows defined. The Pspec 3.1.3 - Process Vehicle On-board Data on the other hand was associated with 22 Functional Objects throughout Arc-IT and a total of 88 Data Flows defined with it. When this is the situation it is not very clear for the architecture user (see Figure 92).

This process is associated with the following data flows:

- | | | |
|---|--|---|
| <ul style="list-style-type: none"> • collision_data • env_sensor_data_from_vehicle • fbv-brake_servo_response • fbv-diagnostics_data • fbv-driver_safety_status • fbv-vehicle_attitude_data • fbv-vehicle_charging_capacity • fbv-vehicle_charging_status • fbv-vehicle_driver_inputs • fbv-vehicle_identity • fbv-vehicle_motion_data • fbv-vehicle_occupants • fbv-vehicle_proximity_data • fbv-vehicle_safety_status • fbv-vehicle_security_status • fbv-vehicle_size • fbv-vehicle_speed • fuel_and_charging_status_for_vehicle • fvds-location • fvds-time • host_vehicle_details_for_emissions • host_vehicle_status_for_eco_drive • intersection_status_data_for_vehicle • parking_vehicle_payment_number • roadside_safety_data_to_vehicle • safety_data • safety_data_for_mcv • safety_message_data_for_remote_vehicles • safety_message_data_from_remote_vehicles • speed_management_data_to_vehicle • tbv-vehicle_status_details_for_driver • toll_vehicle_payment_number • traffic_situation_data_from_vehicle | <ul style="list-style-type: none"> • traffic_situation_data_from_vehicle_for_maps • traffic_situational_data_configuration • tvsc-vehicle_malfunction_data • tvsc-vehicle_system_status_for_diagnostics • tvsc-vehicle_type • vehicle_characteristics_for_emissions • vehicle_characteristics_for_roadside • vehicle_emv_current_status_data • vehicle_emv_location_data • vehicle_emv_size_data • vehicle_emv_speed_data • vehicle_env_sensor_data • vehicle_identity_for_central_payment_admin • vehicle_identity_for_road_use_payment • vehicle_location_for_central_payment_admin • vehicle_location_for_probe_data • vehicle_location_for_road_use_payment • vehicle_mcv_current_status_size • vehicle_mcv_location_data • vehicle_mcv_size_data • vehicle_mcv_speed_data • vehicle_occupants_detected • vehicle_ped_current_status_data • vehicle_ped_device_identity • vehicle_ped_location_data • vehicle_ped_size_data • vehicle_ped_speed_data • vehicle_ped_time_data • vehicle_roadside_control_event_data • vehicle_roadside_current_status_data • vehicle_roadside_location_data • vehicle_roadside_size_data • vehicle_roadside_speed_data • vehicle_roadside_surveillance_location_data • vehicle_roadside_surveillance_size_data • vehicle_roadside_surveillance_speed_data • vehicle_roadside_surveillance_status_data • vehicle_speed_and_distance_for_central_payment_admin • vehicle speed and distance for road use payment | <ul style="list-style-type: none"> • vehicle_status_details_for_broadcast • vehicle_status_details_for_driver_security • vehicle_status_details_for_emergencies • vehicle_status_for_intersection • vehicle_system_characteristics_for_emissions • vehicle_system_characteristics_for_roadside • vehicle_to_transit_current_status_data • vehicle_to_transit_device_identity • vehicle_to_transit_location_data • vehicle_to_transit_size_data • vehicle_to_transit_speed_data • vehicle_to_transit_time_data • vehicle_traffic_probe_data_for_archive • vehicle_traffic_situation_data • vehicle_traffic_situation_data_configuration |
|---|--|---|

Figure 92 - The list of Data Flows that was defined for Pspec 3.1.3 - Process Vehicle On-board Data (source: Arc-IT website)

After analyzing all Pspecs suggested for the Functional Objects for the Mayday Notification Service Package we have created below Data Flow Diagram in Enterprise Architect, representing the Functional View (see Figure 93).

