FTF Report

of the

UML for EAI
Finalization Task Force (revision 2)

to the

Platform Technical Committee

of the

Object Management Group

August 18, 2003

Document Number: ptc/2003-08-11
Accompanied by: ptc/2003-08-13

Template: omg/03-01-07

Table of Contents

Summary of UML for EAI FTF Activities

Formation

- Chartered By: PTC
- On: 16 November 2001 at Dublin
- Comments Due Date: 1 July 2002
- Report Due Date: 25 November 2002, revised to 15 April 2003 to fit in with UML for EDOC

Revision / Finalization Task Force Membership

Member	Organization	Status
Rob Phippen	IBM	Chair, charter
Akira Tanaka	Hitachi	Charter
Ed Seidewitz	Intelidata	Charter, added 1 October 2002
Cory Casanave	DAT	Charter
Dave Frankel	Iona	Charter, removed 1 October 2002
Dai Clegg	Oracle	Charter

Initial Issues from Architecture Board Review:

Many issues were raised by Pete Rivett, a member of the AB, but the normal comments process was used rather than specifically raising them through the AB.

Issue Disposition:

Disposition	Number of Occurrences	Meaning of Disposition
Resolved	113	The RTF/FTF agreed that there is a problem that needs fixing, and has proposed a resolution (which may or may not agree with any resolution the issue submitter proposed)
Unresolved	15	The RTF/FTF agrees that there is a problem that needs fixing, but could not agree on a resolution. This is still work in progress.
Deferred	0	The RTF/FTF agrees that there is a problem that needs fixing, but decided to defer its resolution to a future RTF working on this specification (perhaps because of a lack of time or urgency).
Transferred	10	The RTF/FTF decided that the issue report relates to another specification, and recommends that it be transferred to the relevant RTF.
Closed, no change	7	The RTF/FTF decided that the issue report does not, in fact, identify a problem with this (or any other) OMG specification.
Duplicate or merged	15	This issue is either an exact duplicate of another issue, or very closely related to another issue: see that issue for disposition.

Voting Record:

Poll No.	Closing date	Issues included
1	26 June 2002	4880, 4882, 5248, 4857, 5367, 5380, 4884
2	20 September 2002	5397-5402
3	3 October 2002	5237-44
4	4 February 2003	4878, 4879, 5242, 5345, 5347-49, 5255, 5358, 5259, 5246, 4872, 4868, 5859, 5370-74, 5250, 5352, 5353, 5357, 5387-96, 5405-09, 5952-54, 4965-67, 4958, 4975, 4961, 4955-57, 4971, 4973, 4965, 4947, 4949, 4950-51, 4874, 4948, 5404, 4976-77
5	20 February 2003	5383, 5381, 5378-79, 5368-69, 4896, 4864, 5354, 4894, 4876-77, 5366
6	26 February 2003	4881, 4883, 5245-47, 5222-25, 5249-52, 4854, 5385-86
7	1 May 2003	5384, 5365, 5364, 5363, 5362, 5361, 5360, 4865, 4866, 4867, 4875, 5375, 5376, 5377, 4869 4897 4898 5384 5351 Done
8	6 May 2003	5353, 5356, 5357
9	8 May 2003	4862, 4968, 4969, 4970, 4972, 4974, 4978
10	14 July 2003	4856, 4858, 4863, 4892, 4893, 5382

Voter	Vote in poll 1	Vote in poll 2	Vote in poll 3
Rob Phippen	Yes	Yes	Yes
Akira Tanaka	Yes	Yes	Yes
Ed Seidewitz	-	Yes	Yes
Cory Casanave	Yes	Yes	Yes

Dave Frankel	Did not vote	Did not vote	-
Dai Clegg	Yes	Yes	Yes

Voter	Vote in poll 4	Vote in poll 5	Vote in poll 6
Rob Phippen	Yes	Yes	Yes
Akira Tanaka	Yes	Yes	Yes
Ed Seidewitz	Yes	Yes	Yes
Cory Casanave	Yes	Did not vote	Did not vote
Dai Clegg	Yes	Yes	Yes

Voter	Vote in poll 7	Vote in poll 8	Vote in poll 9
Rob Phippen	Yes	Yes	Yes
Akira Tanaka	Yes	Abstain	Yes
Ed Seidewitz	Yes	Abstain	Yes
Cory Casanave	Yes	Did not vote	Yes
Dai Clegg	Yes	Did not vote	Did not vote

Voter	Vote in poll 10		
Rob Phippen	Yes		
Akira Tanaka	Yes		
Ed Seidewitz	Yes		
Cory Casanave	Did not vote		
Dai Clegg	Did not vote		

Summary of Changes Made

The UML for EAI FTF made changes that:

- Clarified the specification of the integration metamodel by adding OCL constraint specifications
- Transferred issues to the EDOC FTF to clarify the Flow Composition Model
- Modified the Integration Metamodel to integrate changes to the Flow Composition Model
- Removed unnecessary 'operational' information from the EAI Integration Metamodel
- Updated the Common Application Metamodel to fix minor errors discovered during implementation
-

The following is a table that categorizes the issues as to the degree of changes that were made in resolving them.

Extent of Change	Number of Issues	OMG Issue Numbers
Significant – Fixed problems with normative parts of the specification that raised concern about implementability	**0**	**{enter OMG issue numbers here}**
Minor - Fixed minor problems with normative parts of the specification	**0**	**{enter OMG issue numbers here}**
Support Text -Changes to descriptive, explanatory, or supporting material.	**0**	**{enter OMG issue numbers here}**

Disposition: Resolved

OMG Issue No: 4854

Title: Purpose of the CCA Component Library for EAI

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The purpose of Section 6.5, "CCA Component Library for EAI" is not very clear. The text says that "It is an informational supplement to the EAI integration metamodel.", but I am not sure what this means, since CCA is basically a modeling approach with its own metamodel and profile. Does "informational" mean that this section is not even normative?
Recommendation:

The purpose of the CCA Component Library would seem to be to provide a CCA-based modeling notation for the metamodel. In this respect it would seem have a parallel purpose to the Collaboration and Activity Modeling profiles given in Chapters 8 and 9. Therefore, if this library is intended to be normative, then it should be included in Part 3 as a chapter on "CCA Modeling". If it is not intended to be normative, then this section should probably be moved to an appendix.

Resolution:

Change text as follows.

Revised Text:

The original text:

> This section specifies the CCA component library for EAI. It is an informational supplement to the EAI Integration metamodel.

Is replaced with:

> This section specifies the CCA component library for EAI and mapping between EAI and CCA concepts. CCA provides for the modeling of collaboration similar to the EAI models in Chapters 8 and 9. The component library specifies the set of components required in CCA to represent the same concepts as the EAI meta model. By providing this component library and mapping between EAI and CCA users may transform models between EAI and CCA tools, integrating EAI systems with collaborations modeled with CCA. This information may be used by EAI or CCA tool vendors to automate such transformation and integration or may be used directly by users in a manual process.

Disposition: **Resolved**

OMG Issue No: 4855

Title: Non-normative examples

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The examples in Chapters 10 and 11 are useful, but they are non -normative.

Resolution:

Revise Contents page of part 4.

Revised Text:

Add text:
These examples are non-normative; they do not appear as compliance points.

Disposition: **Resolved**

OMG Issue No: 4856

Title: Constraints should be in OCL

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Constraints are presented only as textual descriptions.

Recommendation: Express all constraints in OCL as well as textually.

Discussion:

I propose that we do not take 'special' action to resolve this issue independently but that we use OCL definitions where answering issues that relate to the EAI metamodel elements. Since issues cover all of the core elements of the EAI Integration Metamodel, this has the effect that OCL will be used throughout.

Disposition: **Accepted**

Revised text

Covered by the text of issues that revise the EAI Integration Metamodel (chapter 6)

OMG Issue No: 4857

Title: Deployment of the EAI Configuration

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The document does not address the deployment of different parts of the EAI configuration.

Resolution:

It would useful to have some kind of location binding for either EAIMessageFlow or for each EAI node.

Revised Text:

Section 6.3.10.3, add text at the end of paragraph 1:

> Annotations may be used to provide deployment information to a messageflow.

Disposition: Resolved

Description:
The document does not address the deployment of different parts of the EAI configuration.

Recommendation:
It would useful to have some kind of location binding for either EAIMessageFlow or for each EAI node.

Resolution:
Revised Text:
Section 6.3.10.3, added text at the end of paragraph 1;

> **Annotations may be used to provide deployment information to a messageflow.**

Actions taken:

OMG Issue No: 4858

Title: The class of an operator that is a subclass of a primitive operator

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Even though integration with the "outside world" is out of scope for the spec the examples of such integrations would be really helpful. For instance it's not clear from the DTD what tag/association defines the class of the operator that is a subclass of the one of the standard primitive operators. Is it 'defines association' to the FCMType? (And why does EAIRequestReplyAdapter have EAIPrimitiveOperator.defines but EAICallAdapter does not in the DTD)?

Discussion:

The resolution to this issue is covered by the resolution to issue 4892, which discusses the 'defines' association and the definition of types.

Disposition: **Accepted**

Revised text

See revise text for the resolution to issue 4892

OMG Issue No: 4859

Title: Superclass of EAIAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 6.3.11, 6.5

Description:
There is some confusion in the spec on the superclass for EAIAdapter. Section 6.3.11 says that EAIAdapter is a specialization of FCMFunction. Section 6.3.11.4 says that EAIRequestReplyAdapter is a subclass of FCMCommand. In section 6.5 we see that all adapters including EAIRequestReplyAdapter are a specialized EAIPrimitiveOperator.

Resolution:

All of these adapters should be subclasses of FCMFunction.

Revised Text:

Section 6.3.11: First paragraph, last sentence updated:

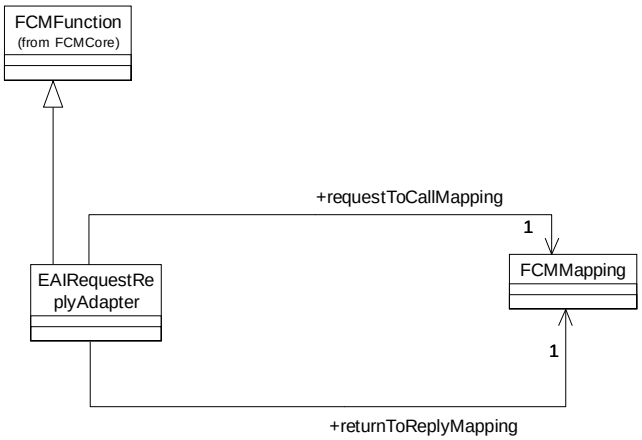
EAI adapters are modeled as a specialization of FCMFunction.

Figure 6-1

1. Remove inheritance from EAIPrimitiveOperator

2. Add inheritance from FCMFunction

Figure 6-1 after edit



First sentence of each of sections 6.5.2.1, 6.5.2.2, 6.5.2.3, 6.5.2.4, 6.5.2.5

Change "EAISourceAdapter is a specialized EAIPrimitiveOperator."

To "EAISourceAdapter is a specialized FCMFunction."

Disposition: Resolved

OMG Issue No: 4861

Title: EAIRequestReplyAdapter/EAICallAdapter temporary link

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.4

Description:
The "replyOut" terminal and the "handleReply" terminal of the EAICallAdapter are connected dynamically via the a temporary EAILink that is not part of EAI configuration. It appears that information required to create the temporary EAILink can be stored only in the message. The only data that is there now is information required to locate replyTo terminal (terminal id I assume). It's not clear what type of the link the temporary link should be: synchronous or asynchronous? What is the name of the queue in case of the asynchronous link? What is the type of the "replyOut" terminal EAITerminal or EAIQueuedOutputTerminal considering that this temporary link can be synchronous or asynchronous?

Resolution:

Update Section 6.3.11.5 (which is the correct section in the Final Adopted Specification, not 6.3.11.4)

Revised Text:

Replace the following paragraph:

> This effectively creates dynamic and temporary instances of EAILink between the "replyOut" terminal and the "handleReply" terminal of the EAICallAdapter that sent the request message.

with:

> Note that, in addition to simply being placed on the “replyOut” terminal, the reply message is transmitted to the reply terminal that is dynamically identified by the incoming request message. Request messages a generated by EAICallAdapters, with the reply terminal of the request message being the “handleReply” terminal of the EAUICallAdapter. Thus, the semantics of an EAIRequestReplyAdapter effectively results in the creation of a dynamic and temporary EAILink between the “replyOut” terminal of the EAIRequestReplyAdapter and the “handleReply” terminal of the EAICallAdapter that generated the request message.
>
> Now, if the identified reply terminal is *not* an EAIQueuedInputTerminal, then the dynamic EAILink is considered to have *synchronization* = *unspecified.*The reply message is simply placed on the identified input terminal. However, if the identified reply terminal *is* an EAIQueuedInputTerminal (see Section 6.3.8), then the dynamic EAILink is considered to have *synchronization* = *asynchronous* and the reply message is placed on the inputQueue of the reply terminal.

