IFML 1.0 Finalization Task Force

Ballot No. 1

IFML 1.0 FTF Formal Issues Resolution Vote - Ballot No. 1 -

Poll start date: Friday, 31 January 2014 (01:00 AM EST - 06:00 GMT)
Poll closing date: Wednesday, 5 February 2014 (7:00 PM EST - 24:00 GMT)

What is being voted on: Proposed issue resolutions for the set of issues listed in the tables on the following pages. The proposer is listed for each resolution, the full text of the issues and corresponding resolutions can be found in the section following the issue tables.

- Only officially registered FTF members in good standing are allowed to vote
- Voters who do not vote in two successive ballots lose their good standing and will be removed from the FTF membership; they can only be reinstated by a TC vote
- Quorum for a vote is half the registered FTF members
- Simple majority (of non-abstaining votes) decides the vote
- Votes should be sent via email to the chair of the FTF marco.brambilla@webratio.com as well as to the ifml-ftf@omg.org list.
- Members can submit their vote anytime between the poll start date and the poll closing date.
- During the polling interval, members are allowed to change their votes. The most recent vote will be assumed to supersede all previous votes cast by a member.
- The possible ways to vote are:
  – Yes
  – No
  – Abstain (Note: "Abstain" does not influence the voting result but does count towards quorum)
- Votes can be cast either for individual issues or for the entire block (but not a mix of both)

Block Vote
<Company name> votes {Yes | No | Abstain} for the entire block of proposed issue resolutions identified below and specified in the appendix to this document.

Individual Issues Vote (NB: ONLY if you choose not to use the Block Vote option above!)
<Company name> votes as follows on the proposed issue resolutions specified in and specified in the appendix to this document. Then vote for one issue resolution per line, like this:
12345 {Yes | No | Abstain}

IFML 1.0 FTF Voting Members

Representative	**Organisation** (alphabetic order)	**Status**
Manfred Koethe	88solutions	Charter
Hiroshi Miyazaki	Fujitsu	Charter
Ed Seidewitz	Ivar Jacobson International AB	Charter
Maurice Nijssen	PNA Group	Added July 19, 2013
Andrey Sadovykh	Softeam	Charter
James D. Baker	Sparx Systems	Charter
David Faure	Thales	Charter
Marco Brambilla	WebRatio	Charter (CHAIR)

Contents

Disposition: Resolved

Disposition: Resolved

OMG Issue No: 18574

Title: Two of the XMI files use the XMI 2.1 namespace instead of XMI 2.4

Source:

Adaptive (Mr. Pete Rivett, pete.rivett (at) adaptive.com)

Summary:

Two of the XMI files use the XMI 2.1 namespace instead of XMI 2.4

Resolution:

XMI namespace version has been fixed to 2.4

Revised Text:

(Nontext)

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18915

Title: Content of Field Class

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Field can contain any sub-viewcomponentpart, not only slot. Remove rule from Field Class:
Constraints
• viewComponentPartsAreSlots
self.subViewComponentPart -> forAll (v | v.oclIsTypeOf(Slot))

Resolution:

Constraint Removed.

Revised Text:

Constraint Removed in clause 8.4.3 - “Field”:

viewComponentPartsAreSlots
self.subViewComponentPart -> forAll (v | v.oclIsTypeOf(Slot))

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18916

Title: Fix Figure 50

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

In figure 50: The model of the interaction flow for moving a message to an existing or newly created tag:

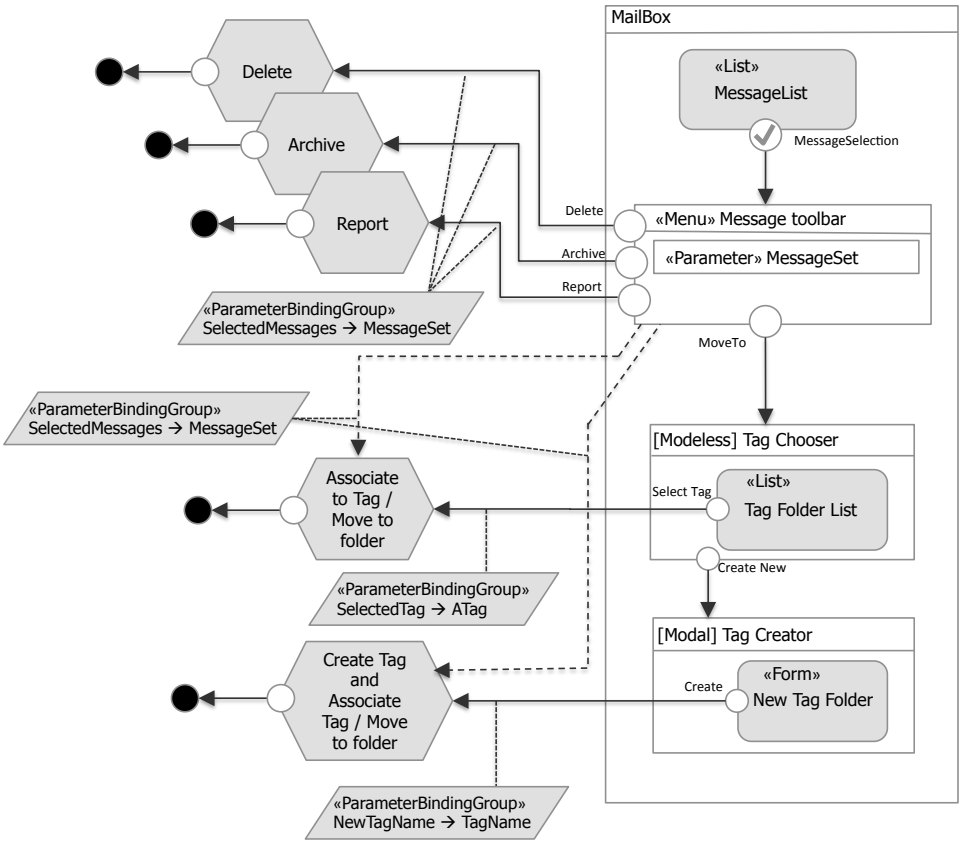
- the XOR container is not needed
- add the needed dataflow to the second action CreateTag
- the missing parent Folder must be added
- fix the newPage event

Resolution:

Figure 50 fixed by: removing the XOR container and adding the missing DataFlow.

Revised Text:

Replace figure 50 with:



Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18917

Title: Parameter scope

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Define the scope of parameters. It's not clear where parameters can be referenced.
A sensible rule can be: Scope of the parameter is the component or container where defined, plus everything contained in it.

Resolution:

Scope and type of parameters are clarified in Clause 8.3.27 in a new piece of text.

Revised Text:

New text added at the end of Clause 8.3.27 - Parameter:

The scope of a Parameter (i.e., the model space where it can be used or referenced) is the InteractionFlowElement that holds the Parameter, plus the incoming and outgoing InteractionFlows. This means that: if the parameter is held by a ViewComponent, it can be referenced only within the ViewComponent itself and the contained ViewComponentParts (plus the incoming and outgoing InteractionFlows); if the parameter is held by a ViewContainer, it can be referenced within the ViewContainer itself, and within the contained ViewContainers, ViewComponents, and ViewComponentParts (plus the incoming and outgoing InteractionFlows).