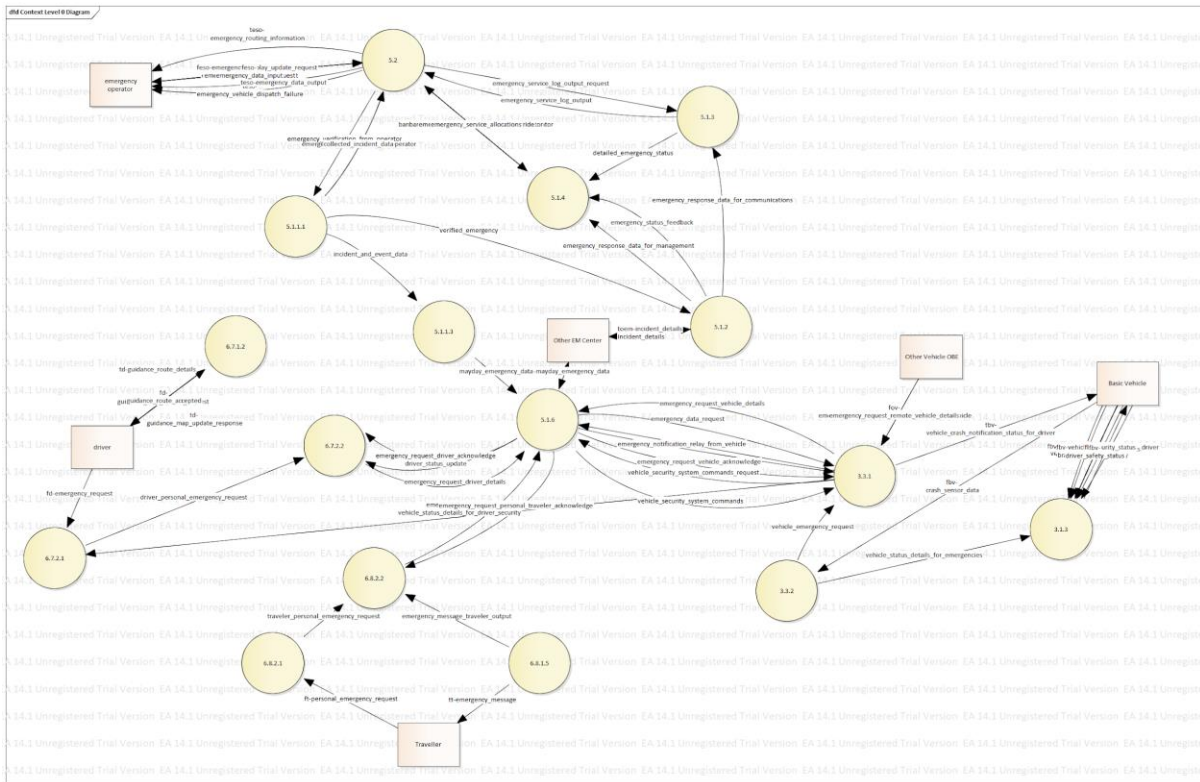


Figure 93 - The Functional View description of the Mayday Notification Service Package architecture (source: author)

Chapter 5 - Conclusion

In this paper we have analysed two major architecture frameworks in field of ITS. With the first chapter we have introduced the general concept of system architecting, we presented the international standard ISO/IEC/IEEE 42010 - Systems and software engineering — Architecture description to be able to introduce the terms and concepts related to the topic. We continued with introducing the European ITS Framework Architecture (FRAME) and the Architecture Reference for Cooperative and Intelligent Transportation (Arc-IT), presented their methodologies and elaborated on their structure and key concepts. In the third chapter we have presented a small portion of present day ITS and created architectures for this system in both FRAME and Arc-IT. In the fourth chapter we introduced a new modelling tool and transferred the Functional Viewpoint from FRAME to the new tool. We have proposed a new method of creating architectures of FRAME with this new tool since we have found the current tools of FRAME to be underdeveloped and missing features that today's architecture tools may offer. Finally we have transferred the Functional View from Arc-IT to this new tool, to contrast the effectiveness and usability of both architectures.

