CONFIGURATION MANAGEMENT IN THE EUROPEAN ITS FRAMEWORK ARCHITECTURE ENVIRONMENT

R.A.P. Bossom, B.Sc (Hons), C.Eng. MBCS, AMIMechE, System Design Services, Siemens Traffic Controls, Sopers Lane, Poole, Dorset. BH17 7ER. UK
Tel: +44 (0)1202 782216 E-mail: richard.bossom@siemens.com
Eur Ing P H Jesty, B.Sc (Eng), M.Sc, C.Eng, MIEE, MBCS, MIEEE, Peter Jesty Consulting Ltd, Warwick Lodge, Towton, Tadcaster, North Yorkshire, LS24 9PB.
Tel: +44 (0)1937 833640 Email: phj@peterjesty.com

SUMMARY

This paper provides the background and description of the Configuration Management practices being implemented by the FRAME Project for the European ITS Framework Architecture. The challenges that this presents arise from the flexibility of use that is built into the Framework Architecture and the freedom that its Users have to modify it when creating their own ITS Architectures. Therefore Configuration Management practices have been developed for both the Framework Architecture maintainers and its Users.

INTRODUCTION

The normal objective of Configuration Management is to maintain consistency of a system or product as changes are being made to its various components. These changes will normally be introduced as part of corrective, adaptive or preventative maintenance that is applied to the system or product. Configuration Management can become more complex if it is also necessary to maintain compatibility with earlier versions of the system or product.

A challenge faced by the FRAME Project (part of the European Commission's 5th Framework Programme) is the application of these objectives to the European ITS Framework Architecture. This is not a trivial task because the Framework Architecture is designed to give its Users a large degree of freedom in the way that they adapt it to suit their own particular ITS deployments. Such a large degree of freedom for User adaptations does not appear to be common amongst other ITS Architectures in use around the World. Generally speaking these ITS Architectures are more proscriptive and hence provide less freedom for the extent to which Users are able to make changes.

This paper provides an outline description of the Configuration Management practices developed by the FRAME-S Project for its maintenance of the European ITS Framework Architecture and for Users developing ITS Architectures. These practices are probably unique for ITS Architectures around the World. To make them easier to understand the paper illustrates the practices with two examples. As an introduction and an aid to understanding, this paper starts by describing ITS Architecture characteristics of the European ITS Framework Architecture and the basic principles of Configuration Management.

ARCHITECTURE CHARACTEISTICS

Introduction

In November 2000 the KAREN Project produced the first version of the European ITS Framework Architecture (the Framework Architecture), as part of the European Commission's 4th Framework Programme. The main objective of the Framework Architecture is to provide a starting point from which Users can develop their own ITS Architectures for application at National, Regional, City or Local level. There are several differences between the characteristics of the Framework Architecture and those of many other ITS Architectures.

Flexibility

The first difference is that unlike many other ITS Architectures, the Framework Architecture provides its Users with a large degree of flexibility about the ways in which they can adapt it. This is because the Framework Architecture does not contain many of the parts that are found in other ITS Architectures. Figure 1 below illustrates this by showing the development stages for both the Framework Architecture and most other ITS Architectures.