Disposition: Resolved

OMG Issue No: 4863

Title: EAICompoundOperator as a new type

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.2 (and DTD): The compound operator can be defined as a group of primitive operators but it's not clear how to define the compound operator as a new type using the FCM4EAI.dtd. Is usage of EAIMessageFlow the answer?

Discussion:

Disposition: **Accepted**

Revised Text:

Replace the text of section EAICompoundOperator with;

CompoundOperator

Description

An instance of an EAICompoundOperator composes more complex message processing behavior from EAIPrimitiveOperators, from other EAICompoundOperators or both. EAICompoundOperator inherits its 'composition' characteristics from FCMCompositeNode and its EAI-specific constraints from EAIOperator. Further constraints are described below

Constraints

context EAICompoundOperator

The EAIType of an EAICompoundOperator must have an association with an FCMComposition

self.type.fCMComposition->size() = 1

Define the implementingComposition derived association

let implementingComposition = self.type.fCMComposition->any()

The implementingComposition must be an EAIMessageFlow;

implementingComposition.oclIsKindOf(EAIMessageFlow)

Define the *nodes* derived association

self.nodes = self.implementingComposition.nodes

Define the FCMOperations implemented by the FCMComposition

let sourceNodes =

self.implementingComposition.nodes->

select(n | n.oclIsKindOf(EAISource))

let sourceOperations = sourceNodes.implements

The operations implemented by the EAISource nodes in the composite are the same as the operations specified for the EAIType of the node.

inv: sourceOperations = self.type.operations

OMG Issue No: 4868

Title: The relationship of EAILink to FCMDataLink and FCMControlLink

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.2 (EAILink)

Description:
This section implies that for any EAILink, there will really be THREE FCM links. The EAILink itself (an FCMTerminalToTerminalLink), an associated FCMDataLink and and associated FCMControlLink. The EAILink and the DataLink, in particular, seem redundant.

Resolution:

Since, conceptually, an EAILink is primarily a data link, it seems appropriate to make EAILink a child of FCMDataLink, rather than a direct child of FCMTerminalToTerminalLink. Then only one additional link, the FCMControlLink, is needed. This is less heavyweight and fits the conception of an EAILink as a data link that also implies a control link.

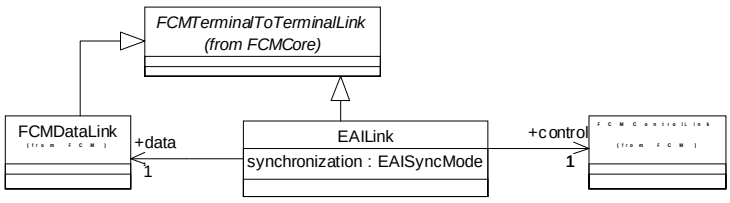
Revised Text:

Updates:

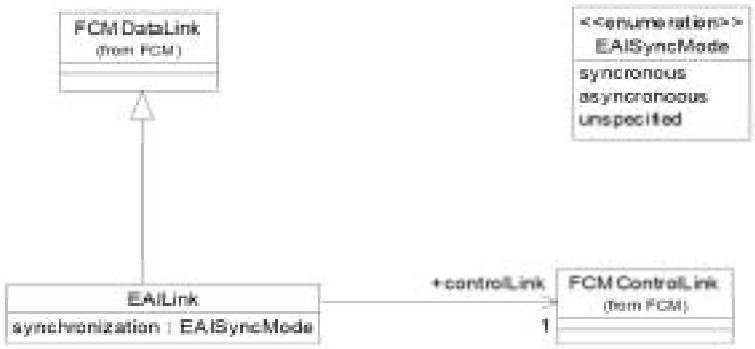
1. Added EAILink inheritance from FCMDataLink
2. 'data' association to FCMDataLink removed
3. 'control' association to FCMControlLink renamed to 'controlLink'

Figure 6-7 fragment (does not show EAISyncMode)

Before changes (fragment)



After Changes (includes EAISyncMode as updated for issue 5367)



Update to text of 6.3.2 'Definition', paragraph 2

Before

> As such, these links represent the flow of both data and control. In the FCM, data and control links are separate, so we introduce EAILink, which consists of one of each.

After

> As such, these links represent the flow of both data and control. In the FCM, data and control links are separate, so we introduce EAILink. EAILink inherits from FCMDataLink (which is a terminal to terminal link), and has an association with a single FCMControlLink.

Update to 6.3.2 'Constraints'

Before

An instance of an EAILink between an output terminal and an input terminal implies that there is an FCMDataLink between the two terminals, and an FCMControlLink from the output terminal to the node that owns the input terminal

After

The source terminal of the EAILink is the same as the source terminal of its *controlLink*

context **EAILink** inv**:**

self.sourceTerminal = self.controlLink.sourceTerminal

The target terminal of the EAILink is part of the *interface* of the *targetNode* of the controlLink

context **EALink** inv**:**

self.controlLink.targetNode.interface->exists(t | t=self.targetTerminal)

An EAILink connects two EAITerminals;

context **EAILink** inv:

inv: **self.sourceTerminal.oclIsKindOf(EAITerminal)**

inv: **self.targetTerminal.oclIsKindOf(EAITerminal)**

Disposition: Resolved

OMG Issue No: 4869

Title: Unnamed derived association between an EAITerminal and a source or sink

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.3 (EAITerminal) Description: The last constraint in this section states that "An EAITerminal on the exterior of a node constructed from an FCMComposition has a derived association with a single source or sink." Since it is not named, it is unclear to what "derived association" this statement refers, if the association even exists in the metamodel. Recommendation: The constraint needs to be made explicit.

Perhaps the "interface" association from FCMNode to FCMTerminal (which is a derived association in Figure 6-2) is meant. In this case, however, note that the constraint cannot be written as a constraint on FCMTerminal, since this association is not navigable from FCMTerminal back to FCMNode. I think the constraint would have to be on FCMComposition (or, better, on EAIMessageFlow).

Resolution:

The FCM in the EDOC submission is now updated to include an association between terminal an FCMSource, FCMTerminal and FCMSink.

Revised text:

Remove the following sentence from the text of 6.3.3;
"An EAITerminal on the exterior of a node constructed from an FCMComposition has a derived association with a single source or sink."

Disposition: Resolved

OMG Issue No: 4872

Title: Inconsistencies between text and diagram for EAIMessageContent

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.4 (EAIMessageContent)

Description:
The text under Constraints in Section 6.3.4 refers to "messageFormat" and "messageDomain", neither of which appear in the diagram in Figure 6-9. (By the way, the statement under Constraints does not really seem to be a constraint, but more a part of the description of the semantics.)

Resolution:

Perhaps "messageFormat" should be replaced by "languageElement", but it is not entirely clear that this is what is intended. If something else is intended, then it should be made explicit.

I think that "messageDomain" is should be simply "domain" (an attribute of EAIMessageContent).

The whole statement should be moved to be part of the description of the semantics of EAIMessageContent.

Revised Text:

Answered by resolution of Issue 5371

Disposition: **Resolved**

OMG Issue No: 4874

Title: The specifiedReplyToTerminal" and "specifiedExceptionTarget" associations

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.4 (EAIMessageContent)

Description:
The derived associations for "specifiedReplyToTerminal" and "specifiedExceptionTarget" are included in the metamodel to capture the "requirement that EAIHeader should specify the information required to locate replyTo and exceptionTarget terminals". However, this is dynamic instance state information that form part of the semantics of the EAIHeader, not its specification. It is thus not appropriate in the metamodel, since this results in corresponding elements in the DTD, even though it is not the intent that this information ever be provided in the

SPECIFICATION of an EAIHeader (which is, after all, what is being modeled and interchanged in the XMI).

Resolution:

The metamodel is a syntactic model, not a semantic model. The "specifiedReplyToTerminal" and "specifiedExceptionTarget" associations should be removed.

Revised Text:

Remove the following paragraph, which is the last paragraph of Section 6.3.4.2, "EAI Header":

> The requirement that EAIHeader should specify the information required to locate replyTo and exceptionTarget terminals is recorded via derived associations with EAITerminal. These derived associations do not form part of the message itself.

Remove the "specifiedReplyToTerminal" and "specifiedExceptionTarget" associations from the metamodel shown in Figure 6-10.

Disposition: Resolved

OMG Issue No: 4875

Title: XML Message Elements

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.4 (EAIMessageContent) Description: Is the intent here to actually change the OMG "XMI Production of XML Schema" specification? Because the generalizations shown in Figure 6-12 do actually change the specification of elements from the XML schema model. And, in any case, why are XMLMessageElements described in Section 6.3.4, rather than in Chapter 7, on the Common Application Metamodel?

Recommendation: Introduce a simple model in the CAM that has new classes for TDLangXSDType, TDLangXSDComplexType and TDLangXSDElementType that multiply inherit from the XML schema and TDLang classes. (Some further study is required of the XML schema model to confirm that this approach will actually work.)

Discussion:

It was agreed that this section was not intended to be normative, and that it should be removed to remove any potential for conflict.

Revised Text

<remove section 6.3.4 XML Message Elements>

Disposition: **Resolved**

OMG Issue No: 4876

Title: The constraint on the parameters of an EAIMessageOperation

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The text under Constraints in Section 6.3.5 states that a parameter of an EAIMessageOperation must have "a 1:1 relationship with EAIMessageContent or a subclass of EAIMessageContent." I assume that this is intended to mean a "1..1" assocation, since a "1:1" (usually read "1 to 1") relationship would require EVERY EAIMessageContent instance to be associated with some EAIParameter, as well as the converse, that every EAIParameter is associated with some EAIMessageContent instance (which, I think, is all that is intended). However, in this case, the constraint is already covered by the 1..1 multiplicity shown on the "message" association from EAIParameter to EAIMessageContent in Figure 6-9, and no other constraint on the association is needed.

Resolution:

The constraint should read simply "Every input and output of an EAIMessageOperation is an EAIParameter." The corresponding OCL is:

self.inputs->union(self.outputs)->forAll(oclIsType(EAIParameter))

Revised Text:

As above

Disposition: **Resolved**

OMG Issue No: 4877

Title: The lack of mention of "faults" for EAIMessageOperations

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.5 (EAIMessageOperation):
EAIMessageOperations are required to have EAIParameters for inputs and outputs, but no constraint is put on the faults for these operations (faults are part of the FCM

specification of FCMOperations, see Figure 6-1). Are EAIMessageOperations allowed to have faults? If so, are they required to be EAIParameters?

Resolution:

Either include a constraint prohibiting an EAIMessageOperation to have faults, or include them in the statement of the constraint requiring the other parameters to be EAIParameters.

Revised Text:

EAIMessageOperation

Description

EAIMessageOperation is a subclass of FCMOperation used to describe operations for which all the inputs and outputs are messages.

Constraints

Every input and output of an EAIMessageOperation is an EAIParameter. EAIMessageOperation may have zero or one faults. If present, the fault must be an EAIParameter.

context EAIMessageOperation

inv: self.inputs->union(self.outputs)->forAll(oclIsType(EAIParameter))

inv: self.inputs->union(self.faults)->forAll(oclIsType(EAIParameter))

inv: self.inputs->size() <= 1

Disposition: Resolved

OMG Issue No: 4878

Title: The "messages" associated with an EAIQueue

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.7 (EAIQueue)

Description:
The description of EAIQueue states that "EAIQueue has an ordered collection 'messages' of EAIMessageContent". This is stated as if it is describing an association in the

metamodel, but no such association appears in the diagram in Figure 6-15. The name "messages" is also used in the OCL statement under Constraints.

Including such a "messages: association would not be appropriate in the metamodel, however, since this represents the dynamic state of the queue, not a part of the specification of the queue. The statement that an EAIQueue has an ordered collection of messages, and the constraint on its state, is part of the description of the semantics of EAIQueue, not part of the definition of the metamodel.

Resolution:

Remove the reference to a "messages" association and remove the constraint. Instead, discuss the behavior of an EAIQueue as part of the description of its semantics.

Revised Text:

Before:

Description

EAIQueue is a queue of finite length, and is modeled as a subclass of EAIResource.

EAIQueue has an ordered collection *messages* of EAIMessageContent. A queue has a name, and the maximum number of messages it can hold is specified by maxLength.

EAIQueue is intended to be an abstraction of queuing infrastructure. We note that most MOM implementations allow machine-to-machine communication via a remote queuing infrastructure that can specify a number of different queue types and relationships between then. This can be modeled as refinement or realization of EAIQueue or (see Section 6.4.1.2) of the EAIPrimitiveOperator *EAIStream*.