FRAME and Arc-IT offers its user more or less the similar thing. That is to catalogue the existing ITS and offering them to be selected by its user. Although both argues that they represent most of the current day ITS, we found slight differences in the systems that they offer. There are some features that was covered by one but not with the other and some other features that provides similar functionality but not exactly the same.

We found FRAME's approach of matching User Needs with systems' functionality to be a better approach compared to Arc-IT's Service Packages due to the level of control provided by the FRAME's approach. With this perspective the architecture user has a bigger say on what is wanted from the system(s) and what the system does to fulfil these needs. User can decide on what to include to the system(s) based on their needs whereas with the Service Packages they are presented the functionalities in a bulk and there is less room for them to customise the system(s) they build according to their needs. It found not to be clear how users' preferences would alter the system(s) they will end up with as a result. We also found that a more comprehensive Functional Viewpoint is offered with the FRAME, where it is clear why each function is added to the architecture compared to Arc-IT where a bunch of functionalities thrown to the user without justifying why the system(s) should include such functions and how not to include any function if they are not needed. The Functional View of Arc-IT was found to not clearly communicate how the data flows between the functions inside the Physical Objects and how they are processed but only showing its users the information flows between the different Physical Objects.

On the other hand we have found the way that Arc-IT supports the stakeholder interaction to be defined within the architecture which is a vital aspect when several entities are involved in building and implementing necessary roles within the system, which is inevitably the case with ITS. The Arc-IT allows not just to define but also to distribute roles to the stakeholders and document the agreements should take place between them while fulfilling their task related to the system(s). Finally we found that Arc-IT provides its user with many beneficial products for their architectures whereas the FRAME only provides a handful of lists of selected elements. It is possible to generate visual representations of the architecture to aid the architecture use to establish common

understanding between the related parties and also documents (even a website) detailing each aspect of the architecture.

We have proposed a new method for the FRAME architecture user to create their own architectures for their projects with a repository based modelling tool, Enterprise Architect to fill the gaps we found in the products that can be produced for their selected architectures. By the method we offer the user has direct access to the visual representation of their architectures. With many documenting options that is offered with the Enterprise Architect the architecture will have access to product that can be used to enhance the understanding and the cooperation between the parties involved with the architecture.

With future work the FRAME architecture should be more developed with the Organizational Viewpoint. It is understood that the Organizational Viewpoint was not developed, not to mandate any structure to the architecture user, but we see that Arc-IT also is not “mandating” any structure of such but offers its user means of setting relationship between the parties that are involved with the architecture. Just as it is in the Arc-IT, user has to be able to “define” the agreements and relationships between the stakeholders throughout the systems’ life cycle and make use of a well developed Organizational Viewpoint that would ease the cooperation between many parties that is involved in a transportation project.