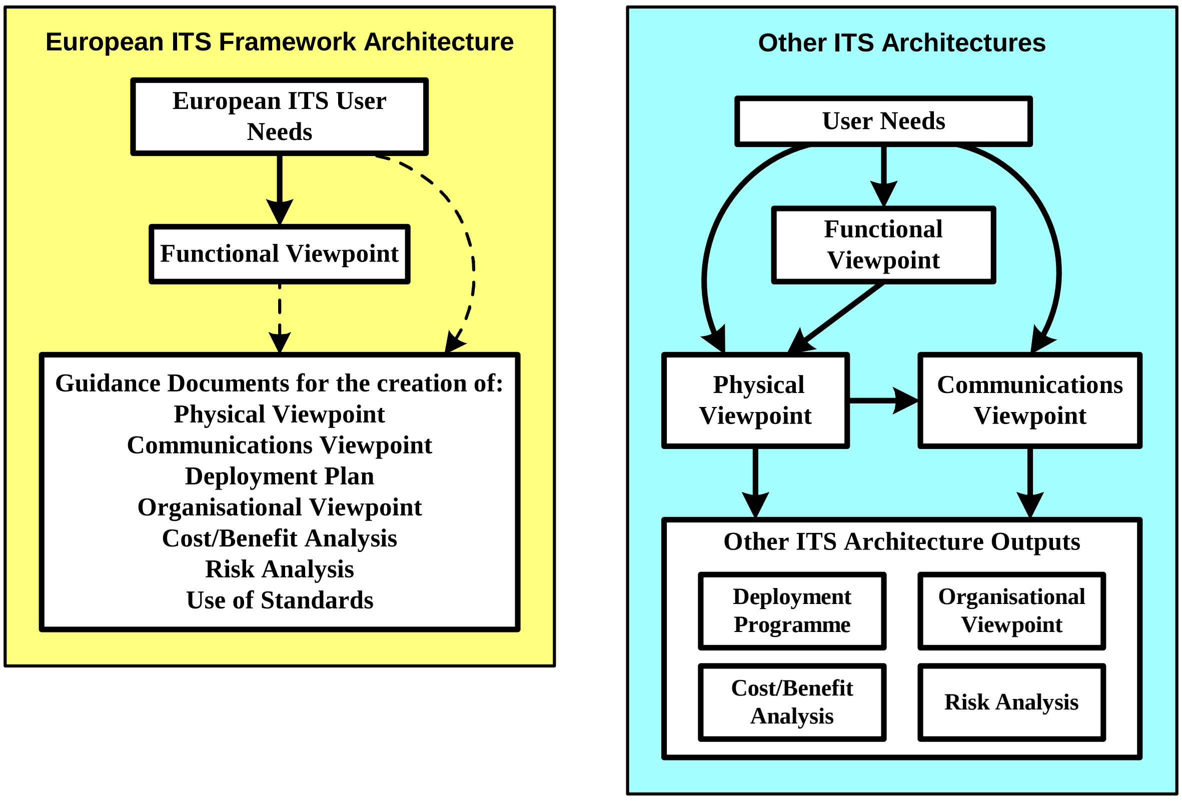


Figure 1 Comparison between Architectures

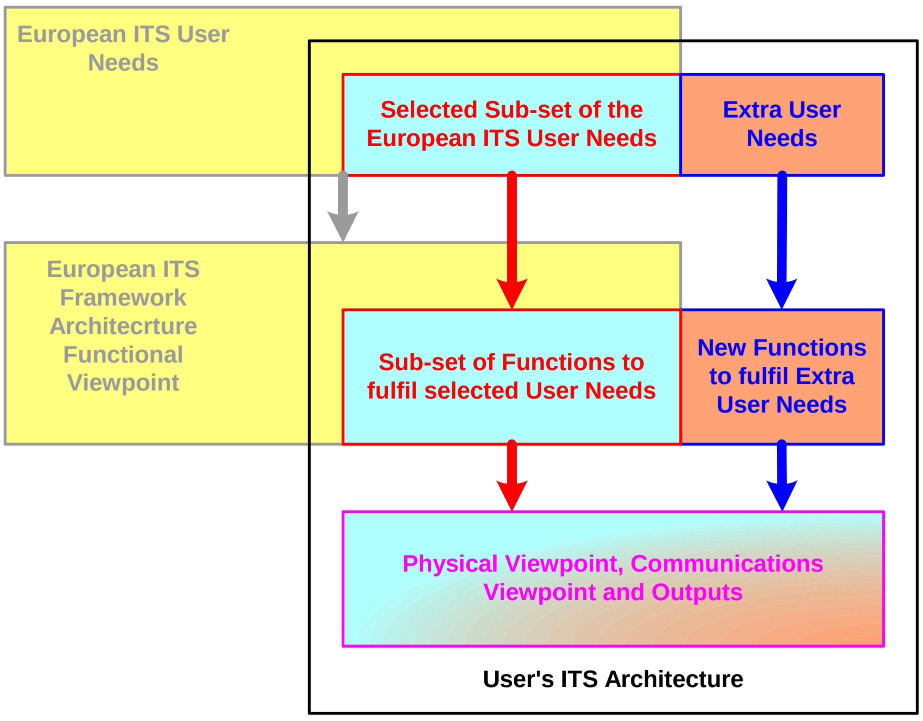
In the above Figure it will be seen that the Framework Architecture consists of the European ITS User Needs that define the services that are to be delivered and a Functional Viewpoint that defines the functionality needed to fulfil these User Needs and hence deliver the services. The equivalents of these two parts are also found in other ITS Architectures. The differences lie in the remaining parts in each of the Architectures.