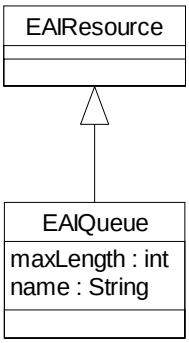


Figure 5 EAIQueue

Constraints

maxLength >= messages->size()

After:

Description

EAIQueue is a queue of finite or unbounded length, and is modeled as a subclass of EAIResource.

EAIQueue has a name, and a Boolean "isBound" showing if the queue length is finite or unbounded. EAIQueue also has a maxLength which specifies the maximum number of messages it can hold.

EAIQueue is restricted to hold specific type of message contents, if an EAIMessageContent is specificed for EAIQueue. Otherwise, EAIQueue can hold any type of message contents.

EAIQueue is intended to be an abstraction of queuing infrastructure. We note that most MOM implementations allow machine-to-machine communication via a remote queuing infrastructure that can specify a number of different queue types and relationships between then. This can be modeled as refinement or realization of EAIQueue or (see Section 6.4.1.2) of the EAIPrimitiveOperator *EAIStream*.

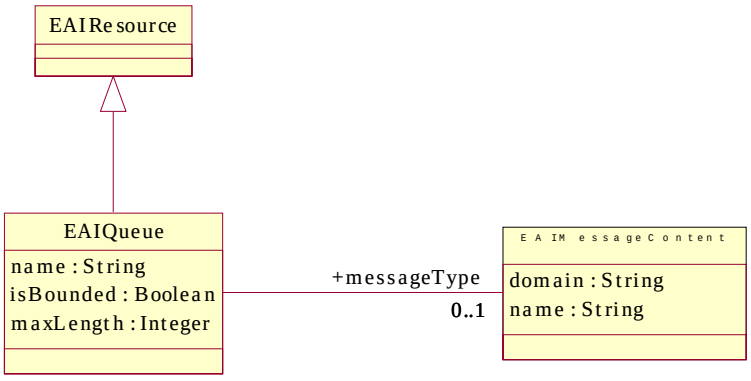


Figure 5 EAIQueue

Disposition: Resolved

OMG Issue No: 4879

Title: There is no way to specify typed EAIQueues

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.7 (EAIQueue)

Description:
As defined in Section 6.3.7, any EAIQueue can contain any kind of EAIMessageContent. There is no way to specify that an EAIQueue is restricted to contain only certain kinds of messages.

Resolution:

Add an optional "messageType" association from EAIQueue to EAIMessageContent. If a specific type of EAIMessageContent is specified for an EAIQueue, then the semantics is that the queue is restricted to only contain that kind of message content.

Revised Text:

See the resolution of issue 4878.

Disposition: Resolved

OMG Issue No: 4880

Title: There is no way to specify unbounded EAIQueues

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.7 (EAIQueue)

Description:
As defined in Section 6.3.7, every EAIQueue must have a maxLength. There is no way to specify an "unbounded" queue.

Resolution:

Add an attribute "isBounded: Boolean" to EAIQueue. The semantics are that, if isBounded is true, then the EAIQueue can hold only up to maxLength messages, otherwise the length of the queue is not bounded.

(Also, the type of maxLength should be "Integer", not the language-specific "int".)

Revised Text:

See the resolution of issue 4878.

Disposition: **Resolved**

OMG Issue No: 4881

Title: Inconsistent statements on the use of queued terminals

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section 6.3.8 states that "We represent the fact that an *Operator* uses queuing via the use of" queued terminals (emphasis added). Later, however, it is stated that "Queued input and output terminals may be used on _any of the EAI constructs that have terminals_", which is more expansive than the previous statement.

Resolution:

Since the concept of "operators" has not yet even been introduced in the linear sequence of the document at this point, it would be better if the initial statement was worded something like: "EAIQueuedInputTerminal and EAIQueuedOutputTerminal are subclasses of EAITerminal that are used to represent message communication that occurs via queuing."

Revised Text:

In section 6.3.8
replace:

> We represent the fact that an Operator uses queueing via the use of EAIQueuedInputTerminal and EAIQueuedOutputTerminal, which are subclasses of EAITerminal.

with:

> EAIQueuedInputTerminal and EAIQueuedOutputTerminal are subclasses of EAITerminal that are used to represent message communication that occurs via queuing

Disposition: **Resolved**

OMG Issue No: 4882

Title: Poor wording in the discussion of EAIQueue

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.8 (EAIQueuedInputTerminal and EAIQueuedOutputTerminal)

Description:
The discussion in Section 6.3.8 says that "...an EAIQueuedOutputTerminal has an association with each of the queues _used by its target EAIQueuedInputTerminals_" (emphasis added). This is poorly worded, since links have targets, not terminals.

There is a constraint that states that "All EAILinks from an EAIQueuedOutputTerminal must _be instances of_ EAIQueuedInputTerminal". This is also poorly worded, since an EAILink cannot be "an instance of" an EAIQueuedInputTerminal.

Resolution:

Change the wording of the first item to "...an EAIQueuedOutputTerminal has an association with each of the inputQueues of the EAIQueuedInputTerminals to which it is linked by EAILinks."

Change the wording of the constraint to "All EAILinks with an EAIQueueOutputTerminal as the sourceTerminal must have an EAIQueuedInputTerminal as the targetTerminal."

Revised Text:

Before:
1)

> An EAIQueuedInputTerminal has an association with the single queue that it reads from, while an EAIQueuedOutputTerminal has an association with each of the inputQueues of the EAIQueuedInputTerminals to which it is linked by EAILinks.

2)

> All EAILinks from an EAIQueuedOutputTerminal must be instances of EAIQueuedInputTerminal.

After:
1)

> An EAIQueuedInputTerminal has an association with the single queue that it reads from, while an EAIQueuedOutputTerminal has an association with each of the queues used by its target EAIQueuedInputTerminals.

2)

> All EAILinks with an EAIQueuedOutputTerminal as the sourceTerminal must have an EAIQueuedInputTerminal as the targetTerminal.

Disposition: **Resolved**

OMG Issue No: 4883

Title: Poor wording on the use of queued terminals with queued sources and links

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The last paragraph under Description in Section 6.3.9 states "Note that EAIQueuedSink and EAIQueuedSource could themselves be specialized to use queued terminals." There is no need to "specialize" these metaclasses in order to use queued terminals -- such terminals can simply be attached to them if desired, since queued terminals are kinds of regular EAITerminals.

Resolution:

Change the quoted sentence to: "Note that the terminals of EAIQueuedSink and EAIQueuedSource (used within the EAIMessageFlow) could themselves be queued terminals." Keep the following sentence unchanged.

Revised Text:

In section 6.3.9
replace:

> Note that EAIQueuedSink and EAIQueuedSource could themselves be specialized to use queued terminals

with

> Note that the terminals of EAIQueuedSink and EAIQueuedSource (used within the EAIMessageFlow) could themselves be queued terminal

Disposition: **Resolved**

OMG Issue No: 4892

Title: Derivation of the "defines" association for EAIPrimitiveOperator

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.1 (EAIPrimitiveOperator) Description: Under Constraints in Section 6.3.10.1 it states that "When used in an EAIMessageFlow, an EAIPrimitiveOperator also 'defines' a type." Since this is listed as a constraint, and since the "defines" association is shown as <<derived>> in Figure 6-18, one would assume that the constraint is intended to give the derivation of the association. But it certainly does not state that clearly, nor do I see any way, in the underlying FCM, for an EAIPrimitiveOperator (as an FCMFunction) to define an FCMType in the context of an EAIMessageFlow (as an FCMComposition).

Recommendation: Either make the association not derived (in which case the constraint statement needs to be expanded into a statement of the semantics of what it means for a primitive operator to "define a type") or make the derivation constraint much more explicit (i.e., provide the OCL).

Discussion:

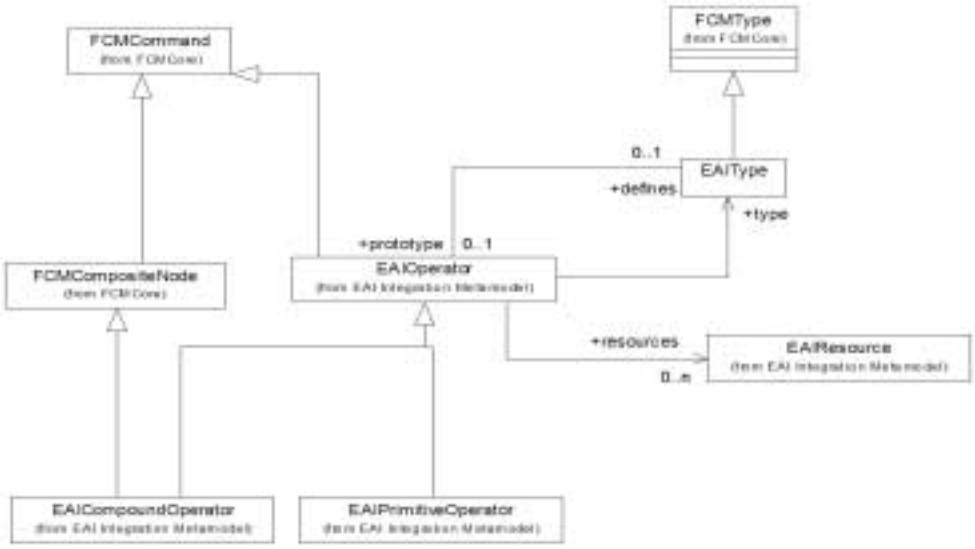
The 'defines' relationship was erroneously labelled as derived in the final adopted submission.

An EAIOperator may 'define a type', in effect by defining a 'prototype'. All instances of the EAIType are identical copies of the prototype, in terms of their properties but not the connectivity of their terminals. This allows the EAI model to use a simple 'template' scheme where multiple instances of essentially the same operator are required.

The resolution to this issue is also affected by changes to the Flow Composition Model in the EDOC Finalisation Taskforce. This has introduced the 'FCMCompositeNode', which is used as the base for nodes defined from compositions of other nodes. The EAIOperator class has been introduced to allow the common characteristics fo EAI operators to be defined.

Disposition: Accepted

Revised text
Replace diagram 6-18 with the following;



Introduce the following section defining EAIOperator;

Operator

Operators act upon messages as they flow between systems. We define EAIOperator to be a subclass of FCMCommand. EAIOperators have a type, EAIType. An EAIOperator prototype can also be used to specify an EAIType. EAIOperator may optionally specify EAIResources that it uses to enact its function.

Constraints

context EAIOperator

Define what it means to be a prototype

let isPrototype = self.defines->size() = 1

let isInstance = self.defines->isEmpty()

An EAIOperator has the same number of terminals as its prototype;

inv: if isInstance then self.interface->size() = self.type.protoype.interface->size()

The prototype for a prototype is itself;

inv: if isPrototype then self.type.prototype = self

All of the terminals of an EAIOperator are EAITerminals;

inv: self.interface->forall(t | t.oclIsKindOf(EAITerminal))

An EAIOperator's terminals have the same names as its prototype;

inv: if isInstance then self.interface->

forall(t | self.type.prototype.interface->exists(tt| tt.name=t.name))

An EAIOperator has the same set of resources as its prototype;

inv: if isInstance then self.resources = self.type.prototype.resources

The 'invokes' association EAIOperation inherited from FCMFunction must be to an EAIMessageOperation;

inv: self.invokes.oclIsKindOf(EAMessageOperation)

Revised text for section xx.xx EAIPrimitiveOperator

PrimitiveOperator

Description

Instances of EAIPrimitiveOperator enact a simple message processing operation. EAIPrimitiveOperator is a subclass of EAIOperator

Constraints

Inherited from EAIOperator.

OMG Issue No: 4893

Title: Missing constraint on the FCMOperation invoked by an EAICompoundOperator

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.2 (EAICompoundOperator) Description: Section 6.3.10.1 includes a constraint that requires the FCMOperation invoked by an EAIPrimitiveOperator to be an EAIMessageOperation. An EAICompoundOperator is a kind of FCMCommand, which is a kind of FCMFunction, and, therefore, it also has an invoked FCMOperation. But there is not constraint on this operation in Section 6.3.10.2. Further, this operation is important, because it would seem to be the operation whose parameters provide the basis for the terminals of the EAICompoundOperator and, therefore, for the terminals of EAISources and EAISinks in the implementingComposition of the operator.

Recommendation: Add the following constraints to EAICompoundOperator.