References

1. US DOT. Federal Highway Administration. “Regional ITS Architecture Guidance”. Version 2.0. July 2006. <http://local.iteris.com/arc-it/documents/raguide/raguide.pdf>
2. International Standard ISO-IEC-IEEE 42010. “Systems and software engineering — Architecture description”. December 2011.
3. Golden B. “A unified formalism for complex systems architecture” École polytechnique. May 2013. <https://www.lix.polytechnique.fr/~golden/research/phd.pdf>
4. European Communities. “Planning a modern transport system”. Issue 2. April 2004. <http://frame-online.eu/wp-content/uploads/2014/10/PlanningGuide.pdf>
5. Hennessy J., Patterson D.. “A New Golden Age for Computer Architecture”. 13 June 2018. <https://californiaconsultants.org/wp-content/uploads/2018/04/CNSV-1806-Patterson.pdf>
6. European Commission, FP3-TELEMATICS 1C - Specific programme of research and technological development (EEC) in the field of telematic systems in areas of general interest, 1990-1994. <https://cordis.europa.eu/programme/rcn/194/en>
7. European Commission, Transport telematics - High level groups <https://cordis.europa.eu/news/rcn/7592/en>
8. Bossom R., Jesty P.. D15 – FRAME Architecture – Part 1: Overview. WP300 EC FP7 project E-FRAME. September 2011 <https://frame-online.eu/wp-content/uploads/2015/09/D15-FRAME-Architecture-Part-1-1.0.pdf>
9. Jesty P.. “D13 – Consolidated User Needs for Cooperative Systems”. WP200 EC FP7 project E-FRAME. September 2011. <https://frame-online.eu/wp-content/uploads/2014/10/D13-Consolidated-UNs-for-Coop-Systems-Issue.pdf>
10. Bossom R., Jesty P.. “D15 – FRAME Architecture – Part 5: FRAME Architecture Methodology”. WP300 EC FP7 project E-FRAME. September 2011. <https://frame-online.eu/wp-content/uploads/2015/09/D15-FRAME-Architecture-Part-5-1.0.pdf>
11. Dolean C. C., Petrușel R.. “Data-Flow Modeling: A Survey of Issues and Approaches”. Informatica Economică vol. 16, no. 4. 2012. <http://www.revistaie.ase.ro/content/64/14%20-%20Dolean,%20Petrusel.pdf>
12. “FRAME Selection Tool User Manual” - Retrieved from: <https://frame-online.eu/wp-content/uploads/2014/10/Selection-Tool-User-Manual-02.pdf>
13. US Department of Transportation. “Key Concepts of ARC-IT”. June 2017. <https://local.iteris.com/arc-it/documents/keyconcepts/keyconcepts.pdf>
14. Public Law 105-178. “Transportation Equity Act for the 21st Century”. Sec 5206. <https://www.fhwa.dot.gov/tea21/h2400-v.htm#5206>
15. Federal Transit Administration. “The Final Rule on Intelligent Transportation System Architecture and Standards”. Federal Register / Vol. 66, No. 5 January 2001. https://ops.fhwa.dot.gov/its_arch_imp/docs/20010108.pdf
16. US Department of Transportation. “Regional its architecture guidance”. July 2006. <https://local.iteris.com/arc-it/documents/raguide/raguide.pdf>
17. Code of Federal Regulation Title 23 Part 450. “Planning Assistance and Standards” <https://www.law.cornell.edu/cfr/text/23/part-450>
18. Gellens R.. “Next-Generation Pan-European eCall”. Internet Engineering Task Force (IETF). May 2017. <http://www.rfc-editor.org/pdf/rfc8147.txt.pdf>. (ISSN: 2070-1721)
19. “The interoperable EU-wide eCall” - Retrieved from: https://ec.europa.eu/transport/themes/its/road/action_plan/ecall_en
20. “eCall in New Cars from April 2018”, European Commission Digital Single Market Newsletter. 28 April 2015. <https://ec.europa.eu/digital-single-market/en/news/ecall-all-new-cars-april-2018>
21. Onstar. <https://www.onstar.com/us/en/services/safety-security/>
22. Mercedes Benz Me Connect. <https://www.mbusa.com/en/mercedes-me-connect>
23. Barca C. D., Ropot R., Dumitrescu S.. “eCall - Minimum Set of Data”. <ftp://ftp.repec.org/opt/ReDIF/RePEc/rau/jisomg/WI09/JISOM-WI09-A13.pdf>

24. Kaminski T., Niezgoda M., Kruszewski M.. "Collision Detection Algorithms in The eCall System". Journal of KONES Powertrain and Transport, Vol 19, no 4. 2012.
http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-article-BUJ8-0020-0032/c/httpwww_bg_utp_edu_plartkones42012j20o20kones20201220no20420vol_201920kami_nski2.pdf
25. Jesty P. H., Grochowski A., Bossom R. A. P.. "Example Case Study of eCall Using The FRAME Architecture". 2017
26. Bossom R. "Frame Architecture artefacts and their inclusion in the new FRAME architecture repository". 2019.