Most other ITS Architectures provide a Physical Viewpoint that defines a range of components into which the functionality will be incorporated and a Communications Viewpoint identifying the mechanisms that will be used by the components to exchange information or data. They may also provide other outputs such as Deployment Plans, an Organisational Viewpoint, plus the results of Cost/Benefit Analysis and Risk Analysis.

The range of components included in the Physical Viewpoint can provided Users of other ITS Architectures with restrictions in the way that they adapt them to enable their own particular services to be delivered. In the Framework Architecture these two Viewpoints are not defined and instead Users are provided with guidance on how to create them and the other outputs from Architecture creation. This means that Users have a great deal of freedom in the ways that they adapt the Framework Architecture so that it will deliver the services required for their particular ITS deployments.

Structure

The second difference is that in the Framework Architecture the User Needs and the Functional Viewpoint are structured to encourage and make their adaptation easy. A User can select the parts of the European ITS User Needs that include the services that they want to provide. These selected User Needs are then use to determine the required functionality from the European ITS Framework Architecture. If necessary Users can add extra User Needs for additional specific services without affecting the User Needs that they have selected. They can then develop functionality to serve these User Needs and combine it with that from the Framework Architecture. This combined functionality is then used to produce the Physical Viewpoint for the User's ITS Architecture, which in turn can lead to the creation of the Communications Viewpoint and the other outputs. Figure 2 below illustrates the way that the Framework Architecture can be selected and modified.



European ITS User Needs
Selected Sub-set of the European ITS User Needs
Extra User Needs
European ITS Framework Architecrture Functional Viewpoint
Sub-set of Functions to fulfil selected User Needs
New Functions to fulfil Extra User Needs
Physical Viewpoint, Communications Viewpoint and Outputs
User's ITS Architecture

Figure 2 Adding to the Framework Architecture

Other Differences

There are two other differences between the characteristics of the Framework Architecture and other ITS Architectures. Although they do not have a great impact on Configuration Management they have influenced the creation of the Framework Architecture and are therefore worth noting here.

The first of these two differences is that some developers need to translate the text of the ITS Architectures that they create from the Framework Architecture into their own, non-English, language to make it more accessible for their own Users. However acronyms and other English language jargon do not always translate well into other languages.

The second of these two differences is that the ITS services that are included in the European ITS User Needs may be acceptable and appropriate in one country, but may not be applicable in other countries. Even if they are, the implementation method that is included in the functionality may be different. This means that the functionality within the Framework Architecture has been defined either at a high level, or structured to enable different national implementations to be accommodated by only selecting the appropriate parts.

COMPATIBILITY AND CONFIGURATION MANAGEMENT

Introduction

The following sections of this paper provide definitions of what “compatibility” means and why it is needed. Without these definitions, the term has no relevance to the 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.

What does compatibility mean?

The content and scope of any set of 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.

Why is compatibility needed?

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

> <u>Reason 1</u>: “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.
>
> <u>Reason 2</u>: funding. Obtaining funding for the development of National, Regional or Local ITS Architectures can be easier if it is known that they will be compatible with the Framework Architecture. This is because organisations will be more

"comfortable" investing in the ITS 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 is because interfaces between the components are well defined and can be made to confirm to recognised standards, such as those produced by CEN, thus making it easier to replace a component with a newer version. All components should work with each other in a way that means that it is easy to replace one without disrupting the operation of the others. Components in one system derived from the ITS Architecture should be easily linked to those in another system derived from the same ITS Architecture.