An EAICompoundOperator invokes an EAIMessageOperation.

self.invokes->oclIsKindOf(EAIMessageOperation)

All EAISources in the implementingComposition of an EAICompoundOperator must implement the EAIMessageOperation invoked by the EAICompoundOperator.

self.implementingComposition.nodes-> select(oclIsKindOf(EAISource)) ->forAll(implements = self.invokes)

Discussion:

The resolution to this issue is covered by the resolution to issues 4863 and 4892

Disposition: **Accepted**

Revised Text

See revised text for issues 4863 and 4892

OMG Issue No: 4894

Title: The usefulness of EAIMessageFlow in itself

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The concept of an EAIMessageFlow is useful in itself, separately from its use in defining a compound operator. (For example, we are using it as the basis for the deployment of pieces of our message broker configuration to different platforms.) However, the fact that it is defined within Section 6.3.10.2 (EAICompoundOperator) makes it seem like it is intended just for defining compound operators.

Recommendation:
Move the definition of EAIMessageFlow out of Section 6.3.10 into its own subsection of Section 6.3. (Moving it to before the discussion of sources and sinks would also make those discussions clearer, though there would then instead be a forward reference to operators in the constraint on the contents of EAIMessageFlows.)

Resolution:

EAIMessageFlow is placed into a peer section at the same level as EAICompoundOperator under 6.3.10 EAIOperator and after 6.3.10.5 EAICompoundOperator. So it becomes 6.3.10.6 (though other changes may affect the precise numbering) numbering)

Revised Text:

6.3.10.6 EAIMessageFlow

An EAIMessageFlow is a subclass of FCMComposition. Each of its nodes (see Figure ? on page ?) must be one of the operator classes (EAIPrimitiveOperator or EAICompoundOperator), and its connections must be EAILinks. In addition it allows nodes to have explanatory annotations attached to them.

Disposition: **Resolved**

OMG Issue No: 4896

Title: Wording error in Section 6.3.10.2.2

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.2.2 ('Exposing' terminals in an EAIMessageFlow):
The first sentence after the bullets in Section 6.3.10.1 reads "This consequently determines the type _of that the_ external EAITerminal represents" (emphasis added).

Recommendation:
Change "...of that the..." to "...that the..."

Resolution:

Accept the issue precisely as worded (note that this now refers to section 6.3.10.7)

Revised Text:

Text before:

> This consequently determines the type of that the external EAITerminal represents"

Text after:

> This consequently determines the type that the external EAITerminal represents"

Disposition: **Resolved**

OMG Issue No: 4897

Title: Missing derivation of the "promotedTerminal" association

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.2.2 ('Exposing' terminals in an EAIMessageFlow) Description: The meaning of the derived "promotedTerminal" association is discussed in Section 6.3.10.2.2, but no clear derivation constraint is given.

Recommendation: Given the FCM metmamodel (see Figure 6-2), the appropriate constraint is not so easy to formulate. I think the following will do, as a constraint on EAICompoundOperator (NOT EAITerminal).

> The output terminal of each EAISource in the implementingComposition of an EAICompoundOperator is promoted to the input terminal of the EAICompoundOperator that represents the same EAIParameter (of the FCMOperation implemented by the EAISource) as the output terminal of the EAISource.
>
> let sourceTerminals = self.implementingComposition.nodes-> select(oclIsType(EAISource)).interface in self.interface->select(terminalKind = #in) ->includesAll(sourceTerminals.promotedTerminal) and sourceTerminals-> forAll(promotedTerminal->notEmpty() and parameter = promotedTerminal.parameter)

(This assumes that there is already a constraint requiring an EAISource to have output terminals that represent the input parameters of the operation it implements.)

Discussion:

The relationship between the 'interior' of an FCMCompositeOperator (the FCMComposition that defines it) and its exterior (the terminals of the FCMCompositeOperator) are defined by updates to the FCM in the EDOC submission. While 'promoted terminals' are not discussed, the concept is not required elsewhere within the either the EAI or EDOC standards.

Recommend that we remove the section.

Revised Text

Remove section 6.3.10.2.2

Disposition: Resolved

OMG Issue No: 4898

Title: Missing discussion of promotedTerminals for EAISinks

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.10.2.2 ('Exposing' terminals in an EAIMessageFlow): The first paragraph of Section 6.3.10.2.2 discusses both EAISources and EAISinks, but the remainder seems to only cover EAISources. It would seem that there also need to be promoted terminals for EAISinks, but with the roles of input and output reversed. Recommendation: Extend the

definition of "promotedTerminal" to include the input terminal of an EAISink being promoted to an output terminal of an EAIMessageFlow. There needs to be a similar constraint on EAICompoundOperators to define this association for terminals of EAISinks as for EAISources.

Discussion:

See discussion for 4897

Revised Text

Remove Section 6.3.10.2.2

Disposition: **Resolved**

OMG Issue No: 4947

Title: Missing constraints on the input terminal of an EAITargetAdapter *(Note that this issue is misnamed. It should be:* Missing name attribute for EAITerminal*)*

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.3 (EAITerminal)

Description:
In subsequent sections on adapters and operators, references are made to the "name" of a terminal. However, an FCMTerminal does not have a name (see Figure 6-2) and no name attribute is given for EAITerminal in Section 6.3.3 (see Figure 6-8). (Actually, a name attribute is shown for EAITerminal in some later diagrams, such as Figure 6-16 and Figures 6-20 and later.)

Resolution:

Conceivably, the name of an FCMTerminal could be given by the name of languageElement of the FCMParameter represented by the FCMTerminal (via the association given in Section 6-6), and this could be defined as a derived attribute of FCMTerminal. However, it is not clear that this makes sense for an EAITerminal, because the status of the languageElement associated with an EAIParameter is questionable (see previous issue). Therefore, it seems simpler to have an explicit name attribute for EAITerminals and leave FCMTerminals unnamed (unless the FCM is separately updated to add names to FCMTerminals).

Revised Text:

Add the attribute "name: String" to the class EAITerminal in the metamodel shown in Figure 6-8 and 6-20. (Note that, despite the comment in the "Description" above, Figure 6-20 does *not* show a "name" attribute for EAITerminal. However, all other figures besides 6-8 and 6-20 seem to.)

Disposition: Resolved

OMG Issue No: 4948

Title: Lack of generalization for message content

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.4 (EAIMessageContent)

Description:
The metamodel has no way to model a generalization/specialization relationship between message content. This is a serious limitation in being able to model general message flows (at least for a number of the things we want to do). Note that the ability to specify generalization/specialization for TDLangElements (which is presumably available, depending on the is not sufficient (and would only seem to be available in language-specific metamodels anyway). The desire is to be able to specify a general message flow that can carry any of a number of potentially differently formatted message contents that all specialize a common message-content specification.

Resolution:

Add to the EAIMessageContent metamodel (Figure 6-9) a generalization/specialization association from EAIMessageContent to itself.

Revised Text:

Add the following paragraph to the end of the "Description" part of Section 6.3.4, "EAIMessageContent":

> One kind of message content may also be defined as a specialization of another, more general, kind of message content. This is represented by the (optional) generalization association from EAIMessageContent to itself. An EAIMessageContent with a generalization is considered to "inherit" all the parts specified for the generalization. It may also specify additional parts of its own, which are in addition to those inherited from the generalization.

On the metamodel shown in Figure 6-9, *add* a unidirectional association from EAIMessageContent to itself, with the opposite-end rolename being "generalization".

Disposition: Resolved

OMG Issue No: 4949

Title: Missing constraint on the output terminal of an EAISourceAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.1 (EAISourceAdapter)

Description:

An EAITargetAdapter (Section 6.3.11.2) is limited to have a single input terminal, but Section 6.3.11.1 does not have a corresponding constraint for EAISourceAdapter. It seemingly allows an EAISourceAdapter to have many output terminals (or even none). This is also inconsistent with the discussion of source adapters under Collaboration Modeling (Section 8.3.6).

Resolution:

Replace the first constraint in Section 6.3.11.1 with: "An EAISourceAdapter has a single output terminal, which is an EAITerminal with the name 'out'."

Revised Text:

Replace the first constraint in Section 6.3.11.1:

> The output terminals of a SourceAdapter are instances of EAITerminal

with:

> An EAISourceAdapter has a single output terminal, which is an EAITerminal with the name “out”.

Disposition: **Resolved**

OMG Issue No: 4950

Title: Missing constraints on the input terminal of an EAITargetAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.2 (EAITargetAdapter)

Description:

Section 6.3.11.2 states that "An EAITargetAdapter has a single input EAITerminal ("in")." However, this constraint is not listed under the Constraints heading in the section.

Resolution:

Add the constraint: "An EAITargetAdapter has a single input terminal, which is an EAITerminal with the name 'in'."

Revised Text:

As recommended, add the following as the first constraint in Section 6.3.11.2:

An EAITargetAdapter has a single input terminal, which is an EAITerminal with the name “in”.

Disposition: **Resolved**

OMG Issue No: 4951

Title: The name "EAITargetAdapter"

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.2 (EAITargetAdapter)

Description:
The name "EAITargetAdapter" is not consistent with the source/sink pairing always used elsewhere.

Resolution:

Change the name to "EAISinkAdapter".

Revised Text:

Change "EAITargetAdapter" to "EAISinkAdapter" and the general term "target adapter" to "sink adapter" throughout the document. *Change* the stereotype name "TargetAdapter" to "SinkAdapter" in Chapter 8.

Disposition: **Resolved**

OMG Issue No: 4952

Title: Misplaced constraints on the terminals of an EAICallAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.3 (EAICallAdapter)

Description:
The constraints on the terminals of an EAICallAdapter in Section 6.3.11.3 are not listed under the Constraints heading.

Resolution:

List the constraints under the Constraints heading.

Revised Text:

Add the following paragraphs at the beginning of the "Constraints" part of Section 6.3.11.3:

> An EAICallAdapter has two input terminals, one of which is an FCMTerminal that is *not* an EAITerminal and the other of which is an EAITerminal with the name "handleReply".
>
> An EAICallAdapter has two output terminals, one of which is an FCMTerminal that is *not* and EAITerminal and the other of which is an EAITerminal with the name "request".

(Note that FCMTerminals that are not EAITerminals cannot have names, so it is not possible to require the names "call" and "out" for them.)

Disposition: **Resolved**

OMG Issue No: 4953

Title: The use of FCMTerminals on an EAICallAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.3 (EAICallAdapter)

Description:
In Section 6.3.11.3, an EAICallAdapter is defined as a specialization of FCMFunction (see Figure 6-23). Now, an FCMFunction invokes a single FCMOperation (see Figure 6-2). The terminals of an FCMFunction represent the FCMParameters of the FCMOperation (see Figure 6-6 and discussion in Section 6.2.5), with input terminals representing input parameters and output terminals representing output terminals (one assumes). However, the semantics of an EAICallAdapter are not properly reflected by the invocation of a single operation with the signature (input call, input handleReply, output request, output out). Rather, the semantics of an EAICallAdapter are to invoke the callToRequestMapping, wait for a reply and then call the replyToOutputMapping. Nevertheless, it is necessary to have EAIParameters that the request and handleReply terminals can represent, since that is the only way that message content typing can be provided for those terminals.

Resolution:

Define the invoked operation for an EAICallAdapter as having the FCMParameters corresponding to the call and out FCMTerminals (this reflects that, from the point of view of the caller, the semantics of an EAICallAdapter is that of an operation with "call" as its input and "out" as its output). Add two associations from EAICallAdapter to EAIParameter (say, "requestParameter" and "replyParameter") and define derivation rules such that these are represented by the "request" and "handleReply" terminals (since the representation association is a derived association -- see Figure 6-6).

Revised Text:

Replace the last paragraph of the “Description” part of Section 6.3.11.3:

> When invoked via its "call" terminal, the EAICallAdapter maps the call parameters into a request message and sends it to the input terminal of an EAIRequestReplyAdapter. It waits for a reply. On receipt of a reply it maps the message as specified in the replyTo OutputMapping, and puts out the result on the "out" terminal.

with:

> From the point of view of the requesting application, the EAICallAdapter is a single FCMFunction that takes an input on its “call” terminal and produces an output on its “out” terminal. Within the EAI model, this function is realized as follows (i.e., this is effectively the behavior of the FCMOperation invoked by the function).

1. Map the “call” input into a request message using the callToRequestMapping.
2. Place the request message on the “request” output terminal.
3. Wait for a reply message to be received on the “handleReply” input terminal.
4. Map the reply message to an output value using the replyToOutput mapping.
5. Place the output value on the “out” terminal.

Add the following paragraphs to the “Constraints” part:

> The FCMOperation invoked by an EAICallAdapter (when considered as an FCMFunction, see Figure 6-2) must have exactly one input FCMParameter and exactly one output FCMParameter.
>
> The input FCMTerminal of an EAICallAdapter (that is *not* and EAITerminal) is associated with the input FCMParameter and the output FCMTerminal (that is *not* and EAITerminal) is associated with the output FCMParameter.
>
> The representation of the requestParameter of an EAICallAdapter is the “request” terminal and the representation of the replyParameter of an EAICallAdapter is the “handleReply” terminal. (The representation association for an FCMParameter is shown on Figure 6-6.)

