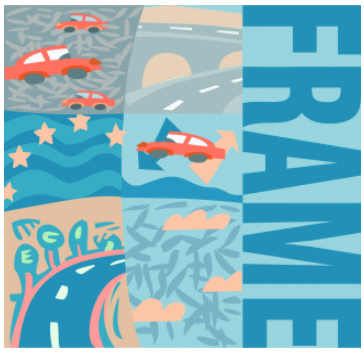


Programme name:	Information Society Technologies (IST)
Sector:	
Project ID:	IST-2000-29663
Project acronym:	FRAME-S
Project name:	Framework Architecture Made for Europe - Support
Working document:	European Commission and Project partners

Working document number:	**D14**
Contractual date of delivery:	N/A
Actual date of delivery:	30.04.2003
Title of working document:	**FRAME-S Guide to Configuration Management and ITS Architecture Documentation**
Work package:	WP 6
Nature of the working document:	Public Report
Author(s):	Richard Bossom (STC)
Project manager:	**R.A.P. Bossom (STC)**

Abstract: This document describes the Configuration Management practices to be adopted by Maintainers of the European ITS Framework Architecture. It also recommends similar practices for those developing ITS Architectures from the Framework Architecture. Guidance is provided on the content of the documentation that is produced to describe these ITS Architectures.

Keyword list: FRAME-S, European ITS Framework Architecture, Compatibility, Configuration Management, ITS Architectures.

European ITS Framework Architecture - Guide to Configuration Management and ITS Architecture Documentation

Version 1

April 2003

Produced by:

Siemens Traffic Controls, Leeds Innovations, ERTICO, Mizar, Mega

Document Control Sheet

Activity Name: FRAME

Work Area: Architecture Updates – WP6

Document Title: European ITS Framework Architecture – Guide to Configuration Management and ITS Architecture Documentation

Document Number: D14

Main author(s) or editor(s): Richard Bossom, Peter Jesty, Tuomo Eloranta, Olivier Le Guellec and Angela Nigro.

Dissemination Level: Public usage

Version history:

Version Number	Date	Editor	Summary of changes
1.0	April 2003	R.A.P. Bossom	First Version

Approval:

	Name	Date
Prepared	*Richard Bossom*	*February 2003*
Reviewed	*FRAME-S, FRAME-NET, ACTIF and ARTIST Project members*	*March 2003*
Authorised	*Richard Bossom*	*April 2003*

Circulation:

Recipient	Date of submission
CEC	April 2003

Issue History

Version	Date	Summary of Changes
1.0	April 2003	None, first issue.

Table of Contents

Tables

Figures

Executive Summary

This Document provides a guide to the Configuration Management practices for the European ITS Architectures that are based on the European ITS Framework Architecture. As a foundation to the definition of the Configuration Management practices, the Document includes a definition of what "compatibility" means for the Framework Architecture.

The Document is divided into four parts the first of which provides background information on the Framework Architecture, "compatibility" and Configuration Management. The background information includes an overview of the types of ITS Architecture that can be developed from the European ITS Framework Architecture and hence the environment in which the Framework Architecture is used. A detailed definition of the types of ITS Architecture that can be created from the Framework Architecture is provided in an Appendix to this Document.

The second and third parts of this Document provide details of the practices to be observed by the Maintainers of the European ITS Framework Architecture and Users of the Framework Architecture respectively. Maintainers will use the practices in the second part when creating Updates to the Framework Architecture and also when making changes to the Browsing and Selection Tools. Framework Architecture Users need to apply the practices in the third part of the Document when they are creating their own National, Regional, Local, or Specific ITS Architectures. Examples of the use of the numbering convention for additions and deletions to the Framework Architecture are provided as Appendices to this Document.

Finally the fourth part of this Document provides a guide for Users abut the creation of ITS Architecture documentation. It includes suggestions for both content and scope of the documentation. This is intended to help Users in the task of the producing the documentation needed to describe the particular ITS Architectures that they have created.

Although this document focuses on Configuration Management of the European ITS Framework Architecture, it is capable of being adopted by those of its Users who are creating their own Framework Architectures.

1. Introduction

1.1 Scope

This Document provides a definition of "compatibility" and a guide to the consequent Configuration Management practices for the European ITS Framework Architecture, and ITS Architectures that are derived directly from it. The Document provides guides for the Maintainers of the Framework Architecture and for its Users, plus a definition of "compatibility" and background information about the Framework Architecture. It also provides a guide to the creation of documents to describe the ITS Architectures created from the Framework Architecture.

1.2 Intended Audience

This Document is intended for Maintainers and Users of the European ITS Framework Architecture. These Users will be those wishing to use the European ITS Framework Architecture as the starting point for the creation of their own ITS Architectures.

Anyone developing their own ITS Architectures from the Framework Architecture must use these Configuration Management practices to remain compatible. The owner of the ITS Architecture should act as the "filter" for passing to the Maintainers of the Framework Architecture details of any adaptations (modifications or additions) that are made as part of the ITS Architecture creation process that might need to be reflected in the Framework Architecture itself.

This Document assumes that its readers will have some knowledge of the methodology that is used for the creation of ITS Architectures. Ideally therefore readers should have attended one of the FRAME Training Workshops, as they cover the methodology of creating ITS Architectures. If this has not been possible, then readers are recommended to study the information available from the FRAME Project Web Site at: http://www.frame-online.net. In particular they should study Sections 2 to 5 of the European ITS Framework Architecture Overview Document – see reference (1) plus the following other parts of the FRAME Web Site.

(a) The brochure and presentations available from "OVERVIEW" page.

(b) The "overview" and "history" sections of the "EUROPEAN ARCHITECTURE" page.

Users should also familiarised themselves with the references included in this Document whilst they are reading it. This will provide them with some knowledge of the contents of the European ITS Framework Architecture.

1.3 Document Structure

This Document is divided into four main parts. These are contained in Sections 2 to 5 and are as follows.

The first part (Section 2) provides some background information about "compatibility" and Configuration Management, plus the environment in which the European ITS Framework Architecture will be used. This is intended to provide the "context" for the remaining three parts of the Document.

In the second part (Section 3) a description is provided of the Configuration Management practices that will be adopted by the Maintainers of the Framework Architecture. It also includes a description of the practices that will be applied to the Browsing and Selection Tools that are being produced by the FRAME Project.

The third part (Section 4) contains a guide to Configuration Management practices that should be adopted by the Users of the European ITS Framework Architecture. The adoption of these practices will make it easier to co-ordinate the parallel ITS Architecture development work across Europe and also to help the maintenance of the Framework Architecture itself.

The last of the four main parts (Section 5) contains a guide to creating the documentation that Users will need produce in order to describe the ITS Architectures they have created. It provides suggestions for both the scope and contents of the documentation, depending on the type of ITS Architecture that is being created.

A set of references used by all parts of the Document is provided at the end of the in Section 6.

The Document includes several Appendices that provide illustrations of the Configuration Management practices that are described in parts of Sections 2, 3 and 4.

1.4 List of Abbreviations

The following abbreviations have been used throughout this document.

CEN	Comité Européen de Normalisation: the European Committee for Standardisation, responsible for creating and maintaining European standards.
DFD	Data Flow Diagram
EC	European Commission
FRAME	FRamework Architecture Made for Europe. This is used to encompass both the FRAME-S Project (responsible for producing this Document) and the FRAME-NET Project.
ISO	International Standards Organisation: responsible for creating and maintaining international standards
IT	Information Technology
ITS	Intelligent Transport System
KAREN	Keystone Architecture Required for European Networks (predecessor of the FRAME Project)

US United States (of America)

Note: 1. Throughout this Document the term “Framework Architecture” is used. This is a shortened form of “European ITS Framework Architecture” and is intended to make the Document easier to read.

2. The term “User” appears throughout this Document to represent those who are using the European ITS Framework Architecture as a starting point for their own ITS Architecture development. It also includes those who are developing Specific ITS Architectures from Framework or Mandated ITS Architectures that were originally developed from the European ITS Framework Architecture.

2. Background Information

2.1 Introduction

This section of the Document is designed to provide background information about the European ITS Framework Architecture, the reasons for the existence and context for the Configuration Management practices that have been developed for its Maintainers and Users. It includes a definition of what "compatibility" means for the Framework Architecture and the ITS Architectures that are derived from it. The Configuration Management practices for Maintainers and Users are described in the next two Sections (3 and 4) of this Document. It is intended that the information in this Section (2) should help those unfamiliar with the Framework Architecture to gain a better understanding of the practices and their implementation.

2.2 European ITS Framework Architecture Information

2.2.1 Introduction

Before describing defining compatibility and the Configuration Management practices to be adopted for the European ITS Framework Architecture, it is necessary to understand the environment in which it is used. This understanding will set in context the practices and will help to explain why they are needed and how they should be applied.

2.2.2 European ITS Framework Architecture history and purpose

The KAREN Project created the Framework Architecture as part of the European Commission's 4th Framework Programme. Version 1 of the Framework Architecture was released through the EC in November 2000. The reason for creating the Framework Architecture was (and is) to provide a starting point for the development of ITS Architectures within Europe. These ITS Architectures are seen as the mechanisms through which compatible and inter-operable deployment of ITS services can be achieved across Europe. This is expected to provide seamless journeys for Travellers and Freight and to lead to the creation of a large marketplace for ITS related produces.

2.2.3 Types of ITS Architecture that can be derived from the Framework Architecture

There are three types of ITS Architectures that can be created from the European ITS Framework Architecture. They are defined as follows.

1. Framework ITS Architecture: - comprises a set of User Needs and a Functional Viewpoint. These will cover all services that it is possible to provide using the Architecture. The Architecture may also optionally include guidance on how to create Physical and Communications Viewpoints, Deployment Studies, etc. It can then be used as the starting point for the creation of Specific ITS Architectures – see point 3 below. *Example: the European ITS*

Framework Architecture, which can be obtained from the "LIBRARY" page of the FRAME Web Site, which is available at: http://www.frame-online.net.

2. Mandated ITS Architecture: - comprises a set of User Needs, plus Functional, Physical and Communications Viewpoints and other outputs. The Physical Viewpoint may include some flexibility, the impact of which must be highlighted in the Communications Viewpoint and other outputs. It may be used for National, Regional or Local (City) ITS Architectures, and (depending on the degree of flexibility) be used as the starting point for component and infrastructure specifications. Mandated Architecture may also be used as a starting point for the creation of Specific ITS Architectures, usually for Regions or particular localities, e.g. Cities. *Example: the French National ITS Architecture (ACTIF), which is available from its Web Site at: http://www.its-actif.org.*

3. Specific ITS Architecture: - comprises a set of User Needs, plus definitive Functional, Physical and Communications Viewpoints and other outputs. It is used for ITS Architectures covering the deployment of a specific set of Services, e.g. Traveller Information, Public Transport Management, in a particular geographic area, and acts as the starting point for component and infrastructure specifications for a particular deployment. *Example: several are included in the UTMC Example Systems document available from the UTMC Web Site at: http://www.utmc.dft.gov.uk/utmc22/utmc22.htm.*

The main differences between these three types of ITS Architecture are in their goals and scope, rather than in their structure. Naturally the European ITS Framework Architecture is of the first type that is described above. Users have the freedom to create from it ITS Architectures of one of these three types to serve the needs of their own ITS deployments.

A more detailed description of these three types of ITS Architecture and the relationship between them is provided in Appendix 1 of this Document.

2.2.4 European ITS Framework Architecture environment

The types of ITS Architecture that can be created from the European ITS Framework Architecture defined in the previous section may not the same as those created from other ITS Architectures. This is because the environment in which the European ITS Framework Architecture will be used is different from those for these other ITS Architectures. These differences can be summarised into the following four points.

1. There are many ways in which Users can create ITS Architectures from the European ITS Framework Architecture. This is because it has been designed to be easily adaptable by Users to suit their own ITS deployments. Most National ITS Architectures are "mandated" and are used in either of the following two ways.
    (a) To create specifications for components and systems to be used in particular ITS deployments.

(b) To create Regional or Local ITS Architectures as sub-sets of the original Architecture.

The Users of these National ITS Architectures are restricted as to what changes they can make to accommodate their own particular ITS deployments. In both cases Users will start from a defined Physical Viewpoint[1], from which they can select the part(s) that suit the specific Services that they need. Users of the European ITS Framework Architecture have no defined Physical Viewpoint that they can use. Instead they must create their own versions of this Viewpoint to suite their particular ITS needs.

2. The European ITS List of User Needs and the Functional Viewpoint (which form part of the European ITS Framework Architecture) are specifically structured so as to make their adaptation easy. Thus extra User Needs can be added or omitted without affecting those that already exist. Similarly changes can also be made to the Functional Viewpoint components, namely the Functions, Functional Data Flows and Data Stores. However in the latter case care has to be taken to ensure that the resulting Functional Viewpoint is both logically and functionally consistent, i.e. error free.

3. The European ITS Framework Architecture may be translated into one of the many European languages. Although some non-UK European Users speak English, they have a natural desire to develop their version of the Framework Architecture in their own language so that it is more accessible to its target audience. Whilst many Users have not actually translated the actual component names, particular problems arise with the use of acronyms. These are used to identify such things as Functional Areas and Terminators in Data Flow names. This imposes constraints on the naming conventions that can be used for the Framework Architecture, but is not a modelling issue for the Architecture.

4. Europe is not a “Single State”, so the ITS services that are acceptable in one country may not be applicable in another. Even when they are, it is possible the implementation method included in the functionality may have to be different. This means that the functionality in the European ITS Framework Architecture has to be provided in a way that does not exclude different national service implementations.

All of these factors have some influence on the Configuration Management practices that have been defined for the European ITS Framework Architecture. They affect both its Maintainers and its Users and are described in Sections 3 and 4 of this Document.

[1] In the first Version of the European ITS Framework Architecture and its first update (Version 1.1) these and other parts were defined as “Architectures”, e.g. Communications Architecture. The change to “Viewpoints” has been made in Version 2 and complies with the IEEE Standard that was introduced after completion of the KAREN Project.

These factors will also affect the scope and content of the documentation produced to describe the ITS Architectures that are based on the European ITS Framework Architecture. Guidance on these aspects of the documentation is provided in Section 5 of this Document.

2.3 Compatibility

2.3.1 Introduction

The following sections of this Document provide definitions of what "compatibility" means and why it is needed. Without these definitions, the term has no relevance to the European ITS Framework Architectures and any ITS Architectures derived from it. The definition of "compatibility" also enables the Configuration Management practices to be defined and given meaning.

2.3.2 What does compatibility mean?

The content and scope of any Configuration Management practices will be driven by the definition of what "compatibility" means. In general terms, for one system to be compatible with another they must possess features and/or characteristics that are the same.

The proportion of the total number of features and/or characteristics that are the same defines the degree of "compatibility". For example if everything is the same, then two systems are said to be "fully" or "100%" compatible. This means that they should have at least one of the following attributes.

(1) Be fully inter-operable, i.e. they should be able to work together without any modifications.

(2) Be exchangeable, i.e. one object can replace the other to perform the same functions.

(3) Use interchangeable components, i.e. it should be possible to move a component from one object to another.

The number of these attributes that apply to any two systems will depend on exactly what components they have and what the systems do. Also if the "compatibility" is less than 100% then some or all of the attributes may not apply.

2.3.3 What will be compatible?

There are two topics to be considered if ITS Architectures are to be compatible with the European ITS Framework Architecture. They concern both its development and its content and are as follows.

Topic 1: Methodology. A study of ITS Architecture developments around the world shows that in some cases different methodologies are being used. They cover the way in which the Architecture is created, the User Needs are structured and the interfaces with the outside World are specified.

Topic 2: Features and Characteristics. These cover the parts of the European ITS Framework Architecture such as User Needs, Terminators, Functions, Data Flows and Data Stores. The ways in which these can be compatible is described below.

The first Topic is being promoted by the FRAME Project through its Training Workshops. These take attendees through the process and methodology of creating ITS Architectures that are based on the European ITS Framework Architecture.

The second Topic needs to be covered if "compatibility" is to have any real value and meaning. For the European ITS Framework Architecture and ITS Architectures derived from it, the features that define "compatibility" are shown in the following table.