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18918

Title: actions in containers

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

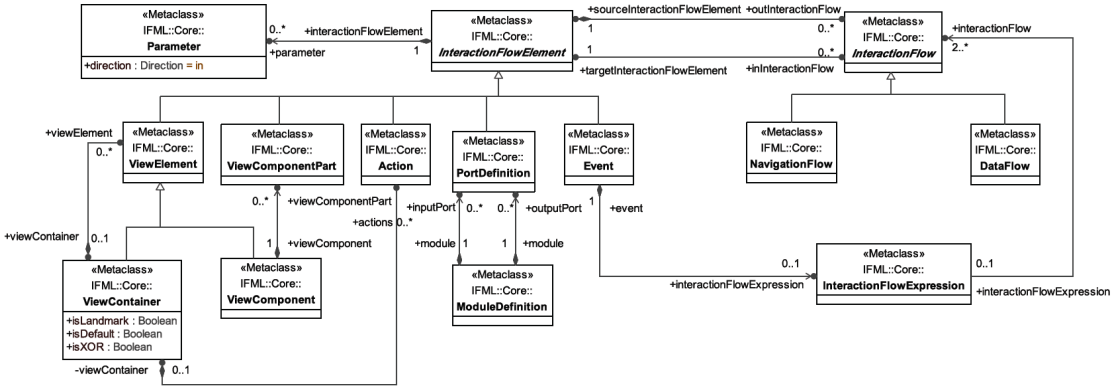
ViewContainers cannot contain Actions. This should be possible.

Resolution:

Add containment relation between ViewContainer and Action. In clause 8.1.3 figure and text have been updated.

Revised Text:

In clause 8.1.3.
Replaced Figure 7 with this:



Added as last sentence in text:

ViewContainers can contain ViewElements (namely other ViewContainers or ViewComponents) or Actions.

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18919

Title: module defs

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

The spec misses the concept of ModuleDefinition, so one can use a module but not define it.
Module definitions should come with a packaging mechanism too.

Resolution:

Add ModuleDefinition and Module Package. Module renamed as ModuleDefinition (as a NamedElement); added ModularizationElement, PortDefinition and ModulePackage in metamodel, with composite pattern. Added respective descriptions. Modules refer to ModuleDefinitions. Flows in PortDefinition can only be connected to elements within the ModuleDefinition. Flows in PortDefinition can only be connected to elements outside the Module. The set of Parameters of the Port is derived from the set of Parameters of the corresponding PortDefinition.
Created new general clause for describing the new concepts and showing the metamodel fragment. Created all the specific clauses needed for describing the new concepts.

Revised Text:

Created new Clause:

8.1.11 – Modularization

Added figure of metamodel fragment describing the new modularization concepts:

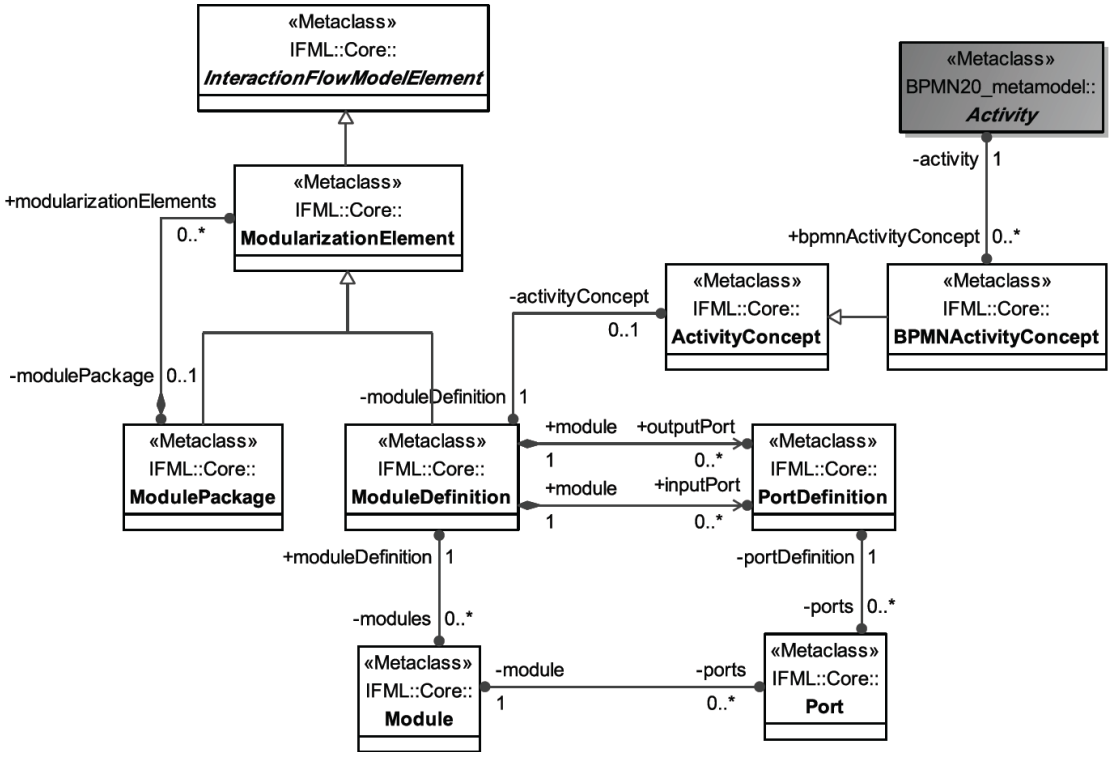


Figure 15: Modularization concepts in IFML

Added respective descriptions in the new document Clause 8.1.11, as follows:

IFML includes a set of concepts that help improving reuse and modularization of models.
The main concept is ModuleDefinition, which allows the definition of an arbitrary piece of IFML model, which can be subsequently reused in models. ModuleDefinitions can be aggregated in a hierarchical structure of ModulePackages. ModuleDefinitions may comprise PortDefinitions. PortDefinitions represent interaction points with a Module. PortDefinitions hold Parameters, for transferring values to and from the ModuleDefinition. An input PortDefinition has outgoing InteractionFlows to the inside of the Module. An output PortDefinition has incoming InteractionFlows from the inside of the Module.
Module is the concept that enables reuse of ModuleDefinitions. Module has a reference to the relevant ModuleDefinition and is associated with a set of Ports, which in turn reference the corresponding PortDefinitions. An input Port (i.e., a Port referencing an input PortDefinition) has incoming InteractionFlows from the outside of the Module, for receiving input Parameters. An output Port has outgoing InteractionFlows to the outside of the Module, for shipping output Parameters.

Added specific sub-Clauses in Clause 8.3, with following descriptive texts:

8.3.33Class ModularizationElement

Abstract: No

Generalization:

- InteractionFlowModelElement
- NamedElement

Description

A ModularizationElement is an abstract concept that represents both ModulePackages and ModuleDefinitions.

Association Ends

- modulePackage [0..1]: ModulePackage - The ModulePackage containing the ModularizationElement

8.3.35Class ModuleDefinition

Abstract: No

Generalization:

- ModularizationElement

Description

A ModuleDefinition is a fully functional collection of user InteractionFlowModelElements and their corresponding Actions, which may be reused for improving IFML model maintainability. ModuleDefinitions can be aggregated in a hierarchical structure of ModulePackages. ModuleDefinitions may comprise PortDefinitions. A ModuleDefinition receives Parameter values from outside and provides Parameter values to the outside. ModuleDefinitions exchange Parameters by mean of input and output PortDefinitions. InteractionFlowModelElements contained in a Module may not be shared or referenced by other Modules or by the main InteractionFlowModel. A ModuleDefinition may comprise a reference to a BPMN Activity (meaning that the Module implements that Activity). Reuse of ModuleDefinition is obtained by adding Modules referencing that ModuleDefinition in IFML models.

Association Ends

- inputPort [0..*]: PortDefinition - Ports that distributes InteractionFlows and Parameters coming into the Module.

- interactionFlowModelElement [1..*]: InteractionFlowModelElement - InteractionFlowModelElements contained by the Module.

- outputPort [0..*]: PortDefinition - Ports that collect the InteractionFlows and Parameters going out from the Module.

- modules [0..*]: Module - The set of Modules that are defined in the IFML model and reference the current ModuleDefinition

- activityConcept [0..1]: ActivityConcept - Reference to a process activity (e.g., a BPMN Activity). If present, the current module is describes the technical implementation of the process activity

8.3.36Class ModulePackage

Abstract: No

Generalization:

- ModularizationElement

Description

A ModulePackage is a container of ModuleDefinitions. ModulePackages can be nested in arbitrarily deep hierarchical structure.