Add to the metamodel in Figure 6-23 two unidirectional associations from EAICallAdapter to EAIParameter, with the opposite-end rollnames “requestParameter” and “replyParameter”.

Disposition: Resolved

OMG Issue No: 4954

Title: Overconstraint on allowed connection to "request" terminal of EAICallAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.3 (EAICallAdapter)

Description:
Section 6.3.11.3 states that an EAICallAdapter sends its requests "...to the input terminal of an EAIRequestReplyAdapter." More stringently, the section includes the constraint that "The 'out' terminal of [an] EAICallAdapter must be connected via an EAILink to the 'requestIn' terminal of an EAIRequestReplyAdapter." This means that the requests

coming out of an EAICallAdapter cannot be routed through any operators, but must flow DIRECTLY to the input terminal of an EAIRequestReplyAdapter. This seems to seriously limit the usefulness of splitting call and request/reply adapters at all, and it certainly prevents a core capability that we need of being able to route request messages just like any other messages. Obviously, at the end of a message flow, a request message generally needs to flow into a EAIRequestReplyAdapter, but there is no reason the connection has to be DIRECT.

Resolution:

Eliminate the constraint.

Revised Text:

Remove both the following constraints from Section 6.3.11.3:

> The "out" terminal of EAICallAdapter must be connected via an EAILink to the "requestIn" terminal of an EAIRequestReplyAdapter.
>
> The "handleReply" terminal of EAICallAdapter is the target of connections via an EAILink from the "replyOut" terminal of an EAIRequestReplyAdapter.

Replace them with:

> The parameter associated with the “out” terminal of an EAICallAdapter must be an EAIParameter with a message that is an EAIRequestFormat.

(Note that appropriate changes to the “Description” text are already covered in the resolution to Issue 4953.)

Disposition: Resolved

OMG Issue No: 4955

Title: specifiedReplyTerminal association of EAIRequestFormat is dynamic state data

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.3.1 (EAIRequestFormat)

Description:
Section 6.3.11.3.1 defines a "specifiedReplyTerminal" derived association (Figure 6-24) for an EAIRequestFormat that "...specifies a terminal to which replies should be sent."

However, this information is not part of the specification of a request message, but it is part of the dynamic state of a request message, since it cannot be determined until that request message is actually created. As a metaclass, an instance of EAIRequestFormat is NOT a request message, but is rather a SPECIFICATION of a request message, and therefore should not include the state of the message itself.

Resolution:

Include the discussion of the identification of reply terminals as part of the semantics of an EAIRequestFormat, not the syntax. Remove the "specifiedReplyTerminal".

Revised Text:

(Note that the correct section in the Final Adopted Specification is 6.3.11.4, not 6.3.11.3.1.)

Replace the text in Section 6.3.11.4:

> EAIRequestFormat is a subclass of EAIMessageContent. A message that conforms to EAIRequestFormat specifies a terminal to which replies should be sent (specifiedReplyTerminal). The association with the terminal is not explicit in the message but may be computed from information in the message.

with:

> EAIRequestFormat is a subclass of EAIMessageContent that is used to specify a request message that may be produced by an EAICallAdapter and received by an EAIRequestReplyAdapter. While the structure of an EAIRequestFormat is just like any other EAIMessageContent, it a request message has the added semantic responsibility of identifying the terminal to which a reply to the message should be sent. How this identification is made is not explicitly defined in the metamodel syntax for an EAIRequestFormant, but it must be computable from the information specified for a request message (e.g., some sort of unique identifier for a reply terminal might be included in a header part of the message or some other sort of language element might be modeled to provided a logical identification of a terminal).

Remove the “specifiedReplyTerminal” association from the metamodel shown in Figure 6-24.

Disposition: Resolved

OMG Issue No: 4956

Title: Missing constraint on the terminals of an EAIRequestReplyAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.4 (EAIRequestReplyAdapter)

Description:
Section 6.3.11.4 states that an EAIRequestReplyAdapter "has a single input terminal 'requestIn' and a single output terminal 'replyOut'", but this constraint is not listed under the Constraints heading in the section.

Resolution:

List an appropriate constraint under the Constraints heading. This constraint should also require that the terminals be EAITerminals.

Revised Text:

The resolution of this issue is covered by the resolution to Issue 4957 below.

Disposition: Resolved

OMG Issue No: 4957

Title: Lack of FCMTerminals for an EAIRequestReplyAdapter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.3.11.4 (EAIRequestReplyAdapter)

Description:

In Section 6.3.11.3, an EAICallAdapter is defined to have FCMTerminals that represent the call from an external system into a message flow. However, Section 6.3.11.4 defines an EAIRequestReplyAdapter to have ONLY the "requestIn" and "replyOut" EAITerminals, without any other FCMTerminals to allow connection to the external system. This is not parallel with the definition of EAICallAdapter, and, indeed, it is inconsistent with the definitions of other adapters, which allow FCMTerminals for external connection (for example, the definition of an EAISourceAdapter places constraints on the output terminal of the adapter, which connects into the message flow, but specifically does not constrain the input terminal(s), to allow external connection).

Resolution:

Define an EAICallAdapter to have two additional FCMTerminals, "callOut" and "handleReturn". Define the "requestIn" and "replyOut" terminals to represent the EAIParameters of the EAIOperation invoked by the EAIRequestReplyAdapter. Add two new assocations from EAIRequestReplyAdapter to FCMParameter (say "callParameter" and "returnParameter") and define the "callOut" and "handleReturn" terminals as representing these parameters (this provides the typing for the new terminals).

Revised Text:

Replace the following text at the beginning of Section 6.3.11.5 (which is the correct section in the Final Adopted Specification, not 6.3.11.4):

> An EAIRequestReplyAdapter is a subclass of FCMCommand. It has a single input terminal "requestIn" and a single output terminal "replyOut".
>
> On receipt of a message that conforms to the EAIRequestFormat, it maps the request message into the format required by the system it interfaces to, calls an operation on that system, synchronously receives a result, and formats the result for return to the "handleReply" terminal specified in the request message.

with:

> An EAIRequestReplyAdapter is used to synchronously invoke a function of a server application. It has two input terminals:
>
> - “requestIn”: an EAITerminal that accepts a message whose content is specified by an EAIRequestFormat (and thus provides some means of identifying a reply terminal)
> - “handleReturn”: an FCMTerminal that receives the reply from the server application
>
> It has two output terminals:
>
> - “replyOut”: the EAITerminal from which the reply message is sent
> - “call”: an FCMTerminal to which the request is mapped to be sent to the server application
>
> The request reply adapter has two mappings, one of which specifies how the “requestIn” input data are mapped to the server application call; the other specifies how the return data are mapped to the output message represented by the “replyOut” terminal.
>
> From the point of view of the EAIModel, the EAIRequestReplyAdapter is a single FCMFunction that takes a request message on its “requestIn” terminal

and produces a reply message on its “replyOut” terminal. This function is realized as follows (i.e., this is effectively the behavior of the FCMOperation invoked by the function).

1. Map the “requestIn” message into the data required for the server application call using the requestToCallMapping.
2. Place the call data on the “call” output terminal.
3. Wait for return data to be received on the “handleReturn” input terminal.
4. Map the return data to a reply message using the returnToReply mapping.
5. Place the reply message on the “replyOut” terminal and also on the reply terminal identified in the request message.

Replace the following text from the “Constraints” part of Section 6.3.11.5:

The "requestIn" terminal expects to receive a message of that conforms to EAIRequestFormat.

with:

An EAIRequestReplyAdapter has two input terminals, one of which is an FCMTerminal that is *not* an EAITerminal and the other of which is an EAITerminal with the name “requestIn”.

An EAIRequestReplyAdapter has two output terminals, one of which is an FCMTerminal that is *not* and EAITerminal and the other of which is an EAITerminal with the name “replyOut”.

The FCMOperation invoked by an EAIRequestReplyAdapter (when considered as an FCMFunction, see Figure 6-2) must be an EAIOperation with exactly one input EAIParameter, with a message that is an EAIRequestFormat, and exactly one output EAIParameter.

The “requestIn” terminal of an EAIRequestReplyAdapter is associated with the input EAIParameter and the “replyOut” terminal is associated with the output EAIParameter.

The representation of the callParameter of an EAIRequestReplyAdapter is the output FCMTerminal and the representation of the returnParameter of an EAIRequestReplyAdapter is the “return” terminal. (The representation association for an FCMParameter is shown on Figure 6-6.)

Add to the metamodel in Figure 6-25 two unidirectional associations from EAIRequestReplyAdapter to EAIParameter, with the opposite-end rollnames “callParameter” and “returnParameter”.

Disposition: Resolved

OMG Issue No: 4958

Title: Missing multiplicity for the "filterCondition" of an EAIFilter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Figure 6-26 of Section 6.4.1.1 does not show any multiplicity for the "filterCondition" of an EAIFilter.

Resolution:

Show a multiplicity of "1..1".

Revised Text:

Add the multiplicity of “1” to the “filterCondition” association for EAIFilter shown in the metamodel in Figure 6-26.

Disposition: Resolved

OMG Issue No: 4965

Title: Multiplicity of the "transformation" association for an EAITransformer

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.4 (EAITransformer)

Description:
Figure 6-29 in Section 6.4.1.4 shows the "transformation" association of EAITransformer with FCMMapping as having a multiplicity of "0..n" [sic]. However, it is unclear what it means for a transformer to have more than one mapping (or to have zero mappings).

Resolution:

Make the multiplicity "1..1".

Revised Text:

Change the multiplicity of “0..n” to “1” on the “transformation” association for EAITransformer shown in the metamodel in Figure 6-29.

Disposition: **Resolved**

OMG Issue No: 4966

Title: Redundant "database" association for an EAIDBTransformer

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.5 (EAIDBTransformer)

Description:
According to Figure 6-18 in Section 6.3.10.1, any EAIPrimitiveOperator may already have any number of associated EAIResources. Since EAIDBTransformer is a descendant of EAIPrimitiveOperator (via EAITransformer), and EAIDatabase is a child of EAIResource, it is not necessary to have a specific additional association from EAIDBTransformer to EAIDatabase. In fact, having the specific association hides the fact that an EAIDatabase is really just in the role of one of the resources of the EAIDBTransformer operator (which may be important to a tool which is managing the general allocation of resources to operators).

Resolution:

Remove the "database" association from Figure 6-30. Instead, add a constraint that "An EAIDBTransformer has exactly one resource, which is an EAIDatabase", with corresponding OCL:

(self.resources->size() = 1) and (self.resources.oclIsKindOf(EAIDatabase))

Revised Text:

Before the following paragraph in Section 6.4.1.5:

> Access to a database as a resource allows the transformation to make use of information contained in the database. In particular, it allows the message to be augmented (or enriched) with data from the database.

add the following paragraph:

> An EAIDBTransformer is an EAITransformer, which is itself an EAIPrimitiveOperator, which may have resources attached to it (see Figure 618). An EAIDBTransformer is specifically required to have exactly one such resource, which must be an EAIDatabase.

At the end of Section 6.4.1.5 (after Figure 6-30), *add* the following constraint:

Constraints

An EAIDBTransformer has exactly one resource, which is an EAIDatabase.

Remove the "database" association from the metamodel shown in Figure 6-30.

Disposition: Resolved

OMG Issue No: 4967

Title: Inclusion of dynamic state in the metamodel for EAIAggregator

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.6 (EAIAggregator)

Description:

Figure 6-31 in Section 6.4.1.6 shows an unnamed association between EAIMessageAggregation and EAIMessageContent. However, this is part of the dynamic state of an EAI aggregator operator, not part of the specification of the operator. An instance of EAIMessageAggregator, with one or more EAIMessageAggregations is a SPECIFICATION of an EAI aggregator operator, not the operator itself, and therefore should not include the dynamic state of the operator.

Resolution:

Remove the association from Figure 6-31.

Revised Text:

Remove the association between EAIMessageAggregation and EAIMessageContent from the metamodel shown in Figure 6-31. (Also remove EAIMessageContent from the diagram but not, of course, from the model.)

Disposition: Resolved

OMG Issue No: 4968

Title: The specification of EAIRouter and EAITimer as compound operators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 6.4.1.7 (EAIRouter) and 6.4.1.10 (EAITimer): The specification of EAIRouter and EAITimer as compound operators does not correctly follow the FCM semantics for FCMCommands (which is the parent of EAICompoundOperator), for two reasons:

1. In order to promote the terminals of the primitive operators contained in the compound operator to terminals of the compound operator, there must be EAISources and EAISinks in contained in the compound operator corresponding to the external terminals. These are not specified in the discussion of the EAIRouter and EAITimer compounds.