Table 1 Features that define "compatibility" with the European ITS Framework Architecture

ITS Architecture		Required for compatibility	Comment
Part	**Characteristic**		
User Needs	Number	No	Cross-reference table **must** be provided if numbers not the same.
	Definition	Yes	
Terminators	Name	Yes	Includes Actors.
	Description	Yes	
Function	Number	No	Cross-reference table **must** be provided if numbers not the same.
	Name	Yes	
	Definition	Yes	Includes Overview, Input/output Data Flows, Functionality and User Needs served.
Data Flow	Name	No	Cross-reference table **must** be provided if names not the same.
	Description	Yes	
	Source & Destination Functions	Yes	Only the name and definitions must be the same – see compatibility for Functions.
Data Store	Number	No	Cross-reference table **must** be provided if numbers not the same.
	Name	Yes	
	Contents	Yes	

All characteristics such as hierarchy structures will be the same as a result of conformance to the above "compatibility" features. The language in which the ITS Architecture is written (if not English) should not be a "compatibility" issue, provided that actual translation is carefully managed to avoid differences of interpretation.

Topic 2 can only be achieved through the use of Configuration Management practices. For the Framework Architecture and ITS Architectures derived from it, these practices are described in Sections 3 and 4 of this Document.

2.3.4 Degree of compatibility

The degree of "compatibility" will be expressed in terms of how much of the ITS Architecture has been taken from the European ITS Framework Architecture. Thus, if the ITS Architecture includes no additional features or modifications, then it will be fully (100%) compatible. This will be true even if it only supports some of the services supported by the Framework Architecture.

2.3.5 Why is compatibility needed?

There are at least three reasons why ITS Architectures that are derived from the European ITS Framework Architecture need to be compatible with it. The first two can be classed as "political" and the third concerns system operation.

Reason 1: "selling"/acceptance. If ITS Architectures are (or are intended to be) compatible with the Framework Architecture then it helps to "sell" their use to stakeholders and other users, and thus make them more acceptable.

Reason 2: funding. Obtaining funding for the development of a National, Regional or Local ITS Architecture can be easier if it is known that it will be compatible with the Framework Architecture. This is because organisations will be more "comfortable" investing in the Architecture development knowing that is based on the something that has already been proven and used by other similar developments.

Reason 3: inter-operability. One of the major benefits that ITS Architecture development can provide is the ease with which components that are developed from it can be made inter-operable. This means that the components will have the following attributes.

(a) The interfaces between the components can confirm to recognised standards, such as those produced by CEN and available through national standard organisations. This means that it is easier to replace a component with a newer version.

(b) The components work with each other in a way that means that it is easy to replace one without disrupting the operation of the others.

(c) Components in one system derived from the ITS Architecture can be easily linked to those in another system. Of course they must either be derived from the same ITS Architecture or from

two Architectures that are both related to the Framework Architecture.

Compatibility needs to be supported by an understanding about the principles behind the Framework Architecture and the communication of what they include. These principles cover things such as the methodology used to develop the Framework Architecture and conventions for things such as the numbering and naming of its parts. They are described in this Document, the reference documents identified in Section 6 and in the materials for the FRAME Training Workshops.

2.4 Configuration Management

2.4.1 Introduction

This section of the Document provides a definition of what Configuration Management means in the general sense. It then looks at how it affects the European ITS Framework Architecture.

2.4.2 What is Configuration Management?

Configuration Management is a mechanism for controlling the way that changes are made to anything that has been produced and is being used. In the Information Technology (IT) domain it can be applied to software, hardware and their supporting documentation.

Configuration Management defines how changes shall be made to software, hardware and their supporting documentation. These changes may be additions, modifications or deletions. The five main objectives of Configuration Management are as follows.

(1) To know what changes have been made.

(2) To prevent indiscriminate changes.

(3) To define the way that changes are made.

(4) To make sure that "backwards compatibility"[2] is a high priority goal of any changes that are made.

(5) To ensure that the mechanisms for maintain "compatibility" between objects and components are defined.

How the above objectives are to be achieved is described in a Configuration Management document produced for the software, hardware and their supporting documentation. Maintainers of software, hardware and its supporting documentation need to abide by the contents of that document.

Intelligent Transport Systems (ITS) consist of components that are similar to those in IT, so it is logical to apply the same Configuration Management practices. An

[2] The term "backwards compatibility" means that the creation of any new version of software or hardware does not invalidate anything created from the versions that already exist. In other words, the previous level of "compatibility" is maintained.

example of practices that have already been recommended is provided by the document in reference (2).

2.4.3 Why is the European ITS Framework Architecture affected?

The European ITS Framework Architecture describes the functionality that is needed to implement ITS Services. The stakeholders in the transport network will want to implement these Services in order to improve the transport of people and the movement of goods. The ways in which stakeholders can be involved in the transport network is described in section 3.1 of reference (9).

The Framework Architecture is designed for use as the starting point for the development of National, Regional, Local, or Specific ITS Architectures by European organisations. These Architectures contain functionality that will be used as the starting point for the development of components to provide the Services that stakeholders require. These components can be produced as software and/or hardware. Changes to the Framework Architecture can have an important effect on this component development work. Thus it is both necessary and desirable to manage the way that these changes are implemented. This can be achieved using Configuration Management.

2.4.4 How is the European ITS Framework Architecture affected?

There are two main goals that must be achieved by the Configuration Management of the European ITS Framework Architecture. These are as follows.

1. Any changes to the Framework Architecture introduced by its Maintainers that update it into new Versions must (as far as possible) not invalidate the "compatibility" of any ITS Architectures created from previous Versions.
2. Any adaptations made by the Framework Architecture Users in their efforts to create their own ITS Architectures must not compromise the Configuration Management of the Framework Architecture. In this case the maintenance of "compatibility" will be the responsibility of the Users.

In order to achieve the above, two sets of practices are set out in the following Sections of this document. One set (Section 3) is for Framework Architecture Maintainers and the other (Section 4) is for Framework Architecture Users. Both need to be followed if Configuration Management of the Framework Architecture is to be achieved and "backwards compatibility"[3] provided for ITS Architectures that are based on it.

As mentioned previously it should be possible for those European ITS Framework Architecture Users who are creating their own ITS Framework Architectures to adapt this Document for their own Configuration Management practices. The alternative is for them to create a document describing their own Configuration Management practices that suit their particular ITS Architecture environment. However it is

[3] See Note 2.

strongly recommended that these practices be based on those for the Framework Architecture that are described in this Document.

2.4.5 Scope of Configuration Management for the European ITS Framework Architecture

The Maintainers of the European ITS Framework Architecture (currently the FRAME-S Project) will be responsible for its Configuration Management practices. The scope of these practices will be restricted to the following parts of the Framework Architecture.

User Needs

Terminators and their Actors

Terminator Data Flows

Functions

Functional Data Flows of all types

Data Stores

Details of the current content of these parts of the Framework Architecture will be found in the List of European ITS User Needs and the Functional Viewpoint[4] – see references (3), plus (6) to (9). As User Needs and Functions are both included in the above list, the results of any changes will also be applied to the Trace Tables – see reference (5).

Although not part of the original (Version 1) Framework Architecture, the Browsing and Selection Tools originally produced by the FRAME-S Project will also be subject to Configuration Management. The ways in which this will be achieved are described at the end of the following Section (3).

3. Configuration Management practice for European ITS Framework Architecture Maintainers

3.1 Introduction

This section defines the Configuration Management practices to be adopted by the Maintainers of the European ITS Framework Architecture. They could also be used as guidelines for developing Configuration Management practices by organisations creating and/or maintaining ITS Framework Architectures that are based on the European ITS Framework Architecture – see type 1 in section 2.2.3.

3.2 General Principle

The general principle that will be adopted by the Maintainers of the European ITS Framework Architecture is that of trying to maintain "backwards compatibility", as defined in Section 2.4. This means that when a new Version of the Framework Architecture is produced, it should not invalidate any existing National, Regional, Local and Specific ITS Architectures that have been created from any previous Versions. Where this policy may not be possible in a general sense it will be highlighted in the following Sections. It will also be highlighted in the documents that accompany the affected versions.

3.3 Framework Architecture Version Numbering

The numbering scheme for Versions of the European ITS Framework Architecture will be based on the use of two numbers. Theses will be used in the following way.

m.n where, m = the number for a new Version that incorporates changes to the ITS Services that can be provided.

n = the number for a new Version that incorporates changes that remove inconsistencies in a previous Version – Maintenance Releases.

The second number (n) can be incremented without the first number (m) being changed for Maintenance Releases. Increments in the first number (m) will cause the second number (n) to revert to zero (0), in which case it may be omitted. Thus the Version numbers will follow the pattern shown in the following Table.

Table 2 Framework Architecture Version Numbers

Framework Architecture Version Number	Comments
1	Initial Release – November 2000.
1.1	Maintenance Release for Version 1 – March 2002.

Framework Architecture Version Number	Comments
2	Update to incorporate new Services.
2.1, 2.2, etc.	Maintenance Releases to remove inconsistencies in Version 2.
3	Update to incorporate further new Services.

Each new Version will be based on the preceding Version. This means that the changes in each Version will be incremental.

3.4 Specific Principles for parts of the Framework Architecture

3.4.1 Introduction

This section sets out the specific principles of Configuration Management that will be applied to the different parts and component types of the European ITS Framework Architecture. It is structured so that there is a separate Section for each part and component.

3.4.2 User Needs

The following principles will be applied to changes to the European ITS User Needs, which are defined in reference (3).

1. A User Need that is no longer required will have its links to Functions deleted but it will not be removed. Its text will be modified to add an indication that it is no longer used.
2. A new User Need will be added within the existing structure. It will take the next appropriate number for a Group, or Service, etc. – see also section 3.4.8.
3. A new User Need may duplicate an existing User Need. In this case the new and existing (duplicate) User Needs will each show the new reference(s) to the other(s).
4. Allocations of User Needs to Functions may be changed, providing that this does not affect the contents (functionality) of the Functions.

Further information about the creation of the European ITS User Needs can be found in reference (4). The allocation of User Needs to Functions can be found in the tables contained in reference (5).

3.4.3 Terminators

The following principles will be applied to changes to the Terminators of the European ITS Framework Architecture.

1. Existing Terminators and any Actors in existing Terminators that are no longer required will not be deleted. In addition, their acronyms will be retained and not re-used.
2. If at all possible, when the Framework Architecture requires an additional representation of the outside World, a new Actor will be added to an existing Terminator rather than a new Terminator being created.
3. If the requirement in 2 is not possible or appropriate, then new Terminators will be created in order to provide additional representations of the outside World.
4. The names of new Terminators and Actors will be in English – see also section 3.4.8

Further information about the existing Terminators and their Actors can be found in Chapter 4 of reference (6).

3.4.4 Terminator Data Flows

The following principles will be applied to changes to all types of Terminator Data Flows in the European ITS Framework Architecture.

1. Existing Terminator Data Flows can only be deleted for one of two reasons. Firstly their source and/or destination Terminators, Actors, Functions or Data Stores have been deleted, and secondly their names have been incorrectly spelt. In both cases the actual deletion process will only involve their removal from the lists of input and output Data Flows for Functions, and as components of other Terminator Data Flows.
2. All new Terminator Data Flows must take names that have not been used before. These must conform to the naming conventions for the particular type of Terminator Data Flow of which they are a part and use the acronym for the Terminator or Actor – see also section 3.4.8.
3. If a new Terminator Data Flow is added then its name must conform to the existing hierarchy for Terminator Data Flows. This means that unless a new Terminator has been added – see previous section, the new Data Flow should be added as a component of an existing High-level Data Flow. If the new Data Flows are for a new Terminator, then the existing conventions for high-level and medium-level Data Flows must be followed. Thus for example each new high-level Data Flow must have a new medium-level Data Flow for each Functional Area that contains an interface to the new Terminator.
4. If an existing Terminator Data Flow is deleted as a component from another existing Data Flow, then the description of the remaining (parent) Data Flow must be updated.

Further information about all types of Terminator Data Flows and a description of their naming conventions can be found in reference (8).

3.4.5 Functions

The following principles will be applied to changes to the Functions in the European ITS Framework Architecture.

1. As a general rule, when any existing Functions are no longer required, they will not be deleted and their numbers and names will not be re-used. The only exceptions to this are as follows.
   (a) A Function is outside the scope of the Framework Architecture and in consequence will never be used.
   (b) A Function becomes illegal, i.e. the use of the Service provided totally, or in part, by its functionality is prohibited at the European level.

   An illustration of how a Function is deleted and its impact is provided in Appendix 4.
2. The names of existing Functions will only be changed to correct an error, e.g. if the name does not reflect the contents of a Function, i.e. what it does. Although renamed, the number, functionality inside the Function and all its input/output Data Flows will not be changed.
3. A new Function will be created if an existing Function needs to be modified in any of the following ways.
   (a) The functionality needs to be modified due to changes in the allocation of existing User Needs, or new User Needs are added – see section 3.4.2. The modification may be through either addition or deletion of functionality.
   (b) Input or output Data Flows are added or deleted to or from the Function. Note that this will also require a change to the functionality.
   (c) If Users of the Framework Architecture find that a particular Low-level Function is too complex. An example of this is when Physical Viewpoints are created and unwanted functionality is being needlessly included in a Sub-system or Module. In this instance the particular Low-level Function may be turned into a High-level Function. The new and existing functionality must be included in two or more Low-level Functions, whose parent is the new High-level Function.

   The only exceptions to the above will be for the correction of errors in existing Functions. For example, if an input or output Data Flow is listed in the Function Specification, but not included in the functionality or shown in the Data Flow Diagram (DFD), then it may be removed, since this does not affect what the Function does.
4. If a High-level Function is created from a Low-level Function, as in 3(c) above then it can retain the same number if its input and output Data Flows are unchanged. Otherwise the Function must be re-numbered – see next point.

5. In order to preserve "backwards compatibility"[4] all new Functions must take numbers and names that have not been used before – see also section 3.4.8. The number must be the next new number in the sequence for the Functional Area in which the Function resides. The Function name will always include a "verb" to show that it "does something" – see Appendix 5 for an illustration.
6. Whenever possible the creation of new Functions will not require the creation of a new Functional Area. Any new Functions will be added to those in the appropriate existing Functional Area. In doing this, "backwards compatibility"[5] between the two versions of the Functional Areas must be preserved wherever possible.
7. Functions that are no longer needed because they have been replaced or deleted will not be shown in the revised versions of the DFD's to which they belong. This practice will be followed in order to avoid making the Diagrams over complicated and confusing.

An overview of the existing functionality and descriptions of all the Data Flow Diagrams can be found in Chapters 6 to 13 of reference (6). Details of the Function Specifications can be found in reference (7).

Appendix 3 provides an illustration of the addition of a new Function. This will be the result of the creation of some new User Needs in order to enable the services provided by the Framework Architecture to be expanded.

3.4.6 Functional Data Flows

The following principles will be applied to changes to all types of Functional Data Flows in the European ITS Framework Architecture.

1. Existing Functional Data Flows can only be deleted for one of two reasons. Firstly their source and/or destination Functions or Data Stores have been deleted, and secondly their names have been incorrectly spelt. In both cases the actual deletion process will only involve their removal from the lists of input and output Data Flows for Functions, and as components of other Functional Data Flows.

2. All new Functional Data Flows must take new names that have not been used before. These must conform to the naming conventions defined in Chapter 2 of reference (8) – see also section 3.4.8.

3. All new Functional Data Flows must have a single source and destination Function or Data Store. Data Flows to or from Low-level Functions must have a source or destination that is another Low-level Function. This means that the same Data Flow must not be used as the source or destination of two different Functions.

[4] See Note 2.
[5] See Note 2.

4. If a new Functional Data Flow is added as a component of an existing Data Flow, then the name of the existing (parent) Data Flow must be changed and its description updated.

5. If an existing Functional Data Flow is deleted as a component from another existing Data Flow, then the name of the remaining (parent) Data Flow must be changed and its description updated.

6. If a Data Flow is added to or removed from the list of Input or Output Data Flows for a Low-level Function or a Data Store, then the they must be given a new name and number that has not been used before – see previous section.

7. Functional Data Flows that are no longer needed because they have been replaced or deleted will not be shown in the revised versions of the DFD's to which they belong. This practice will be followed in order to avoid making the Diagrams over complicated and confusing.

Further information about all types of Functional Data Flows and a description of their naming conventions can be found in reference (8).

3.4.7 Data Stores

The following principles will be applied to changes to the Data Stores in the European ITS Framework Architecture.