Association Ends

- modularizationElements [0..*]: ModularizationElement – Set of ModularizationElements contained in the current ModulePackage

8.3.43Class PortDefinition

Abstract: No

Generalization:

- InteractionFlowElement

Description

PortDefinitions represent interaction points with a ModuleDefinition. They are defined within a ModuleDefinition. They hold Parameters, for transferring values to and from the ModuleDefinition. An input PortDefinition has outgoing InteractionFlows to the inside of the Module. An output PortDefinition has incoming InteractionFlows from the inside of the Module. Modules that reference a ModuleDefinition may comprise Ports, which in turn reference the corresponding PortDefinitions.

Association Ends

- ports [0..*] :Port - Set of Ports referencing the current PortDefinition in some Modules implementing the ModuleDefinition within which the current PortDefinition is defined

Modified descriptive text of existing sub-Clauses in Clause 8.3:

8.3.34Class Module

Abstract: No

Generalization:

- InteractionFlowModelElement
- NamedElement

Description

A Module is a named reference to a ModuleDefinition, which allows reuse of the model part specified in the ModuleDefinition. Module has a reference to the relevant ModuleDefinition and may be associated with a set of Ports, which in turn reference the corresponding PortDefinitions. For every PortDefinition in the ModuleDefinition there shall be 0 or 1 Ports in each corresponding Module. An

input Port (i.e., a Port referencing an input PortDefinition) has incoming InteractionFlows from the outside of the Module, for receiving input Parameters. An output Port has outgoing InteractionFlows to the outside of the Module, for shipping output Parameters.

Constraints

- onlyOnePortPerPortDefinition
self.moduleDefinition.portDefinitions -> forAll(pd | pd.ports -> select(p|p.module = self) -> size() = 1)

Association Ends

- Port s[0..*]: Port - Ports that collect InteractionFlows and Parameters incoming or outgoing from the Module.
- moduleDefinition [1] :ModuleDefinition - The ModuleDefinition that is instantiated by the current Module

8.3.42Class Port

Abstract: No

Generalization:

- InteractionFlowElement

Description

A Port is an interaction point between a Module and the surrounding model within which it is defined.

Module is associated with a set of Ports, which in turn reference the corresponding PortDefinitions. An input Port (i.e., a Port referencing an input PortDefinition) has incoming InteractionFlows from the outside of the Module, for receiving input Parameters. An output Port has outgoing InteractionFlows to the outside of the Module, for shipping output Parameters.

Association Ends

- portDefinition [1] :PortDefinition – Reference to the PortDefinition that defines the interface of the current Port

- module [1] : Module - Module that contains the current Port

Disposition: Resolved

Disposition: Resolved

OMG Issue No: 18920

Title: Default container

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

In ClassViewContainer: The property Default can be true only for containers within a XOR container. One and only one Default subcontainer must be defined for a XOR container.
Consider the option of transforming Default into an association from XOR container to one of its subcontainers.

Resolution:

Added OCL condition on ViewContainer and fixed describing text.

Revised Text:

In clause 8.3.34 - ViewContainer:
In isDefault, added:
This attribute is relevant when this ViewContainer shares the same parent ViewContainer with other ViewContainers, and the parent ViewContainer has property isXOR = true.
In isXOR, added:
If true, the contained ViewContainers of this ViewContainer will be presented to the user only one at the time, as the user interacts with the system. One of the contained ViewContainers must have attribute isDefault = true.

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18921

Title: Add Window in table 1

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Add Window in the list of main concepts in Table 1.

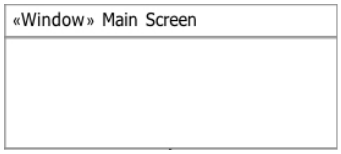
Resolution:

Added new row in Table 2 (not Table 1) with Window, respective symbol, description and example.

Revised Text:

New text in Table 2:
Window - A ViewContainer rendered as a window. - An HTML page or a desktop window.

Window symbol added in third column:



Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18922

Title: Modality in Table 1

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Modal and Modeless containers must be tagged as <<Window>> in Table 1.

Resolution:

Instead of adding tag Window, they must be stereotypes themselves.