Without "compatibility", ITS components will be created as separate entities. One of the results of this will be that many of the benefits of compatibility, such as inter-operability, will be lost, or difficult to achieve. This means that the achievement of the benefits of ITS including "seamless" borders across the nations of Europe will be retarded, leading to disruption and extra costs for travelling.

What will be Compatible

There are two topics to be considered if ITS Architectures are to be compatible with the Framework Architecture. They concern both its development and its content and have been grouped into the following two topics.

Topic 1: Methodology. A study of ITS Architecture developments around the world shows that in some cases different methodologies are being used. These methodologies 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 ways in which parts of the Framework Architecture are defined and described. They include 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-S Project through the material presented during part of the FRAME Training Workshops that are currently being held within Europe. These Workshops take attendees through the process and methodology for creating ITS Architectures that are based on the Framework Architecture.

The second Topic needs to be addressed if "compatibility" is to have any real value and meaning. This can be achieved with the initial development of ITS Architectures being made using the Framework Architecture. However the Framework Architecture is intended to be a "living thing" and it will change with time to reflect the evolution of ITS. Continuing "compatibility" can only be achieved through the use of the Configuration Management practices that are described in the rest of this paper.

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 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" is a high priority goal of any changes that are made, when necessary.

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

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

ITS Architectures specify components, most of which are similar to those in IT. Thus it is logical to apply the same Configuration Management practices to their creation and development and also to the maintenance and upgrade of the Framework Architecture. How this is being implemented by the FRAME-S Project is described in the next section of this paper.

EUROPEAN ITS FRAMEWORK ARCHITECTURE CONFIGURATION MANAGEMENT BY THE FRAME PROJECT

One of the activities of the FRAME-S Project is to maintain the Framework Architecture. It will change the Framework Architecture for any of the following three reasons.

1. To include new services, i.e. expand the scope of the User Needs.
2. To improve its usability, i.e. change some of the definitions and descriptions to make them easier to understand.
3. To correct any inconsistencies.

The FRAME-S Project will apply Configuration Management to the maintenance of the Framework Architecture. It will be applied so that "backwards compatibility" is maintained from one Version to the next. 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.

Any changes, and hence Configuration Management practices, will be applied to the several parts of the Framework Architecture. These are listed below together with a brief outline of the consequences in each case.

<ins>User Needs</ins>: New User Need will be added within the existing structure of the European ITS User Needs, taking a new number for a Group, or Service, etc. New numbers for Groups, Services and Categories should start at 21. The numbers for new User Needs within Groups, Services and Categories should start at 101. Those

User Needs no longer required will have their links to Functions deleted but will not be removed. Their contents will be modified to add an indication that they are no longer used.

Terminators and their Actors: New representations of the outside World will be provided through the addition of new Actors to existing Terminators rather than creating new Terminators. Existing Terminators and Actors that are no longer required will not be deleted and their acronyms will be retained and not re-used.

Terminator Data Flows: All new Terminator Data Flows will take names that have not been used before, conforming to the hierarchies and naming conventions for the appropriate type of Terminator Data Flow. Existing Terminator Data Flows will only be deleted if their source and/or destination Terminators, Actors, Functions or Data Stores have also been deleted. The actual "deletion" process will only involve their removal from the lists of input and output Data Flows for Functions and from the list of components of other Terminator Data Flows, plus the addition of a keyword indicating their new status to prevent their names being re-used.

Functions: A new Function will be created if an existing Function needs to be modified. Modifications will be made either because of changes to the allocation of User Needs to a Function, or the Data Flows to/from a Function are changed, or the Function is too complex and is being split into two or more parts. New Functions must take the next new number for their new place in the functional hierarchy and be given names that have not been used before. Any existing Functions no longer required will have each part of their specification changed to show their "deleted" status so that their numbers and names will not be re-used.

Functional Data Flows: All new Functional Data Flows must take new names that have not been used before, conforming to the appropriate naming conventions. They must have a single source and destination Functions and/or Data Stores. Existing Functional Data Flows can only be removed because their source and/or destination Functions or Data Stores have also been removed. All references to them in Functions specifications will be deleted and they will be removed from Data Flow Diagrams. They will be provided with a keyword indicating their new status to prevent their names being re-used.