1. Existing Data Stores can only be deleted for one of two reasons. Firstly the Functions that provide or receive data to or from them have been deleted, and secondly their names have been incorrectly spelt. In both cases the deleted numbers and names cannot be re-used.
2. When a Data Store is deleted, all of its input and output Functional Data Flows must also be deleted, but their names cannot be re-used – see point 1 in section 3.4.6.
3. If a new Data Store is created, then it must take the next new number in the sequence for the Functional Area in which it resides. It must also be given a name that has not been used before – see section 3.4.8.
4. The Functional Data Flows created to provide access to/from a new Data Store must also be given names that have not been used before – see points 2, 3 and 4 in section 3.4.6.
5. The creation of new Data Stores may require the creation of one or more new Functions. These new Functions will be added to those in the appropriate existing Functional Area – see section 3.4.5, without creating a new Functional Area. In doing this, "backwards compatibility"[6] between the two versions of the Functional Areas must be preserved wherever possible.

[6] See Note 2.

6. Data Stores that are no longer needed because they have been replaced or deleted will not be shown in the revised versions of the DFD's to which they belong. This practice will be followed in order to avoid making the Diagrams over complicated and confusing.

Further information about the existing Data Stores and their naming convention can be found in reference (9).

3.4.8 Choice of new Numbers and Names

When European ITS Framework Architecture updates are being made to include changes that have already been developed by a particular user (or group of Users), the numbers and names that they have used may not be adopted. The way that numbers and names will be handled is as follows.

(a) Numbers – the next available number in the appropriate sequence for the part of the European ITS Framework Architecture (User Need, Function, or Data Store) that is being changed will be used. The numbers (and their suffixes) created by the user(s) will be ignored – see section 4.3 and Appendix 3.

(b) Names – wherever possible the same name will be used if it is in English. If not, then an equivalent English name will be used. If the user has included a country specific suffix, it will be removed – again see section 4.3 and Appendix 3.

When English versions of names are produced, they will conform to the conventions to the part of the Framework Architecture to which they relate. For example the English names of Functions will always include a "verb" to show that they "do something". The naming conventions for Data Flows are described in Chapter 2 of reference (8).

3.4.9 Identification of parts of the Framework Architecture that have been changed

3.4.9.1 *Introduction*

It will be very important to identify what has been changed in the European ITS Framework Architecture in order to create a new Version. Wherever possible options available from the tool used to maintain the Framework Architecture should be used to provide this information. The following sections provide some guidance on what should be done and are intended to be "tool independent".

3.4.9.2 *Parts that are no longer used*

Parts of the European ITS Framework Architecture that are no longer used by a new Version must be clearly identified. The detail of the mechanism for this will depend on the tool being used for its maintenance, but something such as a "deleted" flag should be used for all parts that are no longer used. These parts should be excluded from the Browsing Tool – see section 3.6, but retained within the data model for the Framework Architecture.

3.4.9.3 *Parts that are added*

When new parts are being added to the Framework Architecture, the Maintainers must check to ensure that the new names have not been previously been used. They should be able to do this by using the maintenance tool to check the "deleted" flag referred to in section 3.4.9.2.

Each new part should have a "Version N" flag, similar to the "deleted" flag described about. The "N" would identify the Version in which the part was introduced and would be a permanent record, maintained through several Versions. Thus in Version 4 of the Framework Architecture it would be possible to identify the changes introduced in Versions 2 and 3, as well as those introduced in Version 4.

3.4.9.4 *Parts that are corrected*

When parts of the Framework Architecture are corrected, a "corrected" flag should be applied to each part. Again this will be similar to the "deleted" flag referred to in section 3.4.9.3.

3.5 Documentation of Changes to the European ITS Framework Architecture

3.5.1 Introduction

Changes to the European ITS Framework Architecture will be documented in an Update Document, currently produced by the FRAME-S Project. This will be numbered in the sequence of Deliverable Documents for the Project and identify the Framework Architecture number to which it relates.

The Update Document will provide a record of parts of the European ITS Framework Architecture that have been added, corrected and/or no longer used in a new Version. The changes will have been made using the mechanisms described in the previous parts of section 3. Where possible, automatic generation facilities in the maintenance tool will be used to provide the information for the Document.

3.5.2 Contents of Update Document

The Update Document must provide the following information about changes to parts of the Framework Architecture.

(1) Introduction

This will include a high-level description of the changes that have been made to the European ITS Framework Architecture and the source of all the Update Requests. The reasons for accepting/rejecting the Update Requests will also be included, together notes about any that have been combined.

(2) User Needs

(a) The identity of all new User Needs, i.e. their numbers, plus the Functions that serve them. A reference to any User Needs that are now duplicated will also be included.

(b) The identity of any User Needs that are no longer used, together with the affected Functions.

The definitions of the new User Needs will be provided in an updated version of Appendix F to the European ITS User Needs Document – see reference (3).

(3) Terminators

(a) The name and description of any new Actors that have been added to existing Terminators, plus their acronym and the name of the Terminator to which they belong.

(b) The name and description of any new Terminator, plus its acronym and details of any Actors that it contains – see (a) above.

(c) The names of any new Terminator Data Flows created to service the new Actors and/or Terminators.

The definitions and acronyms of the new Actors and Terminators will also appear in an updated Chapter 4 of the updated version of the European ITS Framework Architecture Functional Viewpoint Document – see reference (6).

(4) Functions

(a) The number and name of any new Functions, with a list of User Needs that they serve, plus the identity of the Data Flow Diagrams (DFD's) in which they are located.

(b) The number and name of any deleted Functions, with a list of User Needs that they no longer serve and the DFD from which they have been removed.

The descriptions of the modified versions of existing DFD's and new DFD's that include the new or deleted Functions will be provided in Chapters 6 to 13 of the updated version of the European ITS Framework Architecture Functional Viewpoint Document – see reference (6). The Specifications for the new Functions will be described in reference (7).

(5) Data Flows

(a) The names of all new Data Flows, plus the identity of the Data Flow Diagrams (DFD's) in which they are located, plus the name of their parent Data Flows (if any) and the identities of their source and destination Terminators, Functions, or Data Stores.

(b) The names of any deleted Data Flows, plus the identity of the Data Flow Diagrams (DFD's) from which they have been removed, plus the name of their original parent Data Flows (if any) and the identities of their former source and destination Terminators, Functions, or Data Stores.

The descriptions of the new Data Flows will be provided in the updated version of the European ITS Framework Architecture Functional Viewpoint Document Annex 2 – see reference (8).

(6) Data Stores

(a) The numbers and names of any new Data Stores, plus the identity of the Data Flow Diagrams (DFD's) in which they are located.

(b) The numbers and names of any deleted Data Stores, plus the identity of the Data Flow Diagrams (DFD's) from which they have been removed.

The descriptions of the new Data Stores will be provided in the updated version of the European ITS Framework Architecture Functional Viewpoint Document Annex 3 – see reference (9).

(7) Trace Tables

There will be no specific mention of the changes to the Trace Tables. This is because these will have been covered by the details provided in (1) and (3) above. The new allocation of User Needs to Functions will be provided in the updated version of the European ITS Framework Architecture Overview – see reference (5).

When there are changes to other part of the text in the European ITS Framework Architecture documents a note of the effected section(s) and the reason(s) for the change will be provided. For long sections the identities of the changed paragraphs will also be provided.

3.5.3 Documentation of parts that have been corrected

The Update Document will provide a list of corrections that have been made to parts of the Framework Architecture. The "list" will include all the corrections made since the Update to include new ITS services. This will be identified by a change to the first digit (m) in the Version number – see section 3.3. Thus a Document describing the corrections made to create Version 3.3 will also include a list of those made to create Versions 3.1 and 3.2.

3.6 Browsing Tool

The Browsing Tool will be updated with each new Version of the European ITS Framework Architecture. This will be driven by the fact that the Tool has to be "regenerated" every time the Framework Architecture is changed. However in order to allow the Tool to be updated independently of the Framework Architecture, it will

use a release number of up to three characters that follows the Architecture version number. This will consist of a letter "r", followed by a major release identity in numerals and a possible minor release letter in lower case (letter "a" can be omitted). The release number will **not** be re-set to "r1" after new Framework Architecture releases. This number will be used in the way shown in the following Table.

Table 3 Framework Architecture and Browsing Tool Version Number Relationship

Framework Architecture Version Number	Browsing Tool Core release	Corresponding Browsing Tool Version Numbers
2	r1, r1b	2.0r1, 2.0.r1b, etc.
2.1	r1b, r2	2.1r1b, 2.1r2, etc.
2.2	r2, r3	2.2r2, 2.2r3, etc.
2.3, etc.	r3, r3b	2.3.r3, 2.b.r3b
3	r3c, r4	3.0r3c, 3.0r4, etc.

The only reason for producing a new version of the Tool without a new Version of the Framework Architecture will be to change some aspect of its appearance, or some other feature of the user interface. Changes in appearance will be specified with a change to the minor release code letter. Major revisions that change the way that the Tool works will be indicated with a change to the release number. This will initialise the minor release code letter. This is illustrated in the central column of the above table. The right hand column of the above table shows the effect of combining changes to the release number with changes to the Framework Architecture Version Number.

A change log for the Browsing Tool will be provided as part of the Tool itself. This will only relate to changes since the previous release, and will not provide a complete historical record of the changes made to produce successive releases.

The Browsing Tool will only display all of the parts that are used in the current Version of the European ITS Framework Architecture. It will **not** be able to display parts that are no longer used by the current, or previous Versions.

3.7 Selection Tool

New Versions of the Selection Tool will be produced independently of the Framework Architecture. There should be no necessity for the Selection Tool to be updated with each new Version of the Framework Architecture. Therefore the Selection Tool Version numbers will be independent and will follow the convention adopted to that for the European ITS Framework Architecture – see section 3.3.

Note that the Selection Tool will not work with Versions 1 and 1.1 of the European ITS Framework Architecture. Any other incompatibilities will be highlighted in the Selection Tool itself and/or in its User Documentation.

4. Configuration Management practice for Users creating European ITS Architectures

4.1 Introduction

This section describes the Configuration Management practice that should be adopted by all Users of the European ITS Framework Architecture. It should be applied to the creation of Framework, Mandated and Specific ITS Architectures. These are defined in section 2.2.3 and described in Appendix 2.

Adherence to the contents of this section will help to ensure that Configuration Management of the European ITS Framework Architecture is maintained and prevent developments of ITS Architectures by individual Users being confused with each other. The subject of the documentation for the user's ITS Architectures is covered by Section 5 of this Document.

4.2 Changing User Needs and the Functional Viewpoint

4.2.1 Introduction

European ITS Framework Architecture Users will be creating ITS Architectures to suit their own needs. They will therefore want to adapt the User Needs and Functional Viewpoint in the Framework Architecture. This can be achieved by modifications, which may involve additions or deletions. The ways in which it is recommended that Configuration Management be achieved for the User Needs and parts of the Functional Viewpoint is described in the following sections.

In the following sections it is assumed that "Users" are those defined in the first paragraph of section 1.3. The recommendations are for changes made to the European ITS Framework Architecture to produce the ITS Architectures that are required by the Users. Thus references to "List of User Needs" and "Functional Viewpoint" are for those that are being created as part of these ITS Architectures.

4.2.2 User Needs

Users should apply the following principles to any modifications that they make to the List of European ITS User Needs when creating their own ITS Architectures. The changes should ensure that the ITS Architecture has all the User Needs required to support the Services that are to be provided by the ITS deployment.

2. A User Need that is not required should be deleted from the List of User Needs for the ITS Architecture. All references to it should be deleted from all Functions, which may mean that some Functions can also be deleted – see point 1 in section 4.2.5.

3. Any new User Need(s) should be added within the existing structure. It should take the next appropriate number for the most appropriate

Group, or Service, etc. This may require the addition of new Functions – see point 2 in section 4.2.5.

4. If a new User Need duplicates an existing User Need, then the new and existing (duplicate) User Needs should each show the new reference(s) to the other(s).

5. Allocations of User Needs to Functions may be changed, providing that this does not affect the contents (functionality) of the Functions.

Further information about the creation of the European ITS User Needs can be found in reference (4). The allocation of User Needs to Functions for the European ITS Framework Architecture can be found in the tables contained in reference (5).

4.2.3 Terminators

Users should apply the following principles to any modifications that they make to the Terminators when creating their own ITS Architectures from the European ITS Framework Architecture.

1. Additional representations of the "outside" World should normally be made using extra Actors within the existing Terminators. This should be perfectly possible since all of the existing Terminators have been made "generic" so that Actors can be easily added.

2. The names of new Terminators and Actors will be in English – see also section 4.3.3(c).

3. Existing Actors and Terminators in the Framework Architecture that are not used should be deleted. Note that this will require the deletion of their Data Flows and any Functions that serve them – see sections 4.2.4, 4.2.5 and 4.2.6.

4. Terminators from the European ITS Framework Architecture that are no longer needed because they have been replaced or deleted should not be shown in the revised versions of the Context Diagram for the ITS Architecture. Adopting this practice should avoid making the Context Diagram over complicated and confusing.

Further information about the existing Framework Architecture Terminators and their Actors can be found in Chapter 4 of reference (6). This also includes a discussion of the concept of Terminators and their role in ITS Architectures.

4.2.4 Terminator Data Flows

Users should apply the following principles to any modifications that they make to all types of Terminator Data Flows when creating their own ITS Architectures from the European ITS Framework Architecture. These modifications should be a direct result of those made to the Terminators themselves – see previous section.

1. The Data Flows for existing Terminators and Actors that are deleted should also be deleted – see point 3 in section 4.2.3. When this is done, the Functions to which the Data Flows are linked will need to be modified – see section 4.2.5.

2. If the deleted Terminator Data Flow is a component from another existing Data Flow, then the description of the remaining (parent) Data Flow must be updated.

3. All new Terminator Data Flows should take names that have not been used before – see section 3.4.9 for the identification of previously used names. These must conform to the naming conventions for the particular type of Terminator Data Flow of which they are a part and use the acronym for the Terminator or Actor – see also section 4.3.3(c).

5. If a new Terminator Data Flow is added then its name should conform to the existing hierarchy for Terminator Data Flows. This means that unless a new Terminator has been added – see previous section, the new Data Flow should be added as a component of an existing High-level Data Flow. The existing conventions for high-level and medium-level Data Flows must be followed. Thus for example each new high-level Data Flow must have a new medium-level Data Flow for each Functional Area that contains an interface to the new Terminator.

6. Terminator Data Flows from the European ITS Framework Architecture that are no longer needed because they have been replaced or deleted should not be shown in the revised versions of the DFD's to which they belong. Adopting this practice should avoid making the Diagrams over complicated and confusing.

Further information about all types of Terminator Data Flows and a description of their naming conventions can be found in reference (8).

4.2.5 Functions

Users should apply the following principles to any modifications that they make to the Functions in the Functional Viewpoint of the European ITS Framework Architecture. These modifications will be driven by those made to the User Needs and Terminators – see previous sections.

1. Delete any Functions in the Framework Architecture that are not required to provide the functionality for the Services to be supported by the ITS Architecture that is being created. These Functions will be those that have had all of the User Needs allocated to them removed as a result of the work in point 1 of section 4.2.2. All of the input and output Data Flows for the Functions should also be deleted – see sections 4.2.4 and 4.2.6.

2. A new Function should be created if none of the functionality in the existing European ITS Framework Architecture can serve any new User Needs that have been created for new Services – see section 4.2.2.

3. A new Function should be created from an existing Function if it needs to be modified in any of the following ways.

   (a) The functionality needs to be modified dues to changes in the allocation of existing User Needs, or new User Needs are added – see section 4.2.2. The modification may be through either addition or deletion of functionality.

   (b) Input or output Data Flows are added or deleted to or from the Function. Note that this will also require a change to the functionality.

   (c) A Function is found to be too complex when the Physical Viewpoint is being created. The evidence for this will be that unwanted functionality is being needlessly included in a Sub-system or Module. In this instance the particular Function will be turned into a High-level Function. The new and existing functionality must be included in two or more Low-level Functions, whose parent is the new High-level Function.

4. If a High-level Function is created from a Low-level Function, as in 2(c) above then it can retain the same number if its input and output Data Flows are unchanged. Otherwise the Function must be re-numbered – see next point.

5. With the exception of the provisions in point 4, all new Functions created through points 2 and 3 must take numbers and names that have not been used before – see also section 4.2. The number must be the next new number in the sequence for the Functional Area in which the Function resides. The name will always include a "verb" to show that it "does something" – see also section 3.4.9 for the identification of previously used names.

6. Wherever possible, the creation of new Functions should not require the creation of a new Functional Area. The new Functions should be added to those in the appropriate existing Functional Area.