2. An FCMCommand is a specialization of an FCMFunction, which invokes a single FCMOperation (see Figure 6-2). The content of the FCMCommand is simply a means to implement this operation. The expected FCM semantics are thus that, when all the inputs are provided, the operation is invoked, producing the specified outputs (via the internal composition, in the case of an FCMCommand). However, the semantics of EAIRouter and EAITimer, as described in Sections 6.4.1.7 and 6.4.1.10, are really to provide the functionality of the contained primitive operators as, effectively, multiple operations of the compound operator. This is not consistent with the FCM semantics, which implies a that an FCMCommand implements a single function from inputs to outputs.

Recommendation: Do not implement EAIRouter or EAITimer as compound operators. Instead, simply provide the component primitive operators, as appropriate. (See subsequent issues for more detailed recommendations.)

Proposed Resolution:

Remove Section 6.4.1.7 “EAIRouter”. See also the resolution to issue 4969. For EAITimer, see the resolution to issue 4976.

Disposition: Resolved

OMG Issue No: 4969

Title: Inclusion of the dynamic state "routingTargets" for the EAIRoutingTable

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.7.1 (EAIRouterUpdate and EAIBroadcaster): From the point of view of an EAIRouterUpdate, the "routingTargets" association on its EAIRoutingTable, as shown in Figure 6-33 in Section 6.4.1.7.1, is dynamic state and therefore not appropriate for a metamodel. From the point of view of an EAIBroadcaster, the "routingTargets" of its EAIRoutingTable may also be dynamic state (if added by an EAIRouterUpdate). However, it is also desirable to be able to statically specify, in the message-flow model, the connection of EAILinks to an EAIBroadcaster. Since an EAIBroadcaster is defined to have its own output terminal, one would assume that these static EAILinks would be connected to it. Do the terminals so connected also need to be statically specified in the EAIRoutingTable? This would be the only reason to keep the "routingTargets" association in the metamodel.

Recommendation: Remove the "routingTargets" association from Figure 6-33. Further, make EAIRoutingTable a child of EAIResource and remove both the "routingTable" association (between EAIRouterUpdate and EAIRoutingTable) and the currentRoutingTable association (between EAIBroadcaster and EAIRoutingTable), instead adding the constraints that EAIRouterUpdate and EAIBroadcaster each have exactly one resource, which is an EAIRoutingTable.

The semantics for EAIRouterUpdate remains essentially unchanged. Define the semantics for EAIBroadcaster as follows:

> The target terminals of any EAILinks connected to the output terminal of an EAIBroadcaster are added to the EAIRoutingTable for that EAIBroadcaster as the initial set of routing targets. This set may be changed by the operation of an EAIRouterUpdate operator. When a message is received on the input terminal of an EAIBroadcaster, dynamic EAILinks are established between the output terminal of the EAIBroadcaster and each of the terminals in the current set of routing targets of the EAIRoutingTable of the EAIBroadcaster. The input message is then copied to the output terminal and thus sent to each of the routing targets.

Proposed Resolution:

(Note that in the Final Adopted Specification, the appropriate section is now 6.4.1.8.)

With the removal of Section 6.4.1.7, per the resolution to issue 4968, globally rename EAIBroadcaster to EAIRouter.

Revised Text

In Figure 6-33:

- *Change* EAIBroadcaster to EAIRouter.
- *Remove* the routingTargets, routingTable and currentRoutingTable associations.
- *Change* EAIRoutingTable to be a specialization of EAIResource.

Replace the entire text of Section 6.4.1.8 with

An EAIRouter routes a message to destinations listed in an EAIRoutingTable, which is maintained by EAIRouterUpdate. An EAIRoutingTable is a kind of EAIResource. An EAIRouter and an EAIRouterUpdate must each be associated with a single resource, which is an EAIRoutingTable.

An EAIRouter is a primitive operator with a single input terminal ("in") and a single output terminal ("out"). The target terminals of any EAILinks connected to the output terminal of an EAIRouter are added to the EAIRoutingTable for that EAIRouter as the initial set of routing targets. This set may be changed by the operation of an EAIRouterUpdate operator. When a message is received on the input terminal of an EAIRouter, dynamic EAILinks are established between the output terminal of the EAIRouter and each of the terminals in the current set of routing targets of the EAIRoutingTable of the EAIRouter. The input message is then copied to the output terminal and thus sent to each of the routing targets.

An EAIRouterUpdate is a primitive operator with a single input terminal ("control") and no output terminals. It expects to receive a message that conforms to the EAIRouterUpdateFormat content type. Such a message can specify either the addition (adds) or removal (removes) of a single terminal from the routing table that is associated with the operator as a resource.

Replace the entire text of Section 8.3.14 "Routers" with

Figure 8-16 shows the general format of the notation used to define a router.

A router is specified using the <<Router>> stereotype. When a router receives a message on its "in" terminal, it resends a copy, via its out terminal, to all terminals listed in an associated routing table. The routing table is shown as a class with stereotype <<RoutingTable>>, with a directed association from the router to it, with role name "routingTable".

A router updater can be used to make dynamic additions or removals of target terminals to or from a routing table. This can be used to model a simple publication channel for messages. A router updater is specified using the <<RouterUpdate>> stereotype, with a directed "routerUpdater" association from the router updater to a routing table. When a router updater receives a message on its "control" terminal that is in a router-update format, it performs the adds or removes given in that message on the associated routing table.

Note that, if a router has static EAILinks on its "out" terminal, then the target input terminals linked to it by those EAILinks are automatically added as the initial contents of the routing table for the router. If no dynamic updating is to be done on this initial contents (that is, no router updater will ever act on it),

then it is not necessary to show the routing table explicitly in the model, and the router need not have a routingTable association.

Constraints

A router must have a single input terminal labeled “in” and a single output terminal labeled “out”. The type of content of the terminals of a router must be stereotyped by <<MessageContent>> or one of its substereotypes.

A router updater must have a single input terminal labeled “control” and no output terminals. The type of content of the “control” terminal of a router updater must have the stereotype <<RouterUpdateFormat>>.

A router updater must have a directed association to a class sterotyped <<RoutingTable>>, with the role name routingTable.

Disposition: Resolved

OMG Issue No: 4971

Title: Missing specification for EAISubscriptionTable

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 6.4.1.8 (EAISubscriptionOperator) and 6.4.1.9 (EAIPublicationOperator)

Description:

The concept of a subscription table is discussed in Sections 6.4.1.8 and 6.4.1.9 as if the table was a reified entity in the metamodel (and, in fact, the class name "EAISubscriptionTable" is used in Section 6.4.1.9). However, Figure 6-35 in Section 6.4.1.8 and Figure 6-40 in Section 6.4.1.9 show, instead, the subscription table defined as individual associations from EAISubscriptionOperator and EAIPublicationOperator to EAISubscription. Not only does this define dynamic state rather than metadata, it defines DIFFERENT dynamic states for the two operators, rather than the single shared table that is necessary.

Resolution:

Remove the "subscriptionTable" association (between EAISubscriptionOperator and EAISubscription) and "currentSubscriptions" association (between EAIPublicationOperator and EAISubscription) from Figures 6-35 and 6-40. Instead, define an EAISubscriptionTable class as a child of EAIResource and put constraints on EAISubscriptionOperator and EAIPublicationOperator that each has a exactly one resource, which is and EAISubscriptionTable. The class EAISubscription (Figure 6-37) is also no longer needed as part of the metamodel.

Revised Text:

(Note that the correct section numbers in the Final Adopted Specification are 6.4.1.9 and 6.4.1.10, instead of 6.4.1.8 and 6.4.1.9.)

Replace the first paragraph of Section 6.4.1.9:

> An EAISubscriptionOperator is a subclass of EAIPrimitiveOperator with a single input terminal ("subscribe") and no output terminals. It expects an EAISubscriptionFormat as input. It adds a single EAISubscription to a subscriptionTable on receipt of an EAISubscriptionFormat.

with:

> An EAISubscriptionOperator is a subclass of EAIPrimitiveOperator with a single input terminal (“subscribe”) and no output terminals. It expects an EAISubscriptionFormat as input. On receipt of an EAISubscriptionFormat, it adds information on the specified to an EAISubscriptionTable that is attached to it as a resource.

Replace the following paragraph in Section 6.4.1.9:

> An EAISubscription relates an EAITerminal to a collection of EAISubscriptionRules. Subsequently the EAIPublicationOperator (Section 6.4.1.9) will forward messages that satisfy the subscriptionRules to the subscribingTerminal.

with:

> An EAISubscriptionTable is an EAIResource that is used to record the subscriptions received by an EAISubscriptionOperator. An EAISubscriptionOperator is an EAIPrimitiveOperator, which may have attached resources (see Figure 6-18). An EAISubscriptionOperator is specifically required to have exactly one resource, which must be an EAISubscriptionTable. An EAIPublicationOperator (see Section 6.4.1.9) referencing the same EAISubscriptionTable may then forward to subscribed target terminals messages that satisfy the subscription rules for those terminals.

Add the following constraints at the end of Section 6.4.1.9:

Constraints

An EAISubscriptionOperator has exactly one terminal, which is an input EAITerminal with the name "subscribe".

The input terminal of an EAISubscriptionOperator is associated with an EAIParameter that has a message that is an EAISubscriptionFormat.

An EAISubscriptionOperator must have exactly one resource, which is an EAISubscriptionTable.

Remove the "subscriptionTable" association from the metamodel shown in Figure 6-35. *Remove* from the metamodel the class EAISubscription shown in Figures 6-35, 6-37 and 6-40.

Replace the first paragraph of Section 6.4.1.10:

The EAIPublicationOperator models the semantics of the publish/subscribe mode of information sharing. It forwards each message to the targets specified in its currentSubscriptions, if they pass the relevant filter.

with:

The EAIPublicationOperator is used to model the publishing portion of the publish/subscribe more of information sharing. It forwards messages to target terminals recorded in the EAISubscriptionTable attached to it as a resource, if the messages meet the relevant subscription rules.

Remove the last paragraph of Section 6.4.1.10:

The diagram below shows the instance diagram for the EAISubscriptionTable after two subscriptions have been added.

and the associated Figure 6-41 (this diagram not only shows instances of EAISubscription, which is to be removed, but it badly mixes syntactic structures and dynamic data).

Add the following constraints at the end of Section 6.4.1.10:

Constraints

An EAIPublicationOperator must have exactly one resource, which is an EAISubscriptionTable.

Remove the "currentSubscriptions" association from the metamodel shown in Figure 6-40.

Disposition: Resolved

OMG Issue No: 4972

Title: The meaning of "subscriptionModes"

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.8 (EAISubscriptionOperator): Section 6.4.1.8 states "There could be some indirection in the specification of the rules and terminl [of an EAISubscriptionFormat], indicated by subscriptionModes." There is an attribute "subscriptionMode" (singular) shown in Figure 6-36 with a type "SubscriptionModes" (plural), but no further definition is given for SubscriptionModes. This provides no information on what "subscription modes" really are, or how they indicate the "indirection in the specification."

Recommendation: Either define the type SubscriptionModes and clarify its semantics or eliminate the concept.

Proposed Resolution:

(Note that the appropriate section in the Final Adopted Specificati on is now 6.4.1.9.)

Revised text

In Section 6.4.1.9, *remove* the sited sentence “(There could be some indirection in the specification of the rules and terminal, indicated by subscriptionModes.)”.

In Figure 6-36, *remove* the attribute subscriptionMode from EAISubscriptionFormat.

OMG Issue No: 4973

Title: Redundant "filterCondition" association on EAISubscriptionFilter

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.8 (EAISubscriptionOperator)

Description:

An EAISubscriptionFilter is shown in Figure 6-38 as a child of EAIFilter. As such, it already inherits a "filterCondition" association from EAIFilter (see Figure 6-26). Therefore, the additional "filterCondition" association shown in Figure 6-38 is unnecessary (and, indeed, would indicate that the EAISubscriptionFilter has two filter conditions, which does not seem to be the intent).

Resolution:

Remove the "filterCondition" association from Figure 6-38. Instead, add a constraint that the filterCondition of an EAISubscriptionFilter must be an EAISubscriptionRule.

(Note also that the multiplicity of the "filterCondition" association shown in Figure 6 -38 is "1..n" [sic], while the multiplicity of the "filterCondition" association for EAIFilter is not shown in Figure 6-26, but is implied in the text to be "1..1". If the multiplicity of the EAIFilter association is ultimately made "1..*", then the constraint on EAISubscriptionFilter should be that all the filterConditions are EAISubscriptionRules. If the desire is to have a "1..1" multiplicity on the EAIFilter association, but still to have multiple EAISubscriptionRules for an EAISubscriptionFilter, then an EAICompositeSubscriptionRule needs to be defined to group multiple EAISubscriptionRules into one FCMCondition.)