Data Stores: New Data Stores must take the next new number in the sequence for the Functional Area in which they will reside and be given names that have not been used before. The extra functionality needed for the new Data Stores must be added in accordance with the rules for new Functions – see above. Existing Data Stores can only be removed if the Functions providing data to or receiving data from them have been removed. The specification of these Data Stores will be changed to show their "deleted" status so that their numbers and names of removed Data Stores cannot be re-used.

All changes to the Framework Architecture will be documented so that it will be possible to see what has been added and removed for each new Version. This is important as it helps Users understand how their existing ITS Architectures will be affected.

EUROPEAN ITS FRAMEWORK ARCHITECTURE CONFIGURATION MANAGEMENT BY USERS

As Framework Architecture Users may work in parallel and (possibly) without reference to each other, their own changes must be made very carefully. Just omitting User Needs or parts of the Functional Viewpoint does not present problems, and the FRAME Selection Tool is able to help with this phase of ITS Architecture development.

To enable national ITS Architectures to develop independently, all new User Needs and Functional Viewpoint component identifications must have guaranteed unique identifications. This will be achieved by including as a suffix to the User Need and Function numbers or the Data Flow names, the international standard abbreviation of the country for which the Architecture is being developed. After each country suffix extra letters can be added for city, regional, or specific Architectures if required.

To avoid clashes with future updates to the Framework Architecture, its Users have been provided with a set of “starting numbers” for User Needs, Functional Areas, Functions and Data Stores. This means that the same numbers will not be used with or without suffixes, depending on whether the part is in the Framework Architecture or a User’s ITS Architecture.

When Users’ changes are included in a Framework Architecture update, the FRAME Project removes the suffixes, sometimes re-numbering or re-naming as needed if a particular number or name is already in use. In this case the Users will have to make these changes to their own ITS Architecture to retain complete compatibility with future Versions of the Framework Architecture.

EXAMPLES OF CHANGING THE FRAMEWORK ARCHITECTURE

Introduction

The Configuration Management practices for both Framework Architecture maintainers and Users described in this paper are not trivial or easy to understand. In order to provide an insight into what they involve, they will now be illustrated by two examples. The first of these concerns the addition of new User Needs and the second the addition of a new Function.

Addition of new User Needs

As will be seen from Figure 3 below, some new User Needs have been produced for ITS Architectures in Hungary, Austria, The Czech Republic and Greece. Some of the User Needs for the first three countries concern cross-border co-operation, whilst those for Greece are for services that are particular to that country. The ITS Architecture developers in each country have numbered their User Needs according to the Configuration Management practices outlined above and have also included the appropriate country specific suffixes.