7. Functions from the European ITS Framework Architecture that are deleted should not be shown in the revised versions of the DFD's to which they belong. This practice will be followed in order to avoid making the Diagrams over complicated and confusing.

An overview of the functionality and descriptions of all the Data Flow Diagrams in the European ITS Framework Architecture can be found in Chapters 6 to 13 of reference (6). Details of the Function Specifications in the European ITS Framework Architecture can be found in reference (7).

4.2.6 Functional Data Flows

Users should apply the following principles to any modifications that they make to all types of Functional Data Flows in the European ITS Framework Architecture. These modifications will almost always be needed as a result of changes made to Functions – see section 4.2.5, or Data Stores – see section 4.2.7.

1. Existing Functional Data Flows should be deleted if their source and/or destination Functions or Data Stores have been deleted.

2. If an existing Functional Data Flow that is being deleted is a component in another existing Data Flow, then the name of the remaining (parent) Data Flow must be changed and its description updated.

3. All new Functional Data Flows should take new names that have not been used before – see section 3.4.9 for the identification of previously used names. These must conform to the naming conventions defined in Chapter 2 of reference (8) – see also section 4.3.

4. If a new Functional Data Flow is added as a component of an existing Data Flow, then the name of the existing (parent) Data Flow must be changed and its description updated.

5. All new Functional Data Flows should have a single source and destination Function or Data Store. Data Flows to or from Low-level Functions must have a source or destination that is another Low-level Function. This means that the same Data Flow must not be used as the source or destination of two different Functions.

6. If a Data Flow is added to or removed from the list of Input or Output Data Flows for a Low-level Function or a Data Store, then they must be given a new name and number that has not been used before – see sections 4.2.5 and 4.2.7.

7. Functional Data Flows from the European ITS Framework Architecture that are no longer needed because they have been replaced or deleted should not be shown in the revised versions of the DFD's to which they belong. Adopting this practice should avoid making the Diagrams over complicated and confusing.

Further information about all types of Functional Data Flows in the European ITS Framework Architecture and a description of their naming conventions can be found in reference (8).

4.2.7 Data Stores

Users should apply the following principles to any modifications that they make to the Data Stores in the European ITS Framework Architecture. These modifications will be made as a result of those being made to the User Needs and Functions – see sections 4.2.2 and 4.2.5.

1. Data Stores from the European ITS Framework Architecture not required by the ITS Architecture should be deleted. This will be because the Functions that provide or receive data to or from them have been deleted – see section 4.2.5.

2. When a Data Store is deleted, all of its input and output Functional Data Flows must also be deleted. Note that their names should not be re-used – see point 4 in section 4.2.6.

3. A new Data Store should be created to serve new User Needs and (in consequence) new Functions. It should take the next new number in the sequence for Data Stores in the Functional Area in which it resides. It must also be given a name that has not been used before – see sections 3.4.9 and 4.3.3.

4. Any Functional Data Flows created to provide access to/from a new Data Store must also be given names that have not been used before – see section 3.4.9 and points 3 and 4 in section 4.2.6. Different names should be used for Data Flows to and/or from each Function – see also point 6 in section 4.2.6.

5. The creation of new Data Stores may require the creation of one or more new Functions. These new Functions should be added to those in the appropriate existing Functional Area – see section 4.2.6, without creating a new Functional Area.

6. Data Stores from the European ITS Framework Architecture that are no longer needed because they have been replaced or deleted should not be shown in the revised versions of the DFD's to which they have belonged. This practice should avoid making the DFD's over complicated and confusing.

Further information about the existing Data Stores in the European ITS Framework Architecture and their naming convention can be found in reference (9).

4.3 Numbers and Names

4.3.1 Introduction

Users need to recognise that there may be several individual ITS Architecture developments going on at any instant in time. They may only be aware of some of them, and may not always know the current extent and scope of the work being done. The work may also be going on at the same time as the work that they are doing.

To guard against the possible duplication of any part of the ITS Architectures being developed from the Framework Architecture, the conventions in the following section have been developed. It is strongly recommended that Framework Architecture Users adopt them.

4.3.2 Numbers

This convention applies to User Needs, Functions and Data Stores that are added by Framework Architecture Users for ITS Architectures they are creating.

(a) Use the next number in the sequence assigned to Users for the part of the European ITS Framework Architecture in which the new item resides. The starting numbers for each part of the Architecture are shown in the following table. They have been assign specific values that should ensure that they are not confused with any extra User Needs, Functions and Data Stores that are added by the Maintainers of the European ITS Framework Architecture.

Table 4 Users' Numbering starting points for parts of the European ITS Framework Architecture

Part	Numbering Convention		User's Starting Number
	Format	Description	
User Needs	N.M.P.Q	Where, N = Group	21
		M = Service (based on ISO list) within an existing Group.	21
		P = Topic within an existing Service	21
		Q = Unique Number within an existing Topic	101
Functions	A.B.C.D…	Where, A = Functional Area	21
		B = Function Number within the Functional Area. For nearly all Functional Areas this is usually a High-level Function.	21
		C = Function Number, within an existing High-level Function	21
		D = Function Number, within an existing High-level Function	21
		… = Function Number, within an existing High-level Function	21

Part	Numbering Convention		User's Starting Number
	Format	Description	
Data Stores	A.X	Where, A = Functional Area	21
		X = Data Store Number within the Functional Area	21

(b) Add a country specific suffix to the number – see section 4.3.4. This guards against the same number being used by ITS Architectures being developed from the Framework Architecture in other countries. If necessary, add extra letters to identify a State, County, City, District or other local area – see section 4.3.5.

An example of the way that Users should create and use new numbers is provided in Appendix 3 of this document.

If the ITS Architecture is being created for more than one country, then a specific letter suffix must be used. This could be the initial letters of the organisation creating the ITS Architecture, or the combined suffixes for the countries that are involved.

4.3.3 Names

This convention applies to Terminators, Actors, Functional Areas, plus all types of Data Flows that are added by Framework Architecture Users for ITS Architectures they are creating.

(a) Use a name that has not been used before – see section 3.4.8 for notes on the selection of new names.

(b) The new names of Functions and Data Stores should be in the language being used for the ITS Architecture. If it is intended to include the new Functions or Data Stores in a future European ITS Framework Architecture Update Request then an English version of the name should also be produced. This will make it easier to integrate them into future new Versions of the Framework Architecture. A country specific suffix is not required for the name as it will have been included in the number – see section 4.3.2.

(c) The new names of Terminators and Actors should be in the language being used for the ITS Architecture. If it is intended to include the new Functions or Data Stores in a future European ITS Framework Architecture Update Request then an English version of the name should also be produced. This will make it easier to integrate them into future new Versions of the Framework Architecture. The new names must have a country specific suffix added to the end of the name – see section 4.3.4. This guards against the same names being used by ITS Architectures being developed from the Framework Architecture in different countries. If necessary, add extra letters to identify a State, County, City, District or other local area – see section 4.3.5.

(d) The new names of Data Flows should be in the language being used for the ITS Architecture. If it is intended to include the new Functions or Data Stores in a future European ITS Framework Architecture Update Request then an English version of the name should also be produced. This will make it easier to integrate them into future new Versions of the Framework Architecture. The new names must have a country specific suffix added to the end of the name – see section 4.3.4.

(e) The description or definition of User Needs, Terminators, Actors, Functions, all types of Data Flows and Data Stores can be written in the local language.

If it is not possible to create and show the new names in English, Users must provide an English equivalent of the name, if they are submitting European ITS Framework Architecture Update Requests. This will help the Framework Architecture Maintainers to identify equivalences to other ITS Architecture developments and help with the creation of new Versions of the Framework Architecture.

All suffixes must use upper case (capital) letters. This should make them obvious when looking at the ITS Architecture documentation.

4.3.4 Country Codes for Suffixes

In all cases ISO-3166 2-letter country codes (see reference 9) should be used in the suffixes for new names. For Europe these currently include the following.

AT	Austria	GB	Great Britain	PL	Poland
BE	Belgium	GR	Greece	PT	Portugal
CZ	Czech Republic	HU	Hungary	RO	Romania
ES	Spain	IE	Ireland	SI	Slovenia
DK	Denmark	IT	Italy	SV	Sweden
DE	Germany	LU	Luxembourg	CH	Switzerland
FI	Finland	NO	Norway		
FR	France	NL	The Netherlands		

4.3.5 Additional Suffix Code Letters

The additional identification letters added to the above for State, County, City District or other local area, should be taken from the second part (Country sub-division code) of ISO 3166 – see reference (10), or if no suitable code exists, chosen freely. However, if several ITS Architectures are likely to be created within a single country, then a suitable local (country) convention should be adopted.

4.4 Compatibility Issues

4.4.1 Introduction

There are three different compatibility issues that will affect the development of ITS Architectures. Firstly there is compatibility with previous versions of the ITS Architectures developed by particular Users. Secondly there is compatibility of ITS Architectures with the European ITS Framework Architecture. Thirdly there is compatibility between ITS Architectures developed by different Users. Each should be considered separately.

4.4.2 Compatibility with previous versions of ITS Architectures

Adoption of the practices described in the previous sections (4.2. and 4.3) is intended to help Users keep their own ITS Architectures compatible with the current version of the European ITS Framework Architecture. It should also help them maintain "backwards compatibility"[7] for their own ITS Architectures. This will be just as important as for the European ITS Framework Architecture, particularly when Users are creating their own Framework Architectures – see section 2.2.3 and Appendix A1.

4.4.3 Compatibility with previous versions of the European ITS Framework Architecture

The practices described in section 3 for the Maintainers of the Framework Architecture will help them to provide "backwards compatibility" when it is updated to a new Version. However a new Version of the Framework Architecture may not be automatically compatible with ITS Architectures that Users have developed from earlier Versions. This is because the Framework Architecture Maintainers have no control over what is included in the ITS Architectures that Users develop for themselves. For example Users can chose to include functionality that supports some or all of the Services included by the Framework Architecture. They can also add an unlimited amount of functionality to support their own particular Services or provide more detailed functionality for the existing Services. Thus it is possible that a particular ITS Architecture although retaining its compatibility with an earlier Version, may not be compatible with the latest Version of the European ITS Framework Architecture. It will be up to Users to make any necessary changes to their own ITS Architectures to make them compatible with the latest Version.

4.4.4 Compatibility between ITS Architectures

Compatibility between ITS Architectures is an issue that only the Users that develop them can resolve. Adoption of the practices described in sections 4.2 and 4.3 will help, but ultimately it will be up to the Users themselves to resolve any issues of incompatibility between their own ITS Architectures.

[7] See Note 2.

4.5 Other Points

4.5.1 Introduction

The following are some other points that should be taken into account by Framework Architecture Users, when creating their own ITS Architectures. Their use will help to make the task of Configuration Management easier.

4.5.2 Translation from English

As noted in point 3 of section 2.2.4, there will be a strong and very natural desire for Framework Architecture Users to translate it into their own native language. However it is recommended that the names of Terminators, Actors, Functional Areas, Functions, all types of Data Flows and Data Stores in ITS Architectures be retained in their English form. One reason for this is that these names appear in DFD's and by adhering to English it will make comparisons with other ITS Architectures easier.

If User Need descriptions are translated from English to another language, this should be done by adding an extra column to the spreadsheet in the List of European ITS User Needs – see reference (3). The existing column with the English version can then be "hidden" and used for reference only. Adopting this practice will also help if new User Needs are incorporated into the List of European ITS User Needs as it provides locations for both the national language and English versions of their descriptions.

4.5.3 Translation to English

Creators of ITS Architectures should note that both the names and descriptions of any new parts that they create for their Architectures (Terminators, Actors, Functional Areas, Functions, all types of Data Flows and Data Stores) might be translated into English. The translation will be carried out when any of the new parts are being incorporated into a new Version of the European ITS Framework Architecture. The purpose of the translation will be to ensure that the Framework Architecture is defined and described in a consistent way.

The translation will be organised by the European ITS Framework Architecture Maintainers as part of its update to a new Version. The creators of ITS Architectures from which the new parts are taken will be consulted to try and ensure that the translation is correct.

4.6 Physical, Communications and other Viewpoints

No Configuration Management practices have been developed for the creation of Physical, Communications and other Viewpoints that may be included in the ITS Architectures created by some European ITS Framework Architecture Users. This is because these Viewpoints do not exist for the Framework Architecture.

It is strongly recommended that the creation of the Physical and Communications Viewpoints follow the recommendations in the Framework Architecture documentation – see references (11) and (12) respectively. Similar information can

also be found in part of the material provided for the second of the FRAME Training Workshops, “Creating and Using an ITS Architecture”.

When applying these recommendations, some of the Configuration Management practices described in Sections 3 and 4 should be followed. For example, the need to maintain “backwards compatibility”[8] needs to be determined, as does the parts of the Viewpoints that will be included in their Configuration Management practices.

Configuration Management has no real relevance to most of the other ITS Architecture creation process outputs, such as Deployment Studies, Cost Benefit Analysis and Risk Studies. The best way of controlling the implementation of different Versions and upgrades is to use conventional document management practices. The exception is the Organisational Viewpoint, in which it may be possible to apply some of the Configuration Management practices that are described in Sections 3 and 4 of this Document.

[8] See Note 2.

5. A Guide for creating documentation for ITS Architectures based on the European ITS Framework Architecture.

5.1 Introduction

This Section of the document provides a guide for producing ITS Architecture documentation, in terms of its scope and content. It is aimed at those who are creating ITS Architectures that are based on the European ITS Framework Architecture.

5.2 General Points

The structure of the documentation used to describe ITS Architectures should follow that of the European ITS Framework Architecture. This means that all the Viewpoints should be described in separate documents, as should the User Needs. If an "Overview" document is not required – see reference (1), then the Trace Tables should be made into a document on their own.

5.3 Scope of the ITS Architecture documents

5.3.1 Introduction

The scope of the documents will depend very much on the type of ITS Architecture that is being produced. This may be one of the three types that are identified in section 2.2.3 and described in more detail by Appendix 2. Figure 1 in Appendix 2 illustrates the relationships that they have with each other. The scope of the documentation that they each need is described in the following sections.

5.3.2 Framework ITS Architecture

This type of ITS Architecture will probably be used by those producing ITS Architectures for National or Regional use – see Appendix 2.2. Its purpose will be to act as the starting point for the development of Specific ITS Architectures – see section 5.3.4. The documentation will therefore need to include the following subjects.

User Needs – provides a list of all those that cover the envisaged scope of ITS deployment in the Nation or Region.

Functional Viewpoint – describes the functionality needed to provide the Services identified in the User Needs.

Physical and Communications Viewpoints – describes how these two Viewpoints should be created. This should include details about how the Component and Infrastructure Specifications should be written and be accompanied by supporting examples. If National or Regional conformity is in presentation and/or content required then templates for any diagrams, documents or specifications should also be included.

Using the ITS Architecture – describes how the Framework ITS Architecture should be used, including guidance on the development of

other things such as an Organisational Viewpoint, Deployment Plan, etc. A list of the European Standards that should be used for such things as communications and human interfaces may also be included. Again if National or Regional conformity in presentation and/or content is required then templates should also be included.

The documentation for this type of ITS Architecture should be very similar to that of the European ITS Framework Architecture. This is because they are intended to serve the same purpose – act as the starting point for the development of other ITS Architectures.

The degree of flexibility that Users have to modify the Framework ITS Architecture to suit their own ITS deployments should be clearly indicated in the documents and accompanied by advice on how it should be done. There should also be a clear description of how Users must document the relationships that their ITS Architectures have with the parent Framework ITS Architecture and ultimately with the European ITS Framework Architecture.

5.3.3 Mandated ITS Architecture

This type of ITS Architecture will probably be used by those producing ITS Architectures for National, Regional or Local (City) use where limited flexibility is required over the way in which ITS will be deployed – see Appendix 2.3. Its purpose may also be to act as the starting point for the development of Specific ITS Architectures – see section 5.3.4. The documentation will therefore need to include at least the following:

User Needs – provides a list of all those that cover the envisaged scope of ITS deployment in the Nation, Region, or local geographic area.

Functional Viewpoint – describes the functionality needed to provide the Services identified in the User Needs.

Physical Viewpoint – describes the distribution of the functionality between the available physical locations. This should include a description of the functionality in each location, written in such a way that Component Specifications can be easily produced from them. There must also be a description of the flexibility that has been included in the Physical Viewpoint, with examples of how it can be used. If National or Regional conformity in presentation and/or content is required for the Specifications then templates should also be included.