Revised Text:

(Note that the correct section in the Final Adopted Specification is 6.4.1.9 instead of 6.4.1.8.)

Remove the “filterCondition” association from the metamodel shown in Figure 6-38. *Add* the following constraint at the end of Section 6.4.1.9:

The filterCondition of an EAISubscriptionFilter is an EAISubscriptionRule.

Disposition: Resolved

OMG Issue No: 4974

Title: The lack of discussion of EAIContentRule

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.8 (EAISubscriptionOperator): Figure 6-39 shows EAITopicRule and EAIContentRule as children of EAISubscriptionRule. EAITopicRule is discussed in Section 6.4.2. However, there does not seem to be any discussion of what EAIContentRule is.

Recommendation: Describe the purpose and semantics of EAIContentRule.

Revised Text

(Note that the appropriate section in the Final Adopted Specification is now 6.4.1.9.)

In Section 6.4.1.9, *replace* the paragraph

> An EAISubscriptionRule has subclasses EAITopicRule and EAIContentRule.

with

> An EAISubscriptionRule has subclasses EAITopicRule and EAIContentRule. An EAITopicRule tests whether a message was published to one or more of an allowed set of topics, as recorded in the header for that message (see also Section 6.4.2.3, "Relationship between topic-based publishers and subscribers," on page 6-43). An EAIContentRule is a predicate that operates on the content of a message.

Disposition: Resolved

OMG Issue No: 4975

Title: EAIPublicationTerminal is not needed

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.9 (EAIPublicationOperator)

Description:

Section 6.4.1.9 asserts that a specialized EAIPublicationTerminal (stated to be a subclass of EAITerminal, though this is not shown in Figure 6-40) is needed because input messages to an EAIPublicationOperator are sent only to subscribers for which "the message conforms to the EAISubscriptionRule for that subscriber", unlike the behavior of a normal EAITerminal, "which sends a copy of the message to every target terminal". However, the EAILinks to the target terminals for messages output from an EAIPublicationOperator are not going to be statically modeled links, but are instead going to be "dynamic" EAILinks, somewhat like in the case of the "replyOut" terminal of an EAIRequestReplyOperator, determined by the current state of the subscription table for the EAIPublicationOperator. As in the case of an EAIRequestReplyOperator, it is

therefore not necessary to have a specialized kind of terminal -- the specialized message distribution behavior is captured in the operator, not in the terminal.

Resolution:

Eliminate the EAIPublicationTerminal. Define the semantics of an EAIPublicationOperator as follows:

When a message arrives at the input terminal of an EAIPublicationOperator, the EAISubscriptionRules for all subscriptions in the current state of the subscriptionTable are evaluated on the message. For each subscription for which the rule is true, a dynamic EAILink is established from the output terminal of the EAIPublicationOperator to the subscriber EAITerminal from the subscription. The input message is then copied to the output terminal and thus distributed to each subscriber.

Revised Text:

(Note that the correct section in the Final Adopted Specification is 6.4.1.10 instead of 6.4.1.9.)

Replace the following paragraphs in Section 6.4.1.10:

> It is modeled as a subclass of EAIPrimitiveOperator, with a single input terminal ("in"), and a single output terminal. Messages sent to the input terminal are sent from the output terminal ("out") to each subscriber (EAITerminal) if the message conforms to the EAISubscriptionRule for that subscriber.
>
> This output behavior is not the same as that of EAITerminal, which sends a copy of the message to every target terminal. Therefore a subclass of EAITerminal is introduced called EAIPublicationTerminal.

with:

> An EAIPublicationOperator is an EAIPrimitiveOperator with a single input terminal (“in”) and a single output terminal (“out”). When a message arrives at the input terminal, the EAISubscriptionRules for all subscriptions in the current state of the EAISubscriptionTable are evaluated on the message. For each subscription for which the rule is true, a dynamic, temporary EAILink is effectively established from the output terminal to the subscriber EAITerminal from the subscription. The input message is then copied to the output terminal and thus distributed to each subscriber.
>
> If the target terminal of a dynamic EAILink is *not* an EAIQueuedInputTerminal, then the dynamic EAILink is considered to have *synchronization = unspecified.*The published message is simply placed on the identified target terminal. However, if the identified target terminal *is* an EAIQueuedInputTerminal (see Section 6.3.8), then the dynamic EAILink is

considered to have *synchronization = asynchronous* and the published message is placed on the inputQueue of the target terminal.

Add the following constraints at the end of Section 6.4.1.10:

An EAIPublicationOperator has exactly one input terminal, which is an EAITerminal with the name “in”, and exactly one output terminal, which is an EAITerminal with the name “out”.

The messages of the EAIParameters associated with the two terminals of an EAIPublicationOperator must be the same.

Disposition: Resolved

OMG Issue No: 4976

Title: Missing specification of a table to hold EAIMessageTimerConditions

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 6.4.1.10.1 (EAITimeSetOperator) and 6.4.1.10.2 (EAITimeCheckOperator)
Description: Figure 6-42 in Section 6.4.1.10.1 and Figure 6-45 in Section 6.4.1.10.2 show the individual associations from EAITimeSetOperator and EAITimeCheckOperator to EAIMessageTimerCondition. Not only does this define dynamic state rather than metadata, it defines DIFFERENT dynamic states for the two operators, rather than the single shared table that is necessary.

Recommendation: Remove the "timeSetConditions" association (between EAITimeSetOperator and EAIMessageTimerCondition) and "timeCheckConditions" association (between EAITimeCheckOperator and EAIMessageTimerCondition) from Figures 6-42 and 6-45. Instead, define an EAITimerConditionTable class as a child of EAIResource and put constraints on EAITimeSetOperator and EAITimeCheckOperator that each has a exactly one resource, which is and EAITimerConditionTable.

Resolution:

This and the associated issue 4976 refer to section 6.4.1.10.1, but 02-02-02.pdf has no such section.
Section 6.4.1.11 - EAI Timer, has a subtitle TimeSetOprerator
Section 6.4.1.12 is entitled EAITimeCheckOperator
Section 6.4.1.13 is entitled EAITimer, with no subtitle.
There is a corruption of the numbering here somehow.

For its resolution, I suggest renumbering/reorganizing existing text:

Rationale: EAITImer is composed of EAITimeSetOperator & EAITimeCheckOperator so arrange it as such.

Revise text and diagrams.

Revised Text:

Keep 6.4.1.11 entitled EAI Timer as is
All text & diagram text is to be transferred f rom section 6.4.1.13 to immediately below this heading
New section 6.4.1.11.1 entitled EAITimeSetOperator
Promote the subtitle of 6.4.1.11 to be this sub-section, it then contains all the text & diagrams of the existing 6.4.1.11
Rename as 6.4.1.11.2 current section 6.4.1.12 EAITimeCheckOperator
Keep existing text & diagrams.

Remove existing section 6.4.1.13.

In Figures 6-42 and 6-45 show EAITimerSetOperator and EAITimerCheckOperator as subclassing EAIPrimitiveOperator without the resource associations to EAITimerConditionTable.

Add a section on EAITimerConditionFormat that defines it as a subclass of EAIMessageContent that provides a definition of a timer condition and a means of identifying the messages to which the condition apply, along with an appropriate metamodel diagram.

Change the first paragraph of 6.4.1.11.1 EAITimeSetOperator (new numbering) to:

> The EAITimeSetOperator is a subclass of EAIPrimitiveOperator, with a single input terminal ("set") and no output terminals. On receipt of a message, which must be specified by an EAITimerConditionFormat, it adds the timer and message applicability conditions given by the message to the list of conditions stored in the EAITimerConditionTable that is attached to it as a resource.

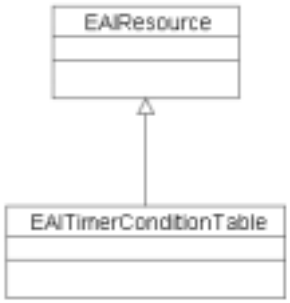
It is also necessary to change the first paragraph of 6.4.1.11.2 EAITimeCheckOperator to:

> EAITimeCheckOperator is a subclass of EAIPrimitiveOperator with a single input terminal ("check") and three output terminals ("ontime", "expiry" and "late"). On receipt of a message, it examines the set of conditions stored in the EAITimerConditionTable that is attached to it as a resource. If there is a timer condition that applies to the message, it checks the condition is actually met. If so, then the message is passed to
> the "ontime" terminal; if not, it is passed to the "late" terminal.

Finally, add the constraints given above to the appropriate sections.

Figure 6.42

Figure 6.45



New figure in Section 6.3.12

Disposition: Resolved

OMG Issue No: 4978

Title: It is unclear how a message is associated with a topic

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.2 (Topic-based publish/subscribe) discusses the relation between topic-based publishers and subscribers. However, the semantics of an EAIPublicationOperator (see Section 6.4.1.9) require that a message conform to an EAITopicRule (a kind of EAISubscriptionRule) in order to be published on a topic. But it is not clear how to determine that a message is on a topic just from looking at that message. Presumably, messages produced by an EAITopicPublisher (Section 6.4.2.1) are somehow tagged as being on a specific topic, but this is not said explicitly.

Recommendation: At the very least, explicitly state in Section 6.4.2.1 that messages produced by an EAITopicPublisher are such that they satisfy the EAITopicRule for a one or more EAITopics relevant to the EAITopicPublisher. However, if the EAIPublicationOperator can then determine topic(s) for a message just by evaluating a condition on the message, it would seem that the topic(s) must be encoded in the message content someplace, in which case it is unclear what the difference is between an EAITopicRule and an EAIContentRule. Perhaps it would be best just to eliminate EAIContentRule, regarding this as being covered by the general case of EAISubscriptionRule, and have EAITopicRule as a specialized EAISubscriptionRule for which the condition is that the message is on one of a given set of topics. In this case, an

association should be added from EAITopicRule to EAITopic with a multiplicity of "1..*".

Resolution:

The relationship between an EAITopicPublisher and an EAITopicRule is, in fact, discussed in Section 6.4.2.3. Note, however, that some confusion may be caused by the fact that that Figure 6-48 is incorrectly a repetition of the diagram in Figure 6-49. Figure 6-48 should instead show the derived association of an EAITopicRule to the EAITopics it allows (which is, an association with multiplicity 0..*, as shown in Figure 6-49).

Disposition: Resolved

OMG Issue No: 5222

Title: Incorrect description of Figure 8-1

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The first paragraph of Section 8.2 describes the "prototypical example" of the notation for terminals shown in Figure 8-1 as follows: "This shows a primitive operator with two input and two output terminals. The output terminals are of the same kind, but the input terminals are not (one is known to be a queued terminal, even though they both handle the same kind of message format). The names of the terminals are, in this case, label1 and label2."

Resolution:

This description has two problems:

- The diagram only shows one input terminal (the queued terminal is not shown).
- The text only lists two of the three terminal labels (the label "outName2" for the second output terminal is not listed.

Recommendation:

1. Show a queued input terminal on Figure 8-1.

2. List all the terminal names, identifying which are the names of input terminals and which of output terminals (if possible, use better names, too).

Revised Text:

Replace the following text in Section 8.2:

> The terminals of an operator are shown by associations to classes with stereotypes <<input>> (for input terminals) and <<output>> (for output terminals), from classes with operator stereotypes (see sections below). A prototypical example showing the definition of terminals for a primitive operator is given in Figure 81Default ¶ Font. This shows a primitive operator with two input and two output terminals. The output terminals are of the same kind, but the input terminals are not (one is known to be a queued terminal, even though they both handle the same kind of message format). The names of the terminals are, in this case, label1 and label2.

with:

> The terminals of an operator are shown by associations to classes with stereotypes <<input>> (for input terminals) and <<output>> (for output terminals), from classes with operator stereotypes (see sections below). Figure 8-1 gives a prototypical example, showing the definition of terminals for a primitive operator. As shown, the primitive operator has two input terminals, named “in” and “queueIn”. While both these terminals handle the same kind of message format, the latter is specifically shown to be a queued terminal. The primitive operator is also shown to have two output terminals, named “out1” and “out2”.

Add a queued input terminal to the diagram in Figure 8-1. *Change* the labels for the input terminals to “in” and “queuedIn” (the latter being the queued terminal) and for the output terminals to “out1” and “out2”.

Disposition: Resolved

OMG Issue No: 5223

Title: Terminal labeling constraints

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Under the Constraints heading in Section 8.3.2 it states that "The input terminal must be labelled 'in' and the output terminal must be labelled 'out'." However, there is no

constraint on the names of the terminals of EAITransformers in the metamodel (see Section 6.4.1.4).

