YAMAICA Franca UML Mapping Guideline

YAMAICA Franca UML Mapping Guideline (Status DRAFT)

- This document is currently being reworked*

This document describes how to map interfaces and their deployment in an UML architecture model to Franca IDL. This document refers to Franca >= 0.9.

Mapping Rules

Mapping UML model structure and name space

Specific UML packages and their contents can be mapped to Franca IDL files. The *«fidl»* stereotype indicates that a package shall be mapped to a Franca IDL file. The UML package node name is used as Franca base file name, complemented by the suffix ".fidl".

Any contents of packages below will be added to the same Franca IDL file, using the resulting name space:

- «fidl»* stereotyped UML packages may be nested. If there is an UML package B marked with stereotype «fidl» below another UML package A marked with stereotype «fidl», then any contents of B will not be part of the Franca IDL file A, but contained within a new separate Franca IDL file B.

If any UML package is stereotyped *«nofidl»*, this package and its sub tree *MUST* be ignored by Franca import/export tools completely.

Franca interfaces *MUST* be specified as direct child elements of any *«fidl»* stereotyped UML package. Any plain UML packages below any *«fidl»* stereotyped UML package is mapped to a Franca type collection. Any deeper sub tree hierarchies are ignored unless another *«fidl»* stereotyped UML package occurs.

UML Namespace

In UML a namespace is defined by the node path, starting from a give root node. Franca IDL package name is derived using the relative UML node path from the namespace root node (namespace root) to the UML package node marked with stereotype «fidl». By default the UML root node is used for Franca package name generation. Any UML package may be defined as namespace root to indicate that all Franca IDL files below shall use package names relative to the marked UML package. The UML namespace root itself *MUST* be included within the Franca IDL package name. The name of *«fidl»* stereotyped UML packages *MUST* be included in the Franca package name.

The namespace root can be defined in two different ways:

EA UML namespace root marker (recommended)

Mark some UML package as "namespace root": in EA project browser right click the UML package \-> "Code Engineering" \-> "Set as Namespace Root".

stereotype «root»

Mark some UML package with stereotype «root».

tagged value overrides

In case the UML model structure is not aligned to the required Franca-IDL file or directory structure, tagged values can be used to override the Franca IDL file name and location.

1. FIDL-Name overrides the Franca-IDL file name (UML package name is not used anymore)
2. FIDL-Namespace-Prefix overrides the Franca-IDL package name prefix (UML model package path is not used anymore); file name will be added as end of the package prefix.

|| Example UML package tagged value overrides || Franca IDL file with name *de/bmw/Test.fidl* || | !uml-root-tagged-values.png|align=right,border=1! | {code:title=interface version specification "4.2"|borderStyle=solid} package de.bmw.Test {code} |

Example + Screenshot (TBD)

Franca requires that package names must not contain certain characters. Spaces will be replaced by "_"; other special characters or numbers at the beginning of package names are forbidden and will cause an error during transformation. The UML usage guidelines / checker shall recommend/enforce exclusion of such special characters and umlauts in UML packages and define as well whether capital letters shall be allowed. Besides the special character treatment the transformation UML <-> Franca just leaves the names as they are.

interpreting package names

The Franca package name is also used to derive a directory name from the Franca-IDL file name. The Franca-IDL file name is the last part of the package name, i.e. the part behind the last dot. The Franca-IDL file will be placed in a directory hierarchy, where each level´s directory name consists of the package name parts between the dots.

Datatypes

All data types are placed within UML packages by selecting a package in the EA project browser, right click \-> Add \-> Add Element and configure the type "Class" and one of the stereotypes *«typedef»*, *«struct»*, {color:red}*«struct polymorphic»*{color}, *«array»*, *«FrancaEnum»*, {color:red}{-}*«enumeration»*{-}{color}, *«union»* or *«map»* in the dialog box. Type definitions will be mapped to Franca type collections for each UML package. The name of a type collection is derived from the relative path between (excluding) the nearest UML package with stereotype «fidl» and the package where the type definitions are placed (including).