Communications Viewpoint – describes the mechanisms that are to be used by the Sub-systems and Modules in the Physical Viewpoint to enable them to communicate with each other and with the outside World. The descriptions should be written in such a way that Infrastructure Specifications could easily be produced from them. If flexibility has been included in the Physical Viewpoint then supporting examples should be provided to show how this is to be used. Again if National or Regional conformity in presentation and/or content is required for the Specifications then templates should also be included.

Using the ITS Architecture – describes how the Mandated ITS Architecture should be used, including guidance on the development of

other four outputs. These comprise the Organisational Viewpoint, Deployment Plan, Cost Benefit Report and Risk Analysis. Again if National or Regional conformity is required for any diagrams, documents or specifications then templates should also be included.

The degree of flexibility that is included in the Physical Viewpoint for this type of ITS Architecture will be a matter of choice for its owners and creators. Typically the flexibility will be provided through the creation of alternative Sub-systems and/or Modules that Users may select for particular deployments that are to be supported by the ITS Architecture. A list of the European Standards that should be used for such things as communications and human interfaces may also be included. It may be possible to produce the Organisational Viewpoint, but this will depend upon the circumstances in which Architecture will be used and (again) the degree of flexibility included in the Physical Viewpoint.

5.3.4 Specific ITS Architecture

This type of ITS Architecture will probably be used by those producing ITS Architectures for Regional or Local use – see Appendix 2.4. Its purpose will be to describe the way in which the functionality for a particular Service (or set of Services) is to be deployed. The documentation will therefore need to include the following:

User Needs – provides a list of all those that are included in the Service(s) to be deployed.

Functional Viewpoint – describes the functionality needed to provide the Service(s) identified in the User Needs.

Physical Viewpoint – describes the distribution of the functionality between the available physical locations. This should include a description of the functionality in each location and all of the Component Specifications.

Communications Viewpoint – provides the description of the data that is to be transferred between the locations in the Physical Viewpoint, plus the Infrastructure Specifications. The Specifications should include the required link characteristics for use by telecommunications providers, with special reference to any standards that must be complied with.

Deployment Plan – contains a description of how and when the components and infrastructure are to be deployed, including the effects on any existing ITS components.

Cost Benefit Report – shows the results from an analysis of the costs and expected benefits that will result from deployment of what is in the Architecture.

Organisational Viewpoint – provides a description of the relationships between the organisations that will be involved in the deployment of what is in the Specific ITS Architecture. The organisations may represent end users, internal users, owners, operators and regulators of the Architecture.

Risk Analysis – this should be carried out for the deployment of what is included in the Physical, Communications and Organisational Viewpoints, plus the contents of the Deployment Plan and Cost Benefit Analysis. The

need for the results to be documented will depend on what risks have been identified and their impact. It is recommended that the RAID report in reference (13) be studied before the analysis is started[9].

5.3.5 General Comments

The documentation that is produced in accordance with what has been described above should be based on that available for the Framework or Mandated ITS Architecture from which the Specific ITS Architecture was created.

The Specific Architecture is the lowest type of ITS Architecture that can be derived from the European ITS Framework Architecture. This is because no other ITS Architectures can be derived from it – see Figure 1 in Appendix 2.

There may be one or more Specific ITS Architectures created from a particular Framework ITS Architecture. Also a Mandated ITS Architecture could be created at the National level with sufficient flexibility to enable Specific ITS Architectures to be created at the Regional and/or City level.

5.4 References to the European ITS Framework Architecture

5.4.1 Introduction

Wherever possible, the documents describing ITS Architectures should include references to the European ITS Framework Architecture. These references should be made in the ways that are described in the following sections.

5.4.2 Documentation References

The identity of specific parts of the Framework Architecture documentation that must be read in conjunction with the ITS Architecture documentation should be detailed. This should be done in preference to copying any parts of the text from the Framework Architecture documentation. Adopting this strategy will help to ensure that any changes made for later Versions of the Framework Architecture are automatically included.

At first sight this may appear to conflict with the need to provide the ITS Architecture documentation in the native language of its target audience. However, unless there is an absolute imperative to provide a translated version of the European ITS Framework Architecture document text, it is recommended that references be used. Any translation should be a précis of the original English to simplify the need for modifications if the original English text is changed in a new Version of the Framework Architecture.

[9] It is also recommended that any documents on "Impact Analysis" produced by the FRAME Projects should be consulted as part of the work on the Risk Analysis. This will help to ensure that other issues such as Strengths, Weaknesses, Opportunities and Threats are covered.

5.4.3 Links between parts of the two Architectures

A table showing the relationship between parts of the user developed ITS Architecture and the corresponding parts of the European ITS Framework Architecture should be created. The table should provide a list of the following.

1. The parts of the European ITS Framework Architecture that have been completely replicated in the ITS Architecture. This means that they have been copied without any alteration, excluding any changes in language.
2. The parts of the European ITS Framework Architecture that have been used as a starting point for the development of specific parts of the ITS Architecture. The entries in the list for these items should include a brief description of the role in the ITS Architecture development that has been played by the European ITS Framework Architecture.

The purpose of implementing these two points is to enable a quick and easy assessment of the impact that has been made by changes to the European ITS Framework Architecture. The table can be provided either as several tables, one in each relevant ITS Architecture document, or as a single table in a specific ITS Architecture document.

6. References

The following references have been used in this Document. Unless otherwise stated, they all point to electronic versions of the European ITS Framework Architecture Documents. These can be downloaded from the "LIBRARY" page of the FRAME Web Site at: http://www.frame-online.net , unless another source is given.

(1) **European ITS Framework Architecture Overview, Main Document**. This contains a guide to the other European ITS Framework Architecture documents plus background information about the Architecture.

(2) "**A Guide to Configuration Management for Intelligent Transport Systems**", **US DoT, Report No. FHWA-OP-02-048**. An electronic version of this document can be downloaded as Document No. 13622, from the US DoT Electronic Document Library, which can be found at: http://www.its.dot.gov/itsweb/welcome.htm .

(3) **List of European ITS User Needs, Appendix F. This contains a l**ist of ITS User Needs according to KAREN Groups[10].

(4) **List of European ITS User Needs, Main Document. This provides an i**ntroduction, description of approach adopted, and analysis.

(5) **European ITS Framework Architecture Overview, Annex 1**. This contains Trace Tables to assist in ITS Architecture development.

(6) **European ITS Framework Architecture Functional Viewpoint, Main Document. This contains a** general description of functional needs and explanation of methodology used to create the Functional Viewpoint.

(7) **European ITS Framework Architecture Functional Viewpoint, Annex 1.** This document contains detailed descriptions of all the Functions, including their Specifications.

(8) **European ITS Framework Architecture Functional Viewpoint, Annex 2.** This document contains detailed descriptions of all types of Functional Data Flows and their naming conventions.

(9) **European ITS Framework Architecture Functional Viewpoint, Annex 3.** This document contains detailed descriptions of the Data Stores and their naming conventions.

(10) **Country Code Letters.** These should conform to the international standard for Country Codes, **ISO 3166,** which can be found at**:** http://www.iso.org/iso/en/prods-services/iso3166ma/index.html **.**

[10] A list of ITS User Needs according to TICS Fundamental Services (ISO) numbering can be downloaded as Appendix G from the same source.

(11) **European ITS Framework Architecture Physical Viewpoint, Main Document.** This contains a description of how functionalities can be grouped to form usable systems.

(12) **European ITS Framework Architecture Communications Viewpoint, Main Document.** This contains a description of how to define the communications links required by the data flows in systems.

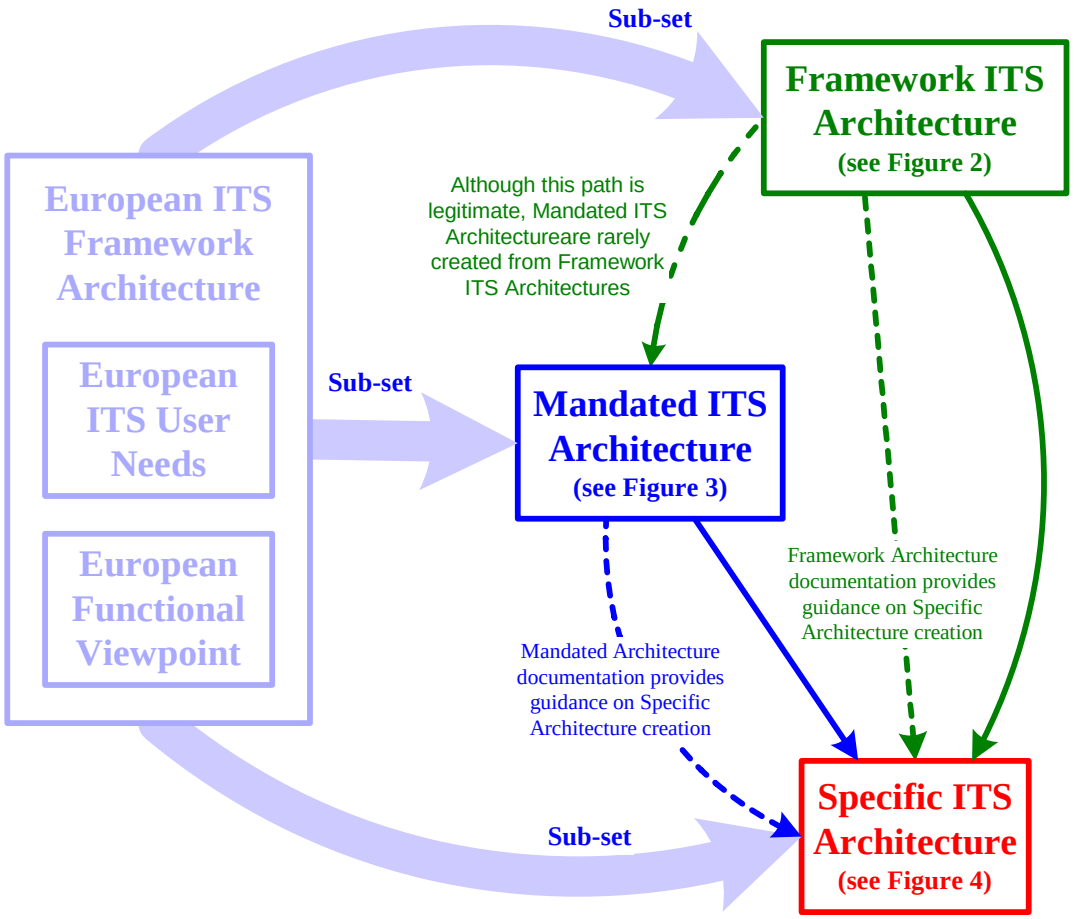
(13) **RAID – Risk analysis for ITS architecture development.** The Main Document describes the approach to the constraints analysis, mitigation strategies and recommendations. Several Annexes are included to provide the results of the analysis and background information.

Appendix 1 *Types of ITS Architectures*

A 1.1 *Introduction*

This Appendix provides a description of what is contained in each of the three types of ITS Architectures that can be created from the European ITS Framework Architecture. It provides an expanded description of what is contained in section 2.2.3. Figure 1 below illustrates the relationship between the three types of ITS Architectures and the European ITS Framework Architecture.

Figure 1 ITS Architecture Relationships

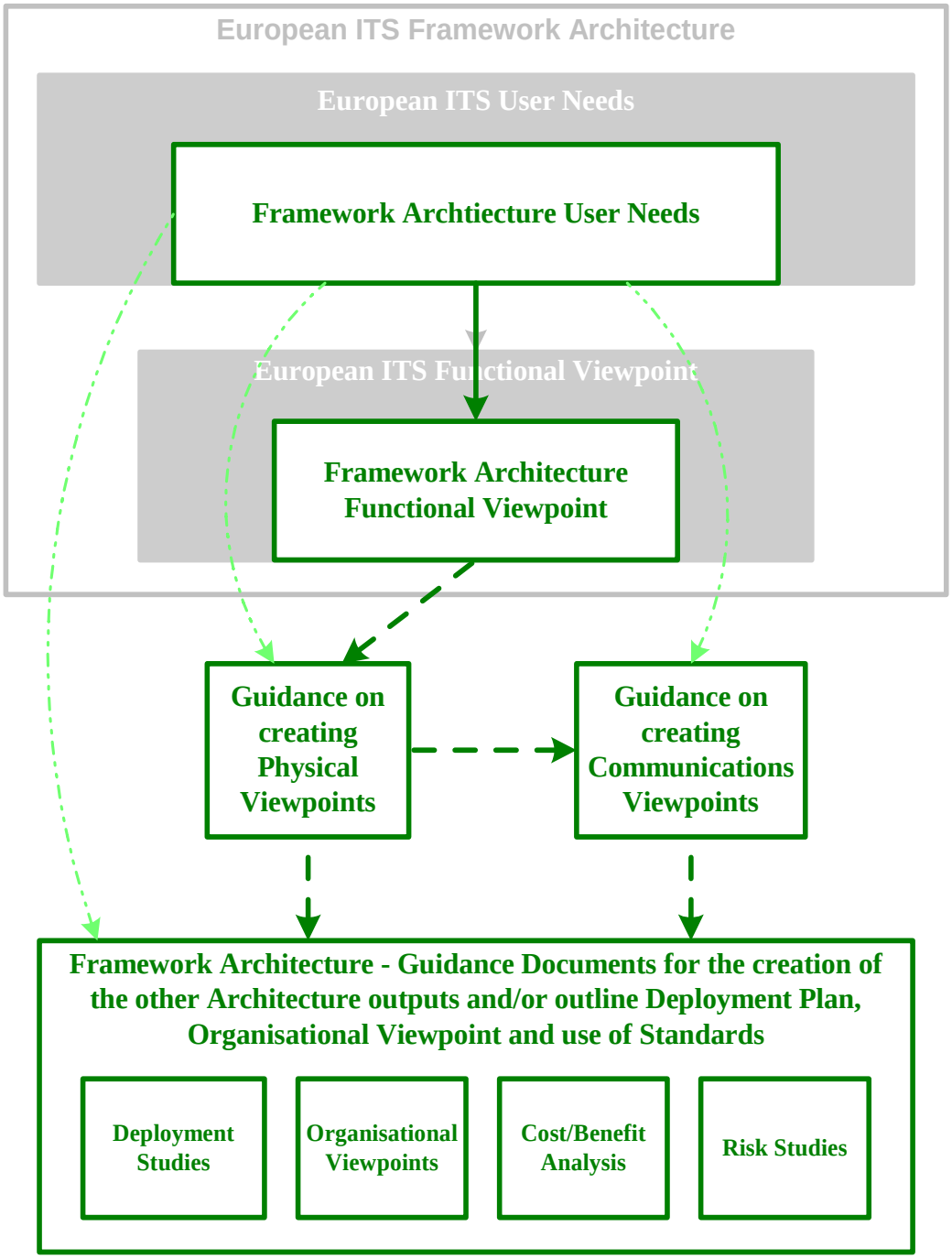


In this diagram the "dotted" lines indicate that the links are through documentation. The "solid" lines indicate that the relationship is through the contents of the Architectures themselves. Those from the European ITS Framework Architecture to each of the three types of ITS Architecture will be achieved using the Selection Tool developed by the FRAME-S Project. Users can create Specific ITS Architectures from Framework and Mandated ITS Architectures using their own tools.

A 1.2 *Framework ITS Architectures*

Framework ITS Architectures are created by National or Regional organisations, and can be used as the starting points for Mandated ITS Architectures, or Specific ITS Architectures. They are based on the European ITS Framework Architecture and are very similar to it in terms of their results. The main difference is that Framework ITS Architectures may only contain a sub-set of the European ITS User Needs, plus (optionally) additional User Needs for any National or Regional specific Services. Figure 2 below illustrates the concept of Framework ITS Architectures.

Figure 2 Framework ITS Architectures



As can be seen in Figure 2 on the previous page, Framework ITS Architectures only contain User Needs and a Functional Viewpoint. The other Viewpoints and all the Architecture creation outputs are not provided. They are replaced by documents that provide guidance about their creation – see part 1 of section 5.3. This is illustrated by the use of "dashed" lines for their creation and output.

The lighter coloured "dash and dotted" lines indicate that there may be specific User Needs relating to some or all of the Physical Viewpoint, Communications Viewpoint and other outputs. These User Needs will be in the group that are additional to those that have come from the parent European ITS Framework Architecture. Also note that the Deployment Studies can be considered in the context of National or Regional ITS deployment.