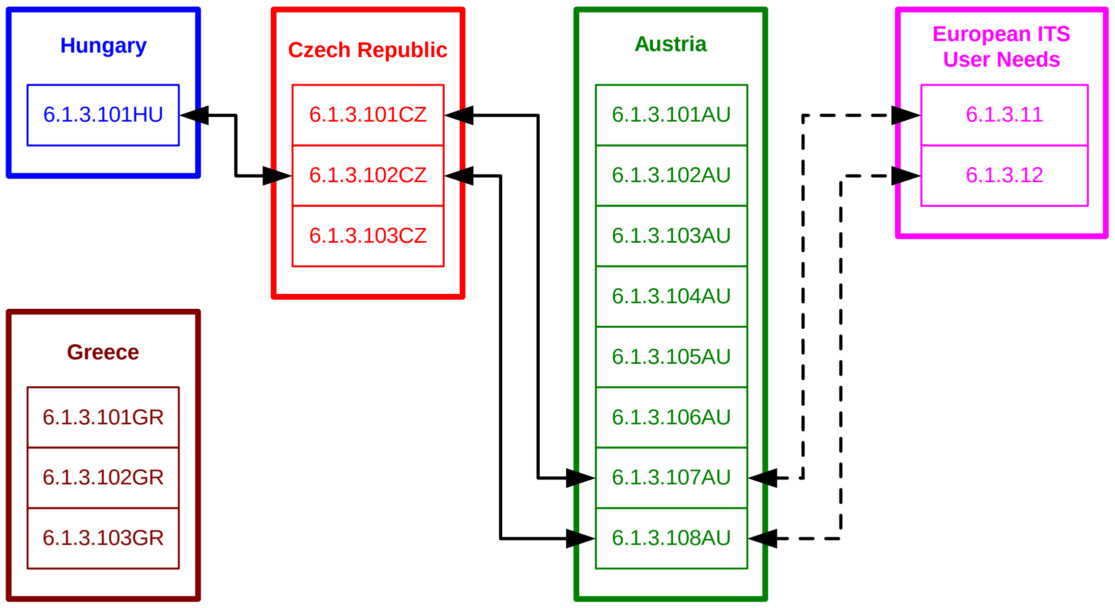


Figure 3 Example of adding new User Needs

The Framework Architecture maintainers in the FRAME-S Project analyse the new User Needs from the four countries. After consultation they decide that some of those from Hungary, Austria and The Czech Republic are compatible (shown by the "full" lines in Figure 3) and should be included in the European ITS User Needs. Those from Greece are too country specific to be included. The result is that two new User Needs are added to the European ITS User Needs by the Framework Architecture maintainers, as shown in the "box" on the right-hand side of Figure 3. The "dotted" lines show the links between the two new "European" User Needs and those created for the three countries.

The owners of the ITS Architectures whose User Needs are compatible with the new European ITS User Needs can initially leave their User Needs unchanged. However this practice is not recommended, as it requires that the relationships shown in Figure 3 are documented as part of each ITS Architecture. Also the relationship may be difficult to understand as the Framework Architecture and/or each ITS Architecture continue to evolve.

Addition of a new Function

The changes to the scope of an ITS Architecture have led in the addition of some new User Needs. In order to fulfil these new User Needs new Functions have been created. During the process of creating these Functions it has been decided that one of them is to be a new Low-level Function within an existing High-level Function.

The "box" in the top left-hand side of Figure 4 on the next page shows this existing High-level Function (F12.2). As can be seen, it contains three Low-level Functions, F12.2.1 – 3, which exchange data with each other and with other parts of the functionality in the ITS Architecture.