{color:red}to be removed after introduction of anonymous type collections with Franca 0.8.9 :{color} {color:red}{-}It is forbidden to place data types within an UML package marked with stereotype «fidl» directly and will raise an error during transformation.-{color} {color:red}{-}UML design guidelines and checker shall enforce this interdiction.-{color}

Typedef of Basic Datatype

Add the freshly created class to a diagram and right click \-> Advanced \-> Parent, enter the origin basic type´s name and press the "Add" button.

Defining Structures

Enter the new elements "Properties Details", click "Attributes..." and add any structure fields you wish with their corresponding types. It is not required to place stereotypes to the fields. The order of declared fields within the struct is retained during any model transformation or code generation process. Because of this reason it is not allowed to use UML constructs which do not explicitly define the order of fields (e.g. when using a number of associations to structure member types). {color:red}To create an inline array definition of a structure member, it must be marked with "Attribute is a Collection" in the EA Attribute Detail view.{color}

Defining Unions

Enter the new elements "Properties Details" dialog, click "Attributes..." and add any union fields you wish with their corresponding types. It is not required to place stereotypes to the fields. {color:red}To create an inline array definition of a union member, it must be marked with "Attribute is a Collection" in the EA Attribute Detail view.{color}

Defining Enumerations

Enter the new elements "Properties Details" dialog, click "Attributes..." and add any enumeration fields you wish with their corresponding data type. {color:red}Since Franca 0.9 string constants are deprecated for enums. To simplify usage of enums in UML, enumeration members may have an empty type field to indicate usage of integers.{color} All fields have to use the same (or empty) data type! It is not required to place stereotypes to the fields. The order of declared members within the enumeration is retained during any model transformation or code generation process. Optional: if a field shall have a constant value, provide it as _"initial value"_. Numeric initial values may be provided in decimal, octal, binary or hexadecimal notation.

Note: Franca currently (Franca 0.8.8) supports String and numeric enumeration constants, while most programming languages like C++ do support only numeric enumerations. It is recommended to NOT to use string constants for enumerations (see [Franca issue 52|http://code.google.com/a/eclipselabs.org/p/franca/issues/detail?id=52]).

Note2: Enterprise Architect contains some implicit logic for UML::enumeration typed elements. They are considered as type "Enumeration", not type "Class". They cannot be extended as Franca allows. Furthermore, if an element of type Class (which can be extended) is created with stereotype "enumeration" and you copy-paste this object, it is transformed to an element of type Enumeration and loses ability to be extended. To overcome this annoying behaviour it is proposed to use UML::Class with stereotype «FrancaEnum».

Defining Maps

Enter the new elements "Properties Details", click "Attributes..." and add the map´s key as member, provide its data type and use stereotype *«key»*. Then add the map´s value as member, provide its data type and use stereotype *«value»*. It is not allowed to define any other members.

Defining Arrays

There are two different kinds of arrays, anonymous _inline arrays_ and named data types defining an array. || anonymous inline array || named array || | {code} struct MyStruct { String [] myarray } {code} | {code} array StringArray of String {code} | {color:red}To create an anonymous inline array of an attribute, structure- or union member in UML, check "Attribute is a Collection" option in its EA Detail view and enter _Multiplicity_ (e.g. "0..*") to make EA show the attribute with square brackets. Inline arrays of method parameters are marked using stereotype *«array»*.{color} | To create a named array in UML, add a directed association with stereotype *«arrayOf»*, oriented from the new array data type to the origin data type.

Inheritance

Add a generalization with stereotype *«extends»*, oriented from the new derived data type to the origin data type. Only single inheritance is allowed, multiple inheritance is forbidden according to Franca specification 0.3.0.

Interfaces

Interfaces are placed within UML packages by selecting a package in the EA project browser, right click \-> Add \-> Add Element and configure the type "Interface".{color:red}{-} and stereotype *«interface»* in the dialog box.-{color} The stereotype shall be *«FrancaServiceInterface»* in order to distinguish between Franca service interfaces and other types of interfaces, e.g. internal library calls. The service interface is mandatory in UML even if it would be empty (e.g. because only broadcasts are to be defined). It may only contain methods and attributes, but no broadcasts.

interface version specification

Interface versions can optionally be configured in EA interface properties dialog and shall be specified like _major.minor_, with major and minor containing only numbers.