A 1.3 *Mandated ITS Architectures*

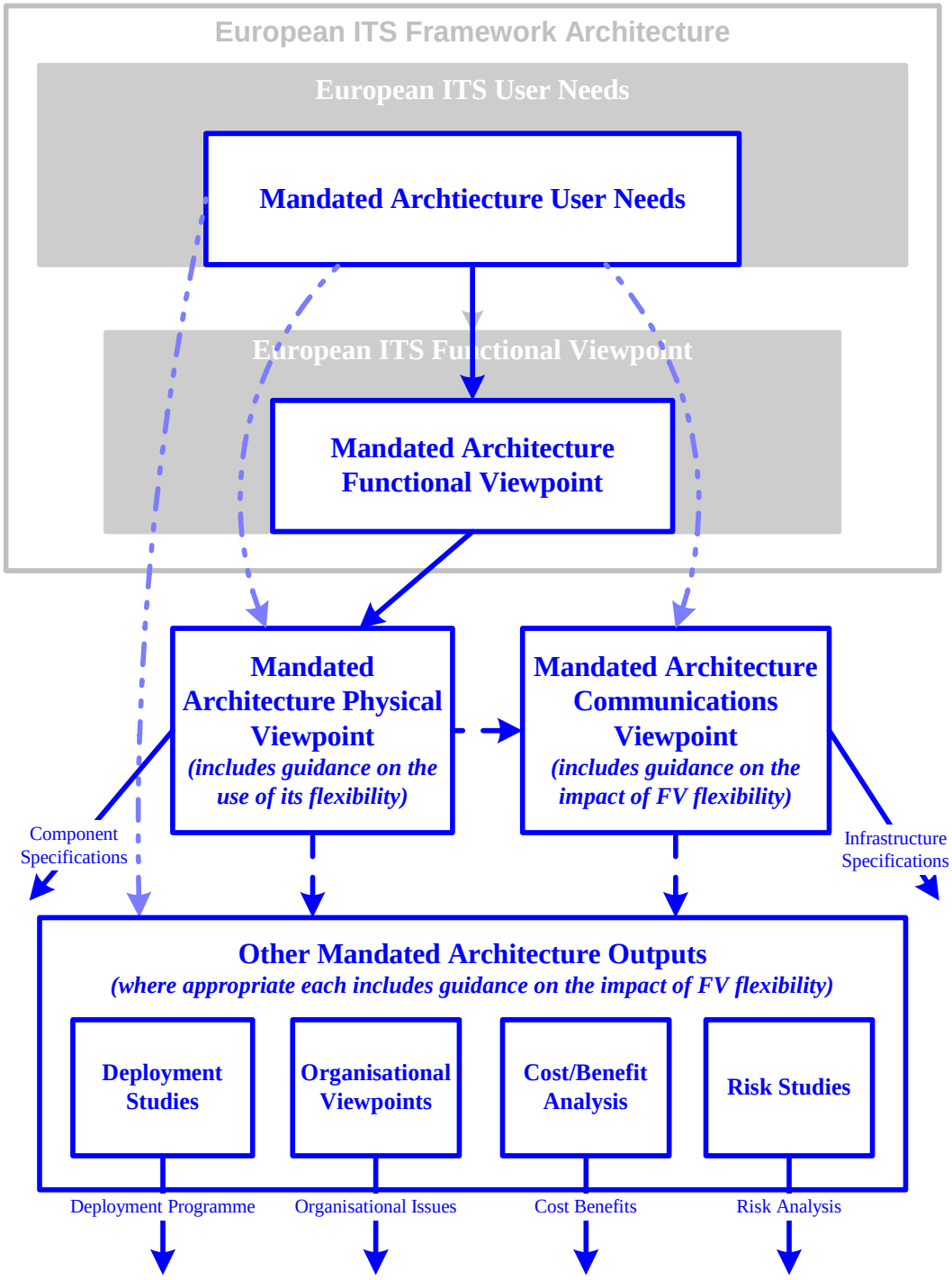
Mandated ITS Architectures are Architectures that describe what is required for the implementation and deployment of specific ITS Services. They may be created for a particular geographic region, e.g. Nation, State, Region, or City. Mandated ITS Architectures take their name from the fact that their use is "mandated" by an organisation that is involved with regulating the use and deployment of ITS. The "mandate" may be in the form of an actual "rule" saying that the Architecture must be used, or a "financial incentive" for its use. This may take the form of funding to support the actual ITS deployment only being available to those that are based on the Architecture. Mandated ITS Architectures can be created from the European ITS Framework Architecture, or a particular National Framework ITS Architecture.

Figure 3 on the next page illustrates the concept of Mandated ITS Architectures. They contain only sub-set of the European ITS User Needs, plus (optionally) additional User Needs for any National, Regional, or City specific Services. However the complete set of User Needs for the Architecture are only those that are relevant for the specific Service(s) that it will support. As may be seen in Figure 1, it is also possible for Mandated ITS Architectures to be created from Framework ITS Architectures, but this creation path is unlikely to be used.

As can be seen, Mandated ITS Architectures contain the Functional and Physical Viewpoints, plus a (possibly) complicated Communications Viewpoint and other Architecture creation outputs. These "(possible) complications" arise because it is possible for a degree of flexibility to be provided in the way that the Physical Viewpoint can be used. This flexibility is achieved by the creation of alternative Sub-systems and/or Modules, containing different functionality and/or in different locations. These enable the creation of either differing System configurations, or Systems deploying different sub-sets of the Services defined by the User Needs. Thus the Communications Viewpoint and the other Architecture creation outputs need to include the same degree of flexibility. Where necessary the documentation of these outputs must highlight the impact that this flexibility will have on such things as the choice of standards for communications interfaces, or the risks to a successful ITS deployment based on the Architecture. Details of the documentation for Mandated ITS Architectures will be found in part 2 of section 5.3.

Guidance must be provided in the Physical Viewpoint documentation about the creation of each possible System configuration within the "flexibility" that has been provided. Advice on the impact of this "flexibility" must be included in the documentation of the Communications Viewpoint and all the other Architecture outputs. This is illustrated by the use of "dashed" lines in Figure 3.

Figure 3 Mandated ITS Architectures



Note that as in the Figure 2, the lighter coloured "dash and dotted" lines indicate that there may be specific User Needs relating to some or all of the Physical Viewpoint, Communications Viewpoint and other outputs. These User Needs will be in the group

that are additional to those that have come from the parent European ITS Framework Architecture. They will be used to define the scope and impact of the "flexibility" in the Physical Viewpoint.

A 1.4 *Specific ITS Architectures*

Specific ITS Architectures are Architectures that describe what is required for the implementation and deployment of specific ITS Services. These Architectures may be created to cover a particular geographic region, e.g. State or City, or by manufacturers wishing to promote their ITS products. Specific ITS Architectures are a "fixed" form of Mandated ITS Architectures and must be supported by some form of "mandate" to ensure that they are used – see Appendix 2.3. They are created from the European ITS Framework Architecture, a particular National Framework ITS Architecture.

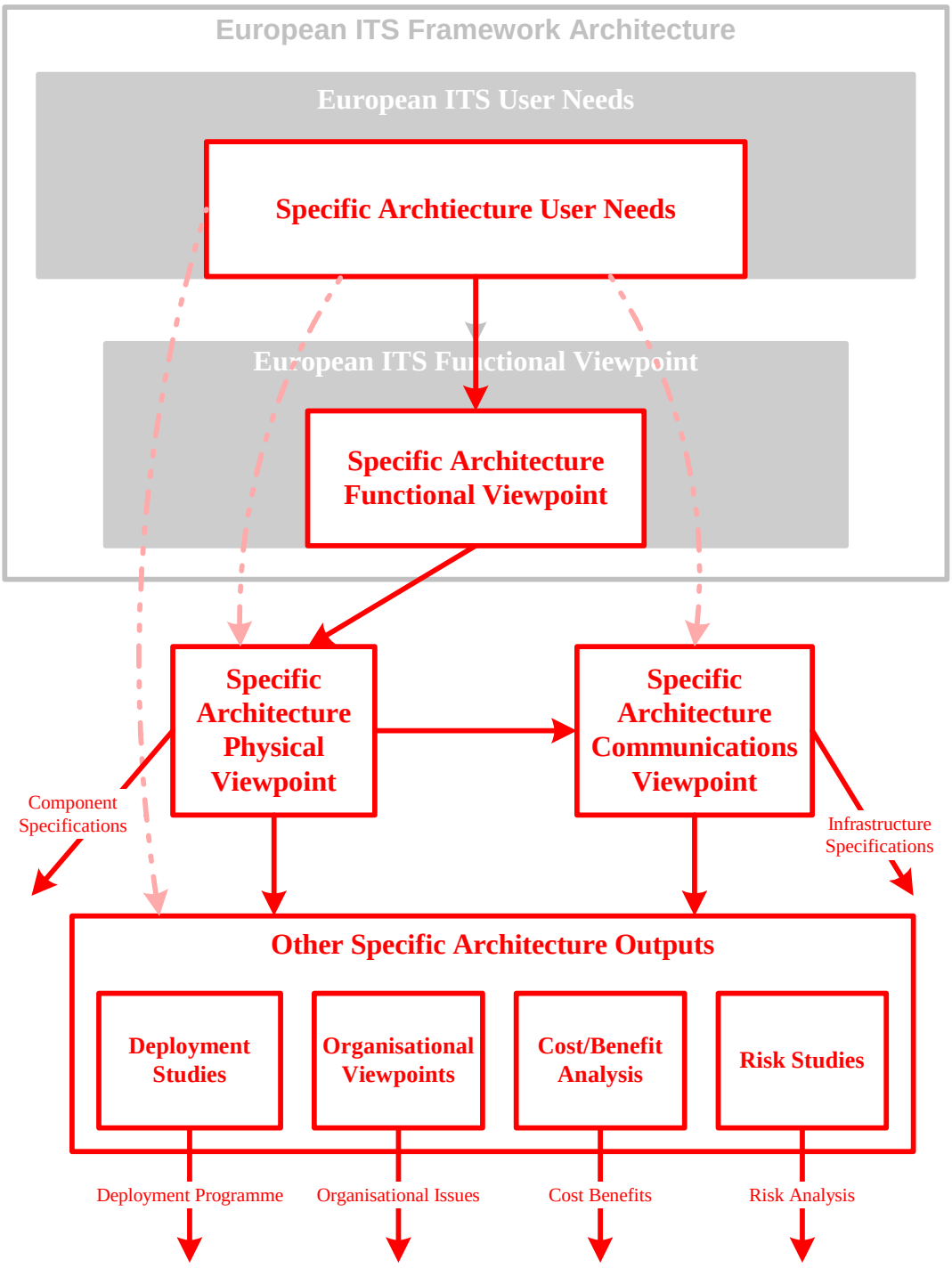
Figure 4 on the next page illustrates the concept of Specific ITS Architectures.

Specific ITS Architectures contain only sub-set of either the European ITS User Needs, or the User Needs of the ITS Architecture from which they have been created. This "parent" ITS Architecture may be either a Mandated or a Framework ITS Architecture. In both cases they may contain additional User Needs for any National, Regional, City, or manufacturer specific Services that are to be supported. However the complete set of User Needs for the Architecture are only those that are relevant for the specific Service(s) that it will support.

As can be seen in Figure 4, Specific ITS Architectures contain all of the Viewpoints, a Deployment Programme, plus the analyses of Cost/Benefit and Risk. These will be used to provide the full set of results comprising, the Component Specifications, the Infrastructure Specifications, the Deployment Plan, the results of analysing any Organisational Issues that may be involved, a Cost/Benefit Analysis, and the results of a study into any Risks to the ITS deployment. The scope of these outputs will be limited to the range of ITS Services that are to be deployed using the Architecture and included in its documentation – see part 3 of section 5.3.

As in Figures 2 and 3, the lighter coloured "dash and dotted" lines indicate that there may be specific User Needs relating to some or all of the Physical Viewpoint, Communications Viewpoint and other outputs. These User Needs will be in the group that have either come from the parent ITS Architecture, or are additional for the Specific ITS Architecture.

Figure 4 Specific ITS Architectures



Appendix 2 *Numbering Conventions for new parts of the European ITS Framework Architecture*

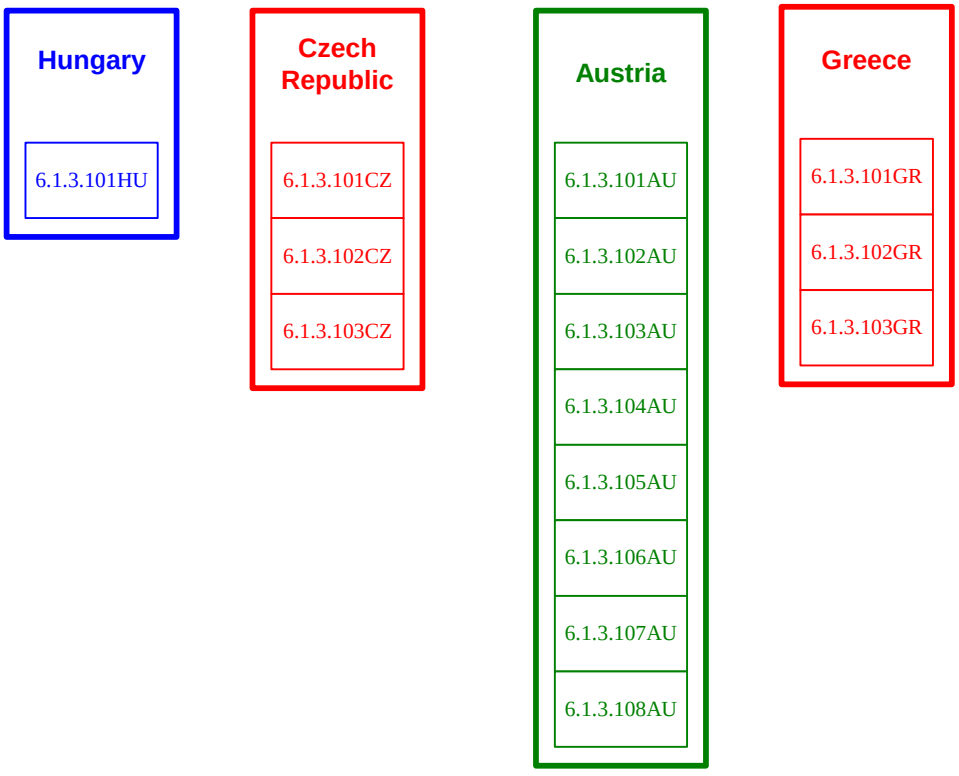
A 2.1 *Introduction*

This Appendix provides examples of the ways that new parts of ITS Architectures must be numbered. The impact on the European ITS Framework Architecture is also described.

A 2.2 *ITS Architecture Development Example 1 – new User Needs*

Four Nations, Hungary, the Czech republic, Austria and Greece are developing their own National ITS Architectures. Each of them is based on the European ITS Framework Architecture. As part of the work in each Nation, new Services have been identified, with the result that some new User Needs must be created and numbered. The four Nations follow the Configuration Management practices for European ITS Framework Architecture Users. These are described in sections 4.2 and 4.3 of this Document. As a result of this work, the situation shown in Figure 5 is created.

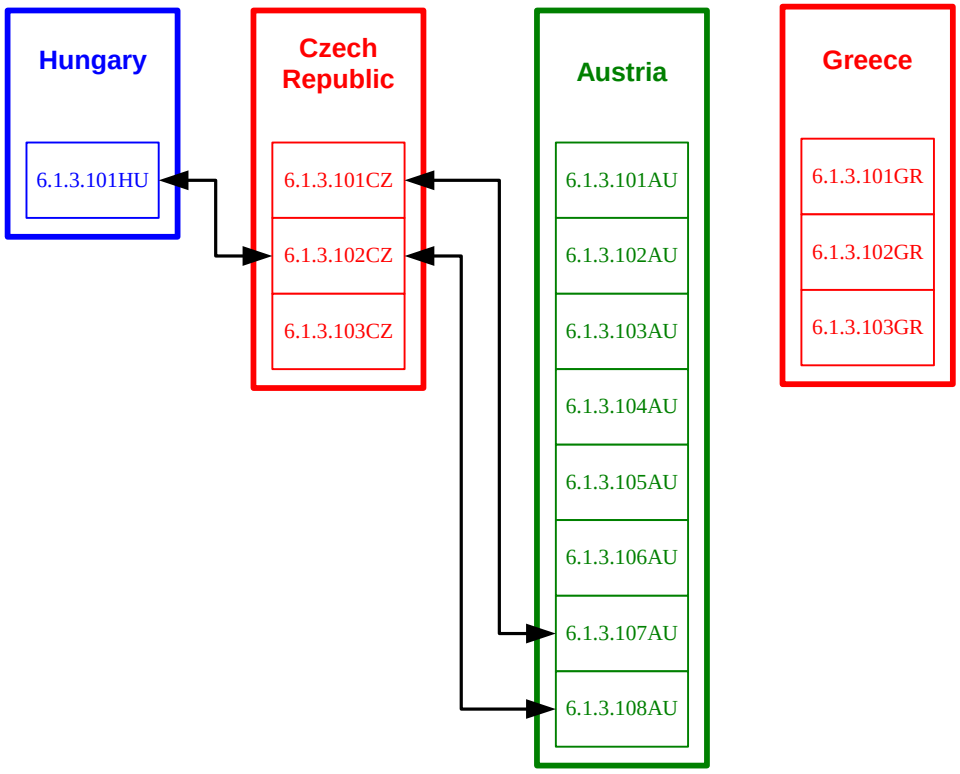
Figure 5 User Configuration Management Practice Example 1 - Part 1



As will be seen from Figure 5, there are four sets of new User Needs, each with new numbers. For each Nation the numbers start from 101 – see Table in section 4.3.2, and include the Nation suffixes, applied in accordance with section 4.3.4.