Revised Text:

Symbols have been fixed by replacing [Modal] and [Modeless] with <<Modal>> and <<Modeless>> in all the figures of the specification document.

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18923

Title: add event type onLoad

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Add a new event type onLoad, applicable to containers and components.

Resolution:

added onLoadEvent metaclass. Correspondingly, all the other catching events have been renamed by adding the “On” prefix in front.
Specific events are discussed in Clause 8.1.6. Events, and therefore their discussion is removed from Clause 8.1.10 Specific Events and ViewComponents

Revised Text:

Added new clause for OnLoadEvent:

8.4.9Class OnLoadEvent

Abstract: No

Generalization:

- SystemEvent

Description

An OnLoadEvent is triggered by the system when a ViewElement is completely computed and rendered.

Removed Text from Clause 8.1.10 Specific Events and ViewComponents. New text:

8.1.10 Specific ViewComponents

IFML includes a basic set of extensions to the core elements that exemplify how IFML may be extended.
List, Details and Form are specializations of ViewComponent (see 8.1.4). The List ViewComponent is used to display a list of DataBinding instances. When a List ViewComponent is associated with an Event, it means that each DataBinding instance displayed by the component may trigger that Event. The Event will in turn cause the passing of the parameter values mapped to the DataBinding instance to a target InteractionFlowElement. The Details ViewComponent is used to display detailed information of a DataBinding instance. When the Details ViewComponent is associated with an Event, the triggering of the Event will cause the passing of the Parameter values mapped to the DataBinding instance to a target InteractionFlowElement. The Form ViewComponent is used to display a form, which is composed of Fields that may display or capture content from the user. Fields have Slots that hold their value. When the Field is a SelectionField, its

associated Slots contain the available selection options and the selected one. When the Field is a SimpleField, the Slot contains the Field value. A Slot value of a SimpleField and the Slots corresponding to the selected options of SelectionFields also behaves as Parameters in order to be passed to other ViewElements or Actions when an Event is triggered. Form ViewComponents have ValidationRules, which determine if a Field value is valid or not.

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18924

Title: Fix Figure 64

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Figure 64 contains a RED [H] tag. Make it black and add the stereotype <<Window>> to all the containers.

Resolution:

This and other figures still contained colored elements. All figures have been transformed in black and white (with grayscale filling of ViewComponents and Modules).

Revised Text:

(Nontext)

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18925

Title: An List - fix typo

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Fix at page 88: An List –> A List

Resolution:

typo corrected

Revised Text:

A List

Disposition: **Resolved**

Disposition: Resolved

OMG Issue No: 18926

Title: Action description

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Action Description (page 30): replace InteractionViewElement with InteractionFlowElement.

Resolution:

Typo fixed and sentence specified better, for clarity reasons.

Revised Text:

In clause 8.3.1 Class Action, the new text is:

8.3.1Class Action

Abstract: No

Generalization:

- InteractionFlowElement
- NamedElement

Description

An Action is an InteractionFlowElement that represents a piece of business logic triggered by an Event. Actions may reference behavior models describing the actual business logic to be performed. Actions may trigger different Events called ActionEvents as the result of business logic computation termination or the occurrence of exceptions. Actions may reside on the server or on the client side. If no ActionEvent (and corresponding outgoing flow) is specified, IFML assumes as default an ActionEvent and NavigationFlow that lead back to the ViewComponent or ViewContainer from which the navigation to the Action was launched.

Constraints

- actionsCannotCallActions
self.actionEvent->forAll(e | e.navigationFlow->forAll(nf | not nf.targetInteractionFlowElement.oclIsTypeOf(IFML::Core::Action)))

Association Ends

- actionEvent [0..*]: ActionEvent - Events triggered by the Action.

- dynamicBehavior [1]: DynamicBehavior – The business logic to be carried out by the Action.

- viewContainer [0..1]: ViewContainer – The ViewContainer that contains the current Action.