{code:title=interface version specification "4.2"|borderStyle=solid} version { major 4 minor 2 } {code} The semantics to use for interface version changes is specified in [1].

Inheritance

Inheritance between interfaces is expressed by an "Generalization" arrow. Only single inheritance is allowed, multiple inheritance is forbidden according to Franca specification 0.3.0.

Attributes

To add an attribute to an interface, select the interface in the EA project browser and right click \-> "Attributes..." to enter the dialog box. Configure stereotype as of one of *«attribute»*, *«attribute readonly»*, *«attribute nosubscriptions»* or *«attribute readonly nosubscriptions»*. Valid name and type properties have to be specified, too. Attributes will appear as items with a blue icon in EA. {color:red}To create an inline array definition of an attribute, it must be marked with "Attribute is a Collection" in the EA Attribute Detail view.{color}

Methods

To add a method to an interface, select the interface in the EA project browser and right click \-> "Operations..." to enter the dialog box. Configure stereotype as of one of *«method»* or *«method fireandforget»*. To specify parameters and return values, use the edit button near the parameters text field. Method call arguments must be specified as direction "in" parameters. Please note that Franca IDL allows to have many parameters as return values, so return values have to be specified as parameters with direction "out". Methods will appear as items with a pink icon in EA. {color:red}Inline array definitions for method parameters must be marked with stereotype *«array»*.{color}

method errors

To add an error return value to a method, configure method parameters with the *«error»* stereotype and direction out. There are several use cases that can be specified for a method error. Franca IDL allows maximum one Franca error to be specified per method, so even if there are several error parameters specified in UML they will be mapped to one Franca error.

1. reference to a predefined enumeration

Create method parameter with stereotype «error» and select a predefined enumeration type from your UML model for "type" field. The default value of the parameter must be left empty.

1. anonymous enumeration of error codes

Create method parameter with stereotype «error» and empty "type" field. Several error arguments like this may be specified for the same method to support multiple alternative error codes to be returned. If constant values for the error codes are required they may be specified within the "default value" field optionally.

1. reference to a predefined enumeration with additional extensions

Combine UML error parameter entries from usecase 1 and 2 within a method.

Note: Any restrictions that apply to definition of enumerations are also valid for method errors.

Broadcasts

In general broadcasts are multicast signals sent from a server component to a client component. Franca IDL specifies attributes, methods and broadcasts within one Franca interface. In UML (and implementations) there must be separate service (sender) and broadcast (receiver) interfaces in order to provide attributes/methods and receive broadcasts. This is opposite logic between UML and Franca IDL.

The solution to this problem is to move all broadcasts of a Franca interface to a separate broadcast interface. The broadcast interface shall have the same name as the service interface (which will have the name of the Franca interface) with additional _"Client"_ as suffix. For example, Franca interface Foo would be modeled as a pair of Foo (with attributes/methods) and FooClient (with broadcasts) in UML. The broadcast interface shall have the stereotype *«FrancaClientInterface»*. The broadcast interface is optional, if no broadcasts shall be specified for a Franca interface it can be left out completely. Broadcast interfaces may be empty. In order to model the relationship between these two UML interfaces, an undirected "association" relation from broadcast interface to service interface is created to define the relationship to the service interface. The association has to use the stereotype *«FrancaInterface»*.

!Interfacebroadcast.jpg|border=1!

Furthermore, the broadcasts in the broadcast interface shall be modeled as methods with stereotype *«broadcast»* or *«broadcast selective»*.

To add a broadcast to a broadcast interface, select the broadcast interface in the EA project browser and right click \-> "Operations..." to enter the dialog box. Configure stereotype as of one of «broadcast» or «broadcast selective». Several "in" parameters may be specified, "out" parameters are invalid. To specify parameters, use the edit button near the parameters text field. Broadcasts will appear as items with a pink icon in EA. Regular methods and attributes are not allowed in broadcast interfaces.

Managed Interfaces

{color:red}Franca can express that one kind of interface is responsible for managing another type of interface. An example could be an USB host controller interface, which is responsible for managing USB device profile interfaces. A management relation can be expressed using an directed association with stereotype *«manages»*.{color}

Modeling Sequences

Using above rules is not sufficient to draw UML sequence diagrams. Although regular method calls and broadcasts can be pictured, attribute get/set access or change events require additional conventions.