When each of the Nations is happy that its new User Needs are correct, they submit the details to the Maintainers of the European ITS Framework Architecture, as Update Requests[11]. The "Maintainers" examine what each Nation has done and decide that there is some equivalence between the three of the new sets of User Needs. Figure 6 below illustrates this equivalence through the lines that link some of the new User Needs from each Nation.

Figure 6 User Configuration Management Practice Example 1 - Part 2



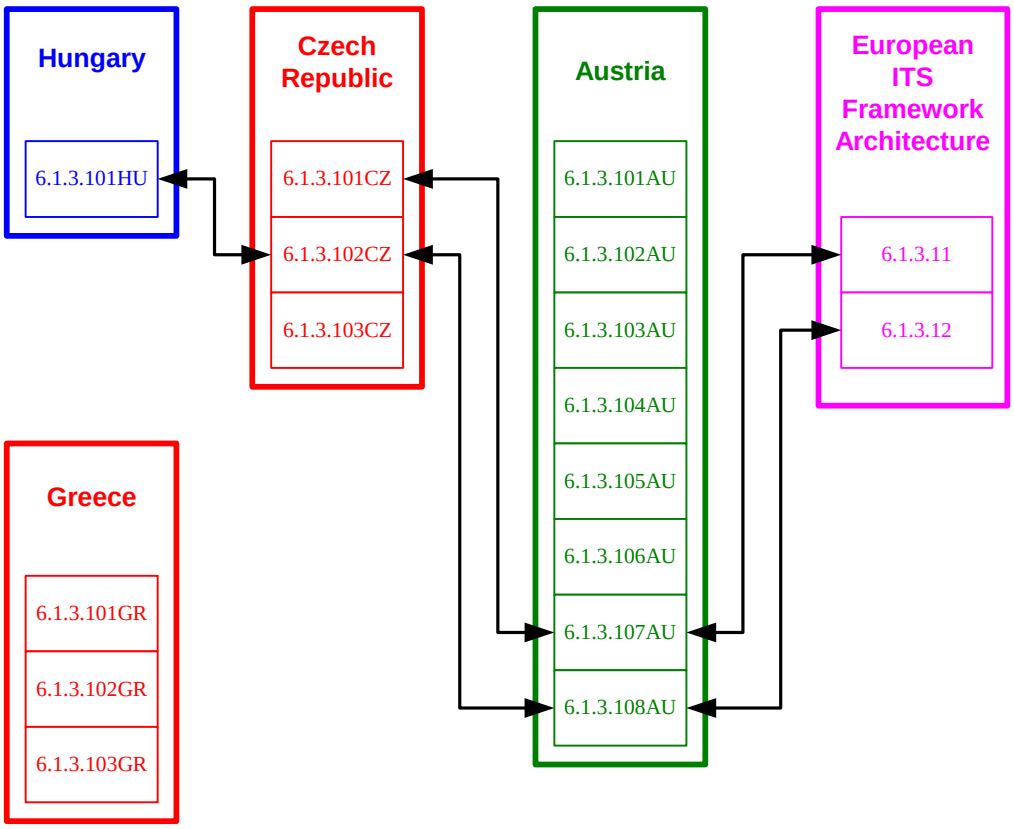
As a result of the establishment of this equivalence, the "Maintainers" of the European ITS Framework Architecture decide to add some new User Needs to the List of European ITS User Needs – see reference (3). The "Maintainers" combine the three sets of equivalent User Needs to form a single set of two new "European" User Needs. The User Needs for the fourth Nation are left alone and will continue to be used by that Nation.

The two new "European" User Needs created by the "Maintainers" take the next available numbers in the European List of User Needs for the Group, Service and Topic in which they will reside. The equivalence will be recorded in the documentation describing the Updates to the European ITS Framework Architecture that have produced the new Version. It will also be recorded in the new version of the List of European ITS User Needs.

[11] Details of how to submit Update Requests (currently to the FRAME Projects) can be found in the FRAME Web Site at: http://www.frame-online.net .

Figure 7 below illustrates how the equivalence has been combined to create two new "European" User Needs that will now appear in the List of European ITS User Needs – see reference (3).

Figure 7 User Configuration Management Practice Example 1 – Final Part



The three Nations involved in the equivalence have the option to either change the ITS Architectures that they have created so that they use the new European ITS User Needs (with the European numbering), or continue to use their own new User Needs (and numbering). Whichever they choose the original equivalence will still be recorded in the new version of the List of European ITS User Needs.

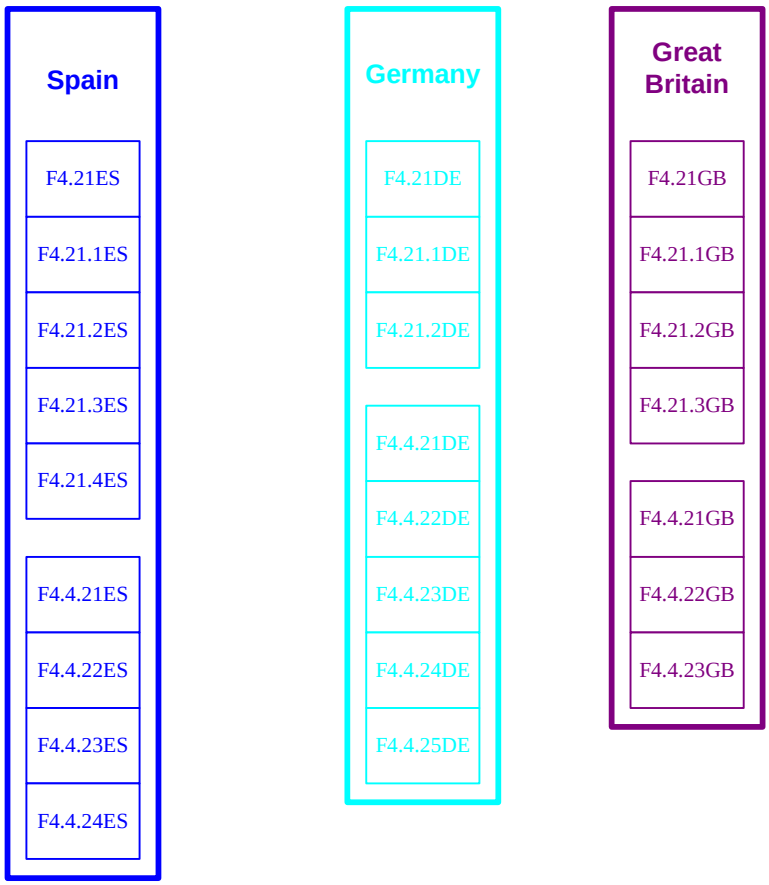
The creation of these new European User Needs will lead to a review of the existing "European" functionality. This will mean that the "Maintainers" will have to create new Functions and Functional Data Flows to fulfil these User Needs. Example 2 in the following Section illustrates the creation of new functionality in the European ITS Framework Architecture.

A 2.3 *ITS Architecture Development Example 1 – new Functions*

Three Nations, Spain, Germany and Great Britain are developing their own National ITS Architectures. Each of them is based on the European ITS Framework Architecture. As part of the work in each Nation, new Services have been identified, with the result that some new User Needs have been produced. This has led to the creation of new functionality in the Functional Viewpoints for their ITS Architectures.

The three Nations follow the Configuration Management practices for European ITS Framework Architecture Users. These are described in sections 4.2 and s4.3 of this Document. As a result of this work, the situation shown in Figure 8 below is created.

Figure 8 User Configuration Management Practice Example 2 - Part 1



As will be seen from Figure 8, there are three sets of new Functions, each with new numbers. For each Nation the numbers start from 21 – see Table in section 4.3.2, and include the Nation suffixes, applied in accordance with section 4.3.4.

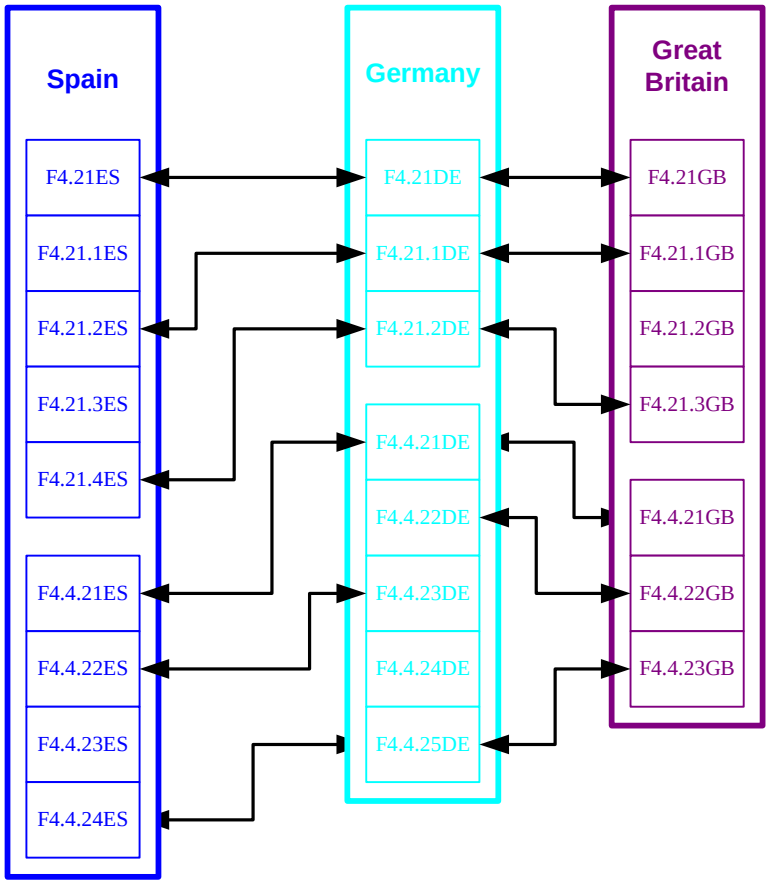
When each of the Nations is happy that its new User Needs and Functions are correct, they submit the details to the "Maintainers" of the European ITS Framework Architecture, as Update Requests[12]. The "Maintainers" examine what each Nation has done and resolve the issue of the new User Needs – see illustration provided by Example 1 in this Appendix.

As a result of the work on the User Needs, the "Maintainers" decide that there is some equivalence between the two sets of new Functions that each Nation has created. Figure 9 on the next page illustrates this equivalence through the lines that link some of the new Functions.

[12] See note 11.

Note that the equivalence of the User Needs must be established first. Once this is done, then any corresponding equivalence between the Functions can be established.

Figure 9 User Configuration Management Practice Example 2 - Part 1

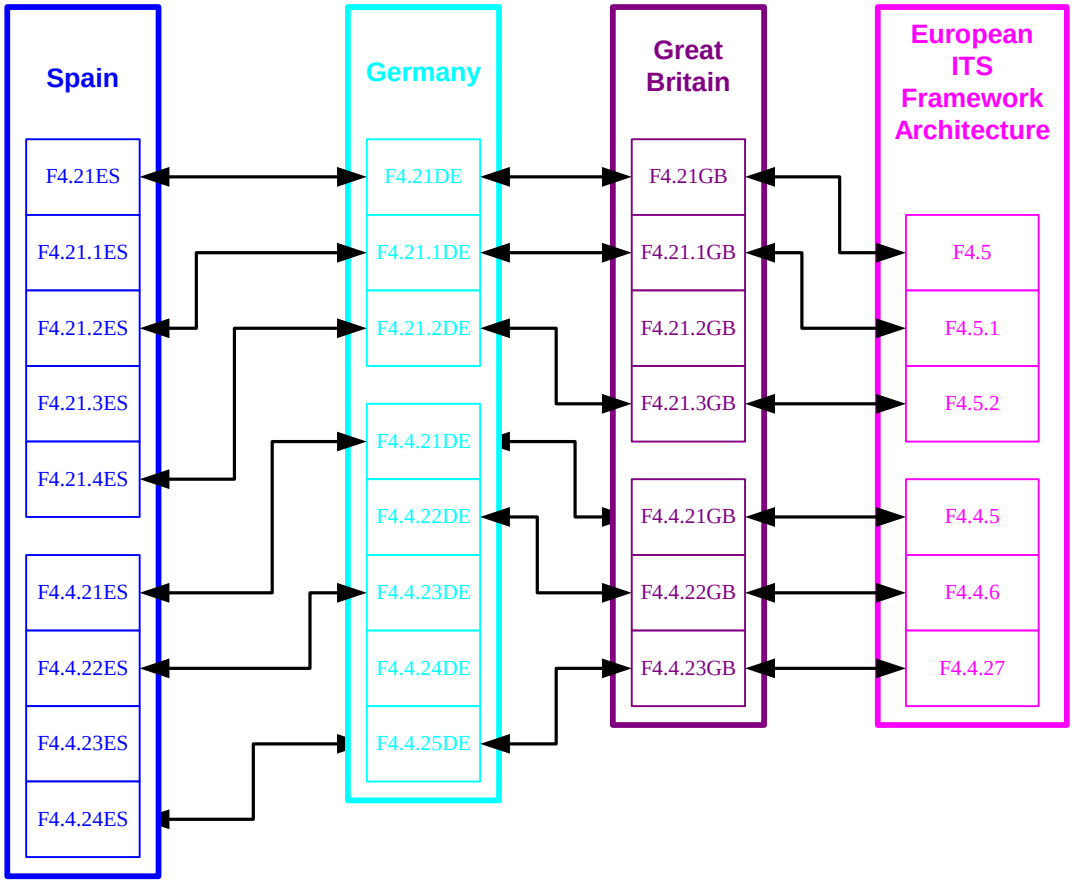


As a result of the establishment of this equivalence in the Functions, the "Maintainers" of the European ITS Framework Architecture decide to add some new Functions to the European ITS Framework Architecture, Functional Viewpoint – see references (6) and (7). The "Maintainers" combine the two sets of equivalent Functions created by each Nation to form two sets of new "European" Functions.

The new "European" Functions created by the "Maintainers" take the next available numbers in the sequence of European Functions for the Functional Area and High-level Function in which they will reside. The equivalence will be recorded in the documentation describing the Updates to the European ITS Framework Architecture that have produced the new Version. The changes will be shown in the new version of the European ITS Framework Architecture Functional Viewpoint – see references (6) and (7), produced as a result of the Update. The Trace Tables – see reference (5), will also be changed to show the link between the new "European" Functions and the new "European" User Needs and issued as part of the Update.

Figure 10 on the next page illustrates how the equivalence has been combined to create to new "European" Functions.

Figure 10 User Configuration Management Practice Example 2 – Final Part



The three Nations involved in the equivalence have the option to either change the ITS Architectures that they have created so that they use the new “European” User Needs and Functions (with the “European” numbering). Alternatively they can continue to use their own new User Needs and Functions (and numbering). Note that it will **not** be possible to adopt the new “European” User Needs without adopting the new “European” Functions, and vice versa.