Disposition: Resolved

Disposition: Resolved

OMG Issue No: 18927

Title: Menu ViewContainer

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

Add a new viewcontainer called Menu in the metamodel, as an extension, together with Window.
Peculiarity: only contains events, no sub-containers or components

Resolution:

Menu class added as a specialization of ViewContainer, with OCL constraint and describing text.

Revised Text:

New Clause added:

8.4.8Class Menu

Abstract: No

Generalization:

- ViewContainer

Description:

A Menu is a special kind of ViewContainer used to model the concept of a menu of options in IFML. It cannot contain ViewComponents or sub-ViewContainers.

Constraints

- self.viewElements -> select(e | e.oclIsTypeOf(ViewContainer) or e.oclTypeOf(ViewComponent)) -> isEmpty()

Disposition: Resolved

Disposition: Resolved

OMG Issue No: 18928

Title: wrong component in figure 54

Source:

WebRatio (Dr. Marco Brambilla, marco.brambilla (at) webratio.com)

Summary:

In Figure 54: Message Full Search viewcomponent must be replaced by <<Form>> Message Writer view component. The send event must be a submit event.
Also fix the description of Figure 54 accordingly (and mention MessageWriter as a view container, not component)

Resolution:

Figure fixed by replacing the <<Form>> component as requested. “Message Full Search” text in <<Form>> has been replaced with “Message Writer”. Send Event changed to Submit event. “Message Keyword Search” in <<Details>> replaced with “Message Details”.
Caption of Figure 53 fixed: MessageReader → MessageDetails

Revised Text:

Figure 54 is replaced by:

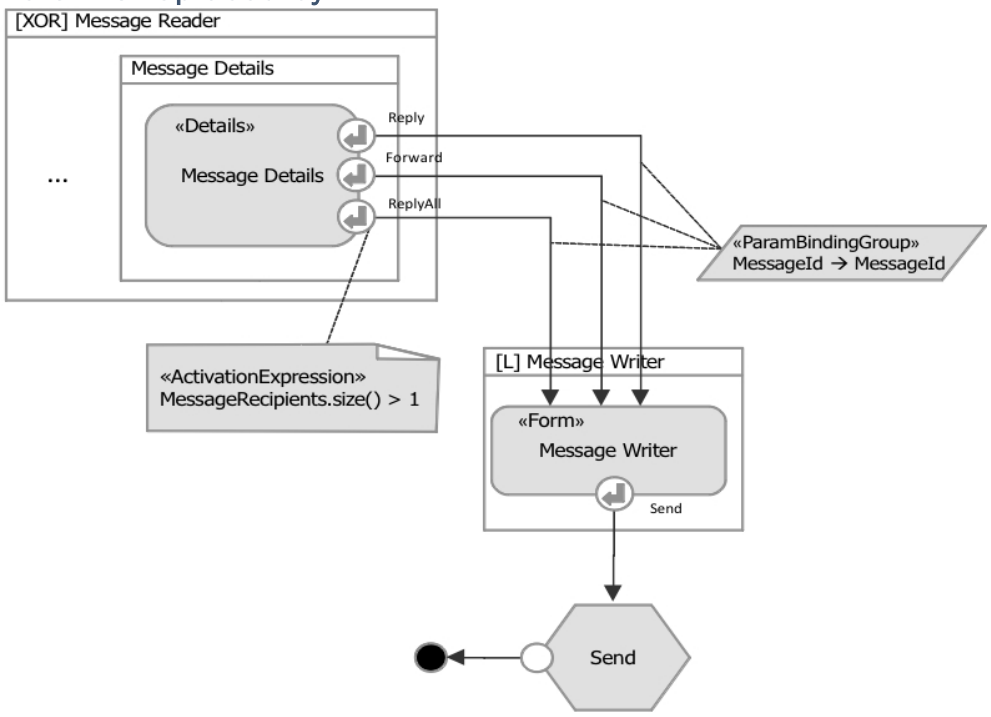


Figure 54 caption is now:

The *MessageList* and the *MessageDetails* view components are shown in alternative

Textual description of Figure 55: Message reader replaced by:

OMG Issue No: 18928

MessageDetails

Disposition: **Resolved**