Attribute Access via get/set methods

Enterprise Architect supports _Properties_ to automatically generate get/set accessory methods for attributes. To create property accessor methods for a given attribute (already defined as specified above) is done by setting the check box "Property" in the interface attribute property dialog (select an attribute, then right click "Attribute Properties...").

Attribute Access via notification

Franca also offers the semantics of attribute change notifications, which can be specified in UML using «attribute» or «attribute readonly» stereotypes. In order to support modeling of UML sequences with attribute change notifications, it is required to define one receiver method in the client interface for each attribute offering change notifications and provide the stereotype *«callback»*. The name of the change notification method should follow a naming rule, which makes mapping to its adjacent attribute easy. Any method in the client interface with stereotype «callback» will not be visible in Franca, it is just a workaround to support sequence diagrams.

Example

| {code:title=Example UML tree|borderStyle=solid} |_GENIVI Model |_«root» Logical View |_SW Platform Components |_«fidl» MyComponent |_«interface» TestInterfaceA |_«attribute» myattr |_datatypes |_«struct» MyStructType |_MySubComponent |_«interface» TestInterfaceB |_«attribute» x {code} | {code: title=resulting Franca-IDL file: "Logical_View/SW_Platform_Components/MyComponent.fidl"|borderStyle=solid} package Logical_View. SW_Platform_Components.MyComponent

typeCollection datatypes { struct MyStructType { } }

interface TestInterfaceA { attribute datatypes.MyStructType myattr }

interface MySubComponent.TestInterfaceB { attribute datatypes.MyStructType x } {code} |

This example will produce one Franca IDL file named _"Logical_View/SW_Platform_Components/MyComponent.fidl"_, which contains both interfaces and a type collection named datatypes. Its Franca package name is _"SW_Platform_Components"_.

Summary of Stereotypes

UML Stereotype

Status	Used on UML Type	Description	
«array»	agreed	Class	define a named array data type
«array»	agreed	method parameter	define an anonymous inline array
«arrayOf»	agreed	Connector	associate the referred type to specify an array upon
«attribute»	agreed	Class	define an attribute with read/write access and notification
«attribute readonly»	agreed	Class	define an attribute with read only access and notification
«attribute nosubscriptions»	agreed	Class	define an attribute with read/write access without notification
«attribute readonly nosubscriptions»	agreed	Class	define an attribute with read only access without notification
«broadcast»	agreed	Class	define a broadcast method as part of a FrancaClientInterface
«broadcast selective»	agreed	Class	define a selective broadcast method as part of a FrancaClientInterface
«enumeration»	deprecated	Class	define an enumeration data type
«error»	agreed	Class	define method error codes
«extends»	agreed	Connector	inherit from another interface
«fidl»	agreed/TBD	Package	mark an UML package and its sub tree as contents to be represented within a Franca FIDL file. TBD für Franca versions >= 0.9.0
«FrancaEnum»	agreed	Class	define an enumeration data type
«FrancaClientInterface»	agreed	Class	mark an UML interface as a Franca client interface which receives broadcast notifications
«FrancaInterface»	agreed	Connector	specify association between FrancaServiceInterface and FrancaClientInterface parts
«FrancaServiceInterface»	agreed	Class	mark an UML interface as a Franca service interface
«key»	agreed	Class	define a map key

«manages»	tbd	Class	specify managed interface relationships
«map»	agreed	Class	define a map data type
«method»	agreed	Class	define an interface method
«method fireandforget»	agreed	Class	define an interface method which does not return any results nor execution status
«nofidl»	agreed	Package	exclude an UML package and its sub tree from Franca contents specification
«root»	agreed	Package	define the root of a Franca namespace
«struct»	agreed	Class	define a structure data type
«struct polymorphic»	proposed	Class	define a structure data type which supports polymorphism if used as method parameter or attribute
«typedef»	agreed	Class	define a named data type
«union»	agreed	Class	define a union data type
«value»	agreed	Class	define a map value

Glossary

FQN: Fully qualified name

References

Specification 0.3.0