Appendix 3 *Example illustrating the addition of a Function*

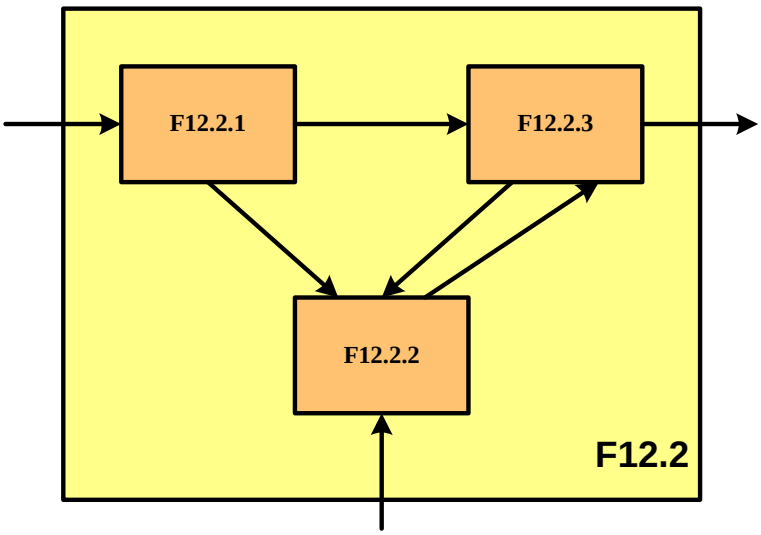
A 3.1 *Introduction*

This third Appendix provides illustrations of the addition of a Function according to the Configuration Management practices described in section 3.4.5. The deletion of a Function using these practices is described in Appendix 4.

A 3.2 *Starting Point*

For this example, consider a High-level Function (F12.2) in Functional Area 12. Note that this Functional Area has been chosen because it does not exist in the European ITS Framework Architecture and thus there should be no confusion with anything that currently exists. The High-level Function has three Low-level Functions as shown in the Figure below.

Figure 11 High-level Function F12.2 without additions

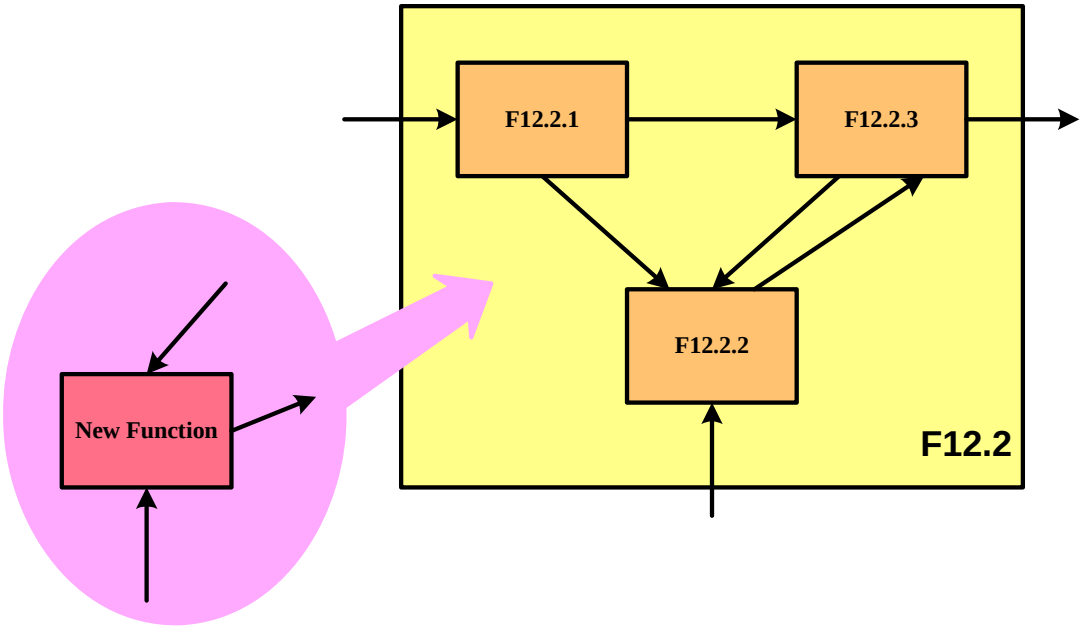


A 3.3 *The changes*

As a result of changes to the services that are included in the scope of the ITS Architecture, some new User Needs have been created. This is not uncommon, as services will be continually evolving over time, due to changes in end user expectations and travel requirements.

In order to fulfil the new User Needs, some new Functions have been created. During the process of creating these Functions it was decided that one of them is to be a new Low-level Function within the High-level Function F12.2. The reason for this is that part of the functionality required by the new User Needs has direct links with its existing constituent Low-level Functions in F12.2. This is illustrated by the following Figure on the next page.

Figure 12 Extra Low-level Function to be added to High-level Function F12.2



In order to add the links between the new Low-level Function and the existing Low-level Functions, two new Functional Data Flows are needed. The addition of these Data Flows will require new functionality to be added to that which already exists in two of the Low-level Functions (F12.2.1 and F12.2.2).

A 3.4 *Implementing Configuration Management practices*

Following the Configuration Management practices described in section 3.4.5 of this Document, the High-level Function (F12.2) must be re-numbered and re-named. The reasons for this are as follows.

(a) The addition of a new Low-level Function will change the overall functionality within the parent High-level Function (F12.2).

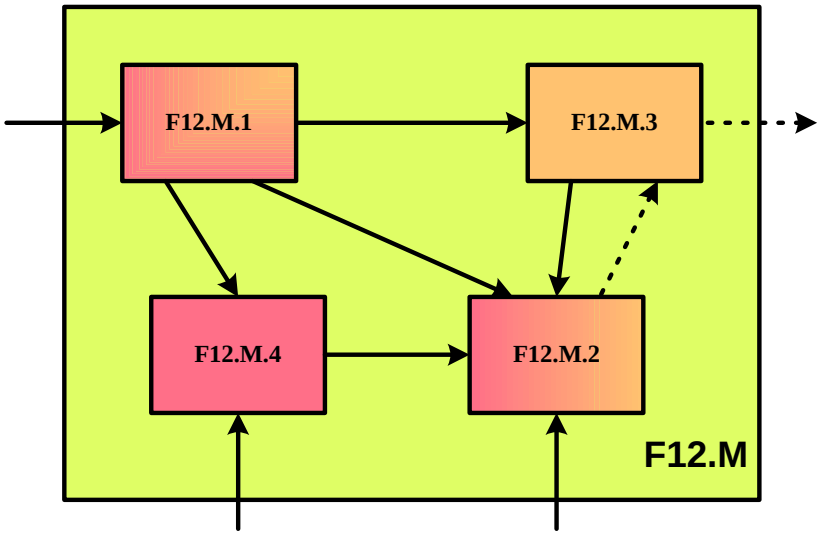
(b) The need for new functionality within the existing Low-level Functions (F12.2.1 and F12.2.2) to work with the Data Flows to/from the new Low-level Function means that they must also be changed.

Therefore a new High-level Function is created (F12.M), taking the next available number (M) in the High-level numbering sequence for Functional Area 12. The three original Low-level Functions from F12.2 are included in it and the new Low-level Function added. The existing un-modified Low-level Function (F12.2.3), the two modified Low-level Functions plus the new Low-level Function are given the numbers F12.M.1 to 4. The following points should be noted.

1. There is no need to re-number Low-level Functions F12.2.1 to 3 as F12.M.1 to 3 – though it probably makes sense to do so.
2. Although Low-level Function F12.2.3 is not changed, it must be moved and re-numbered as part of the new High-level Function F12.M.

The new High-level Function (F12.M) shown in the following Figure illustrates the results of implementing these changes.

Figure 13 The new High-level Function F12.M



A 3.5 *Impact on Data Flows*

Note that two of the Functional Data Flows shown in the Figure have been changed into “dotted” lines, as they may have to be changed into new versions of the original two Data Flows. This is because some of the additional functionality included in the new and modified Low-level Functions may result in changes to the data being transferred between these Functions.

A consequence of the possible change to the Functional Data Flow that goes outside of High-level Function F12.M (see the Data Flow output from F12.2.3 in the top right-hand corner of the Figure) is that changes may be needed to Functions elsewhere in the ITS Architecture, e.g. Functional Area 12 and/or other Functional Areas.

Appendix 4 *Example illustrating the deletion of a Function*

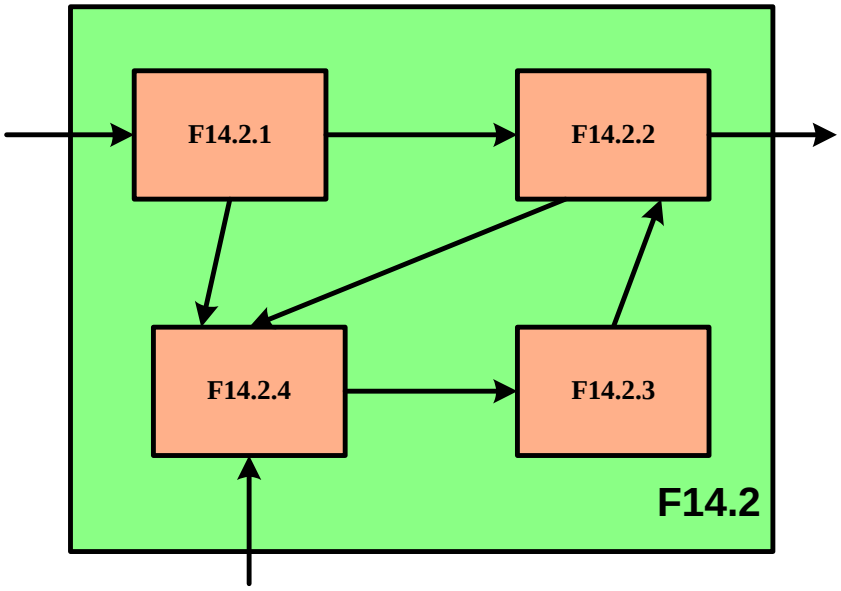
A 4.1 *Introduction*

This fourth Appendix provides illustrations of the deletion of a Function according to the Configuration Management practices described in section 3.4.5. The addition of a Function using these practices is described in Appendix 3.

A 4.2 *Starting Point*

For this example, consider a High-level Function (F14.2) in Functional Area 14. Note that again this Functional Area has been chosen because it does not exist in the European ITS Framework Architecture and thus there should be no confusion with anything that currently exists. The High-level Function (F14.2) has four Low-level Functions as shown in the Figure below.

Figure 14 High-level Function F14.2 without deletions



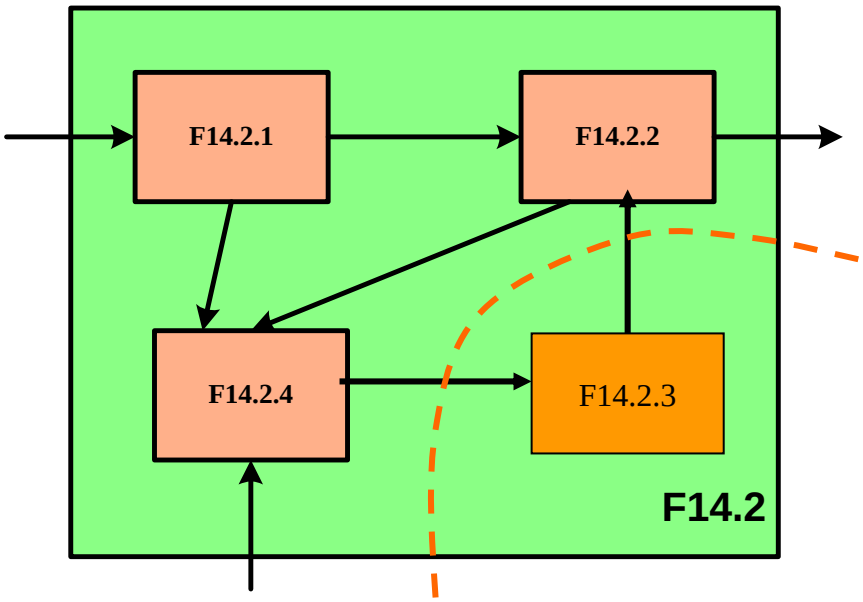
A 4.3 *The changes*

As a result of reductions in the services that are included in the scope of the ITS Architecture, some User Needs no longer need to be served. Possible reasons for the reduction in services are described in point 1 of section 3.4.5.

A result of the removal of these services is that some of the User Needs are no longer required – see point 1 of section 3.4.2 for details of how the Configuration Management practices will affect these User Needs. A consequence of these User Needs no longer being required is that one of the Low-level Functions (F14.2.3) is no longer needed and can be removed. The reason for this Function being no longer needed is that all of the User Needs that it helps to satisfy are not needed.

The removal of this Function (F14.2.3) from its parent High-level Function (F14.2) is illustrated by the following Figure.

Figure 15 High-level Function F14.2 with Low-level Function removed



Note that if only some of its User Needs were no longer needed, then the Function F14.2.3 would have required modifications and would not need to be deleted. This is because the only Functions that are removed are those that have all the User Needs that they serve removed.

As a result of the deletion of the Low-level Function (F14.2.3), two of the existing Data Flows have been deleted. These are the Data Flows linking the Function to the other Low-level Functions (F14.2.2 and F14.2.4). This means that these two Low-level Functions will have to be modified to remove the functionality that creates and receives these Data Flows.

A 4.4 *Implementing Configuration Management practices*

Implementing the Configuration Management practices described in section 3.4.5 means that the High-level Function (F14.2) must be re-numbered and re-named. The reasons for this are as follows.

(a) The deletion of the Low-level Function (F14.2.3) will change the overall functionality within the parent High-level Function (F14.2).

(b) The need for modifications to the functionality within two of the existing Low-level Functions (F14.2.2 and 4) due to the removal of the Data Flows to/from the deleted Low-level Function means that they must also be changed.

Therefore a new High-level Function is created (F14.N), taking the next available number (N) in the High-level numbering sequence for Functional Area 14. The three remaining Low-level Functions from F14.2 (F14.2.1, F14.2.2 and F14.2.4) are included in it. Two of these Low-level Functions (F14.2.2 and F14.2.4) are modified

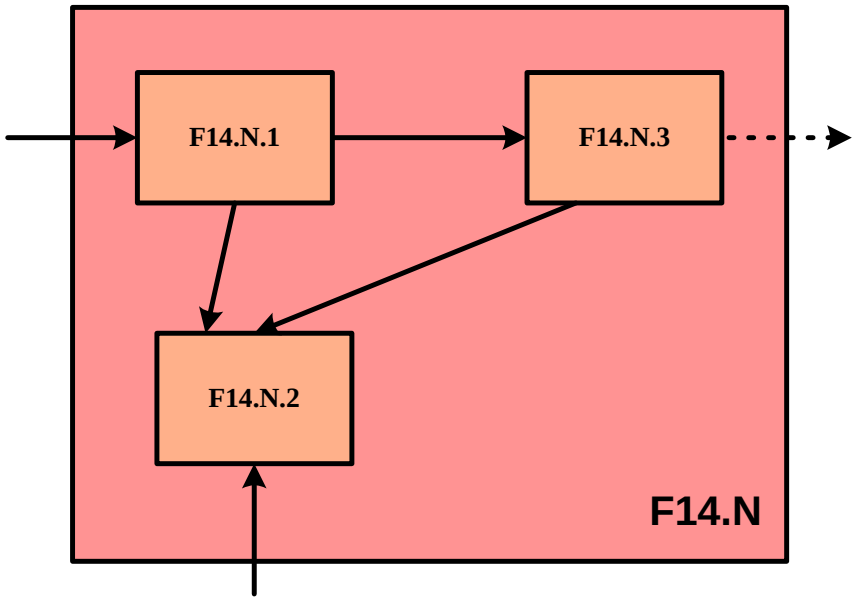
because of the removal of the Data Flows linking them to the deleted Function (F14.2.3). The following points should be noted.

1. There is no need to re-number Low-level Functions F14.2.1, 2 and 4 in the same order, though it probably makes sense to keep them as similar as possible.

2. Although Low-level Function F14.2.1 is not changed, it must be moved and re-numbered as part of the new High-level Function F14.N.

The new High-level Function (F14.N) shown in the following Figure illustrates the results of implementing these changes.

Figure 16 The new High-level Function F14.N



A 4.5 *Impact on Data Flows*

Note that one of the Functional Data Flows shown in the Figure has been changed into a “dotted” line, as it may have to be modified into a new version of the original Data Flow. This is because the removal of the Low-level Function that has led to the creation of the new High-level Function will have changed the data going into Function F14.N.3. A consequence of changing this Data Flow is that changes will be needed to Functions elsewhere in the ITS Architecture, e.g. Functional Area 14 and/or other Functional Areas.