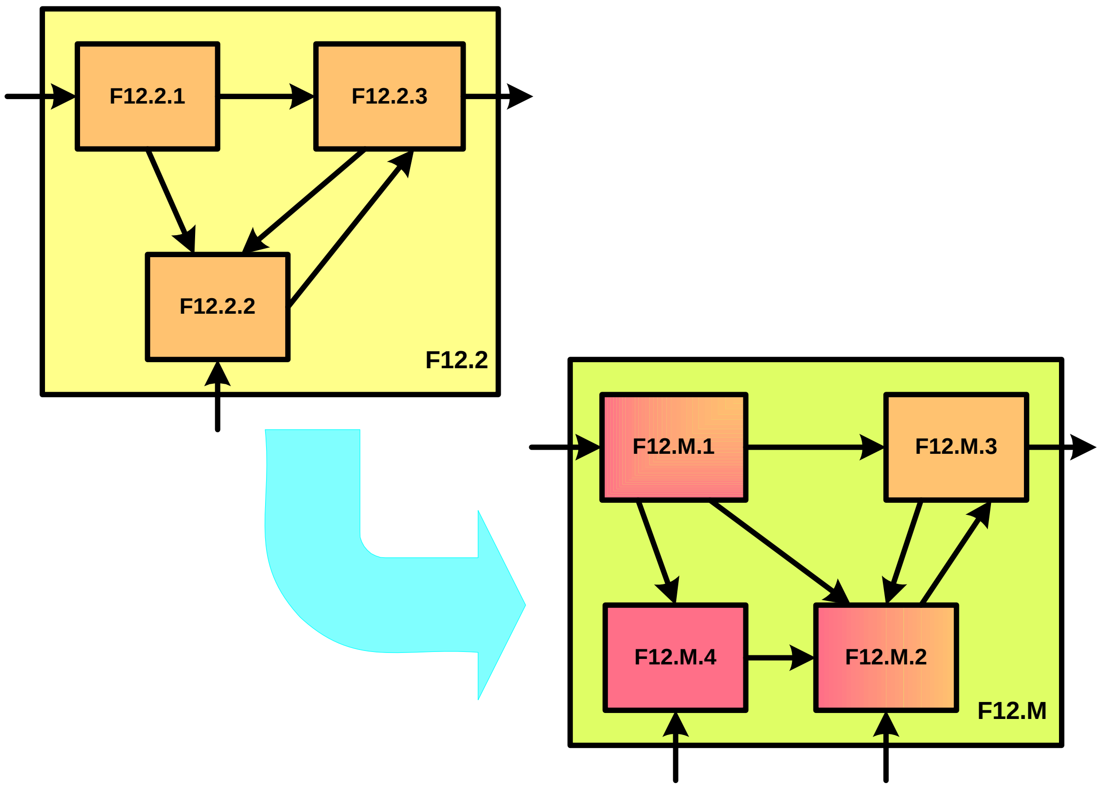


Figure 4 Example of adding a new Function

Following the Configuration Management practices defined above, adding a new Low-level Function requires that the High-level Function (F12.2) must be re-numbered and re-named as its functionality will change. The change is because the new Low-level Function has been added and so it is no longer the same as before. The existing Functions will also have to be changed in order to communicate with the new Function and will also have to be re-numbered as they are part of a new High-level Function.

Therefore a new High-level Function is created (F12.M), taking the next available number (M) in the High-level numbering sequence within the hierarchy for Functional Area 12. The three original Low-level Functions from F12.2 are included in it but with changes that enable them to communicate with the new Low-level Function through new Data Flows where necessary. The new Low-level Function is added and all four Functions are given the numbers F12.M.1 to 4. The “box” on the right-hand side of Figure 4 on the previous page shows how they appear in the new High-level Function (F12.M).

ITS Architecture owners who do not want to include the new User Needs and hence the new functionality need not make any changes immediately. This is because the existing functionality in the Framework Architecture has been replaced but not deleted. However in the long term, the changes may need to be made, as further Versions of the Framework Architecture will be based on the new functionality.

Changes that Users make for their ITS Architectures

The changes that Users make to the Framework Architecture as part of their ITS Architecture development should follow the practices illustrated by these two examples. The important difference will be in the choice of new numbers and names. As noted previously, “starting” numbers have been defined for Users to apply to new User Needs, Functional Areas, Functions and Data Stores. Also country (plus region, city, etc.) suffixes need to be added to provide uniqueness in parallel ITS Architecture developments.

FURTHER COMMENTS ON IMPLEMENTING CONFIGURATION MANAGEMENT PRACTICES

Aside from those needed to remove inconsistencies, changes to the Framework Architecture will be as a result of changes to the European ITS User Needs. These will be driven by changes to the services (and hence the scope) supported by the European ITS Framework Architecture. In the two examples, additions have been shown, but deletions are also possible.

For changes involving deletions the same Configuration Management practices used for additions must be applied. Thus for example, removing User Needs and hence changing functionality will require new High and Low-level Functions to be created, following the practice shown in the second example. The new Functions will contain less functionality then those that they replace but will still require the creation of new Data Flows to support them.

CONCLUSIONS

Configuration Management is being applied to the European ITS Framework Architecture without restricting its adaptation to suit individual ITS deployments. This is achieved by having rules for the FRAME Project to apply to the Framework Architecture updates, and for Users to apply to their adaptations of it. Thus Users are able to create Architectures for individual ITS deployments within a managed structure that does not stifle innovation.

The Configuration Management practices outlined in this paper are non-trivial. So far as is known, they are unique, since most of the other ITS Architectures being used around the World are more proscriptive than the Framework Architecture. This means that these other Architectures provide less freedom for Users to make changes to suit their own particular ITS deployments.

A much more detailed description of the Configuration Management practices outlined in this paper is provided in a document produced by the FRAME-S Project (D14). Copies of this document can be downloaded from the FRAME Web Site at: http://www.frame-online.net .

ACKNOWLEDGEMENTS

The work described within this paper was funded by the FRAME-S support action of the Information Society Technologies Programme of the fifth Framework Programme of the European Community; the ideas contained therein do not necessarily represent the views of the Commission.