(Similar constraints appear in Sections 8.3.3, 8.3.4, 8.3.5, 8.3.6, 8.3.7, 8.3.10, 8.3.11, 8.3.12, 8.3.15, 8.3.16 and 8.3.17 without corresponding constraints in the metamodel.)

Resolution:

Unless there an overriding notational reason can be stated for requiring specific name s for terminals, do not require names when they are not required by the metamodel. (Though it might be appropriate to recommend specific consistent naming conventions.)

Revised Text:

Delete the following text under the Constraints heading in Section 8.3.2:

> The input terminal must be labelled in and the output terminal out.

Replace the following text:

> The content format of in and out must match the format of the parameter and result, respectively, of the transform operation.

with:

> The content format of the input and output terminals must match the format of the parameter and result, respectively, of the transform operation.

Make similar changes in Sections 8.3.3, 8.3.4, 8.3.5, 8.3.6, 8.3.7, 8.3.10, 8.3.11, 8.3.12, 8.3.15, 8.3.16 and 8.3.17.

Disposition: Resolved

OMG Issue No: 5224

Title: Poor wording of constraint on association rolename of a database resource

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The last constraint in Section 8.3.2 states that "For database transformers, there must be a directed association to a database resource (i.e., a class with stereotype <<database>>). This should be labeled 'database'."

In the second sentence, it would seem that "this" refers to the datanase resource or the directed association. In reality, it is the rolename of the resource that should be "database".

Resolution:

Change the second sentence to say that the rolename of the database resource must be 'database'.

Revised Text:

Replace the following text at the end of Section 8.3.2:

> For database transformers, there must be a directed association to a database resource (i.e., a class with stereotype <<Database>>). This should be labeled database.

with:

> For database transformers, there must be a directed association to a database resource (i.e., a class with stereotype <<Database>>), with the rolename “database” at the database resource end.

Disposition: **Resolved**

OMG Issue No: 5225

Title: Lack of semantics for a "false" terminal on an EAIFilter in the metamodel

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section 8.3.3 states that a filter has two output terminals, labeled "true" and "false", and, if a message content meets the filter criteria, then "the content is sent to the 'true' output terminal, otherwise it is sent to the 'false' output terminal." However, this is inconsistent with Section 6.4.1.1 on EAIFilter in the metamodel, which states that "A filter's output is a copy of its input. No output occurs if the input message does not satisfy the filter condition." There is no semantics given for a "false" terminal.

Resolution:

Having "true" and "false" outputs on a filter is quite useful. The semantic descriptions in Section 6.4.1.1 should be changed to reflect the semantics of true/false output terminals.

Revised Text:

Replace the following text at the end of Section 6.4.1.1:

> A filter's output is a copy of its input. No output occurs if the input message does not satisfy the filter condition.

with:

> A filter has two one input terminal and two output terminals. The output terminals must be named “true” and “false”. If the message on the input terminal satisfies the filter consition, then it is copied to the output terminal named “true”. Otherwise, the message is copied to the output terminal named “false”.

Disposition: Resolved

OMG Issue No: 5237

Title: Update to Type Descriptor Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

```
This model needs to be updated to contain additional mapping information
from source language files.  Issue discovered at implementation time.

This description has 3 problems:
Introduce Bi-DirectionStringTD class
Add attributes to InstanceTDBase, PlatformCompilerInfo, StringTD, and
DateTD classes
Change structure of NumberTD classes.
```

Resolution:

```
Update model to include missing information
```

Revised Text:

```
1. Remove the ‘Formula’ suffix to the following attribute names:
   “offsetFormula” and “contentSizeFormula” (from InstanceTDBase
   class), “strideFormula,” “upperBoundFormula”, and
   “lowerBoundFormula” (from ArrayTD class).

Reason: Unnecessary suffix naming convention.

2. Rename the following attributes in InstanceTDBase:
   “allocSizeFormula” to “size”, “formulaInBit” to “attributeInBit”.

Reason: Removal of ‘Formula’ suffix to naming convention.  See point
4 above.
```

3. Rename “arrayAlign” attribute in ArrayTD to “alignmentKind” and changed return datatype from int to AlignType. Change return type of alignment attribute in BaseTDType from int to AlignType.

Reason: ‘array’ prefix naming for “arrayAlign” is redundant for the holder class. Return datatype changed from int to AlignType provides enumeration of possible alignment values.

4. Add “format” attribute to SimpleInstanceTD class.

Reason: To support declaration format of elements for languages such as COBOL. E.g., 01 DATE 9999/99/99.

5. Rename the following attributes in PlatformCompilerInfo:

“osVersion” to “OSVersion”, “addressSize” to “defaultAddressSize,” “defaultEncoding” to “defaultCodepage”.

Reason: Capitalize ‘OS’. Make “addressSize” attribute a default attribute. Encoding information is now captured in ExternalDecimalSignValue. Information for codepage is needed.

6. Add the following attributes to PlatformCompilerInfo:

“language” and “defaultExternalDecimalSign”.

Reason: The “language” attribute specifies the language associated with the instance TD model. “defaultExternalDecimalSign” specifies the encoding of external decimal signs for languages such as COBOL.

7. Remove “union” attribute from AggregateInstanceTD

Reason: Information is captured in language model

8. Rename “encoding” attribute in StringTD class to “codepage”.

Reason: Information for codepage is needed.

9. Remove “maxLengthFormula” attribute and add “prefixLength” and “DBCSOnly” attributes in StringTD class.

Reason: Changes in requirement.

10. Change association of Bi_DirectionStringTD class from inheritance from StringTD to aggregated association with StringTD and PlatformCompilerInfo.

11. Remove the following attributes from NumberTD class: “signCoding,” “checkValidity,” “packedDecimalSign,” “baseUnitEncoding,” “format,” and “sign”.

12. Add the following attributes from NumberTD class: "signed" and "virtualDecimalPoint".

13. Create the following subclasses (with their respective attributes) under NumberTD class: ExternalDecimalTD, PackedDecimalTD, and IntegerTD.

14. Remove "length" attribute from BinaryTD class.

15. Add "codepage" attribute to DateTD class.

16. Update text in section 7.3.5 to describe the meaning of "level-1 data structure" and "level-1 parent."

Clarify text for TDLangModelElement class on instantiating TDLangClassifier and TDLangElement subclasses.

Disposition: **Resolved**

OMG Issue No: 5238

Title: Update to TDLang Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

This model needs to be updated to contain additional mapping information from source language files. Issue discovered at implementation time.

This description has 1 problem:
Change association type for TDLangComposedType to TDLangElement

Resolution:

Update model to include missing information.

Revised Text:

1. Change bi-directional association between TDLangComposedType to TDLangElement to uni-directional.

   Reason: TDLangElement already has access to parent class TDLangComposedType thru tdLangGroup. No need to set bi-direction.

2. Remove '/' marks from model to show associations are derived from subclasses.

   Reason: Associations are not in fact derived. Rather they emulate the association relationships of its subclasses.

3. Add {Ordered} decoration to TDLangComposedType to TDLangElement association.

Reason: Addition of each instance of TDLangElement to TDLangComposedType should be noted as ordered.

Disposition: **Resolved**

OMG Issue No: 5239

Title: Update to COBOL Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

Discovered model needs to be updated to contain additional mapping information from original COBOL source files. Issue discovered at implementation time.

Resolution:

Update model to include missing information.

Revised Text:

1. Remove getCanonicalPictureString() method from COBOLSimpleType class.

2. Change return type of “maxUpper” and “minUpper” attributes in COBOLFixedLengthArray and COBOLVariableLengthArray classes, respectively from Integer to int.

3. Rename “currencySymbol” attribute in COBOLNumericType and COBOLNumericEditedType class to “currencySign” and change return type from char to String.

4. Add “national” as an enumeration value in COBOLUsageValues

Disposition: **Resolved**

OMG Issue No: 5240

Title: Update to C Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

Discovered model needs to be updated to contain additional mapping information from original C source files. Issue discovered at implementation time.

This description has 1 problem:
Update associations between two C Classes.

Resolution:

Update model to include missing information

Revised Text:

1. Added CDirectionKind enumeration.

2. Add a 'C' prefix to the following datatype and enumeration classes: "String," "Integer," and "Boolean".

3. Change association of CInteger, CFloating, CBitField, and CVoid to inheritance association under CDatatype.

4. Add association from CBitField to CInteger.

5. Remove CNamedElement class. Pass down "name" attribute to CBehavioralFeature.

6. Rename "derives" and "derived" associations between CTypedElement and CDerived to "container" and "contains", respectively.

Disposition: Resolved

OMG Issue No: 5241

Title: Update to MFS Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

Discovered model needs to be updated to contain additional mapping information from original MFS source files. Issue discovered at implementation time.

This description has 3 problems:
Introduce DIVISION class
Add more attributes to classes
Update association to TDLang

Resolution:

Update model to include missing information

Revised Text:

1. Rename the following classes: MFSMessageDescriptor to MFSMessage, MFSDeviceDescriptor to MFSFormat, MFSDeviceType to MFSDevice, MFSDeviceDivision to MFSDivision, MFSFunctionKeyType to MFSFunctionKeyList, MFSCursorType to MFSCursor, MFSAttributeType to MFSAttribute, MFSOutliningType to MFSOutlining, MFSPositionType to MFSPosition, MFSPageFormattingType to MFSLineFormat, MFSHightlightingType to MFSHighlighting, MFSValidationType to MFSValidation, MFSDetectabilityType to MFSDetectability, MFSIntensityType to MFSIntensity, MFSExtendedAttributeType to MFSExtendedAttribute, MFSExitType to MFSExit, MFSCompressionType to MFSCompression MFSColorType to MFSColor, MFSOperatorType to MFSConditionOperator, MFSJustifyType to MFSJustification, and MFSDescriptorType to MFFormatType.

2. Add {Ordered} decoration to the following associations: “logicalPages,” “devices,” “divisions,” “devicePages,” “physicalPages,” “deviceFields,” “messageFields,” “conditions,” “segments,” and “functionKeys”

3. Add the following classes (with their respective attributes and associations): MFSLogicalPageCondition, MFSDivision, MFSPhysicalPage, MFSFunctionKey, MFSIfCondition, MFSPageFormat, MFSPen, MFSConditionType, MFSControlFunction, MFSSystemLiteral, MFSMessageType

4. Changed all attribute return types from “Boolean” to “boolean”.

5. Update the following attributes in MFSDeviceField: Change return type of “pen” from String to MFSPen. Add “password” attribute with a return type of Boolean.

6. Update the following attributes in MFSMessage: Rename “ignoreSource” attribute to “ignore”. Change return type of “type” attribute from MFSDescriptorType to MFSMessageType.

7. Update the following attributes in MFSMessageField: Change return type of “length” attribute from MFSLengthType to int. Rename “value” attribute to “literal”. Add “firstByte,” “systemControlArea,” and “systemLiteral” attributes.

8. Add “lineLength” attribute to MFSFeature.

9. Update the following attributes in MFSFunctionKey: Remove “functionList” attribute. Add “controlFunction” and “literal” attribute.

10. Update the following attributes in MFSDevice: Remove “dsca,” “page,” and “pfk” attributes. Add “defaultSystemControlArea,” “functionKeyList,” and “pageFormat” attributes.

11. Remove the following classes: MFSConditonType, MFSLengthType, and MFSPageType

12. Update attributes in MFSIfCondition, MFSOutlining, MFSDevicePage, MFSLogicalPage, MFSIntensity, and MFSPosition class.

Disposition: **Resolved**

OMG Issue No: 5242

Title: Update to BMS Metamodel

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

Discovered model needs to be updated to contain additional mapping information from original BMS source files. Issue discovered at implementation time.

This description has (X number of) problems:
Add more attributes to classes
Update association to TDLang

Resolution:

Update model to include missing information.

Revised Text:

1. Change uni-directional association from BMSField to TDLangElement to BMSField inherit from TDLangElement.

2. Add the following classes (with their respective attributes and associations): BMSWriteableType, BMSLineType, BMSPSType, BMSPartitionType, BMSFieldJustifyType, BMSColumnType, BMSDSAttributeTypes, BMSYesNoType, BMSDSectType, BMSWebField, BMSDisplayableType.

3. Rename BMSJustifyType to BMSMapJustifyType.

4. Update attributes in the following classes: BMSMapJustifyType, BMSAttributeType, and BMSControlType.

Disposition: **Resolved**

OMG Issue No: 5243

Title: Update to Convergent Metamodel (Figure 64)

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

```
additional subclasses to TDLangElement.

This description has 1 problem:
Add BMS and MFS models under TDLangElement
```

Resolution:

```
Update figure to include missing information.
```

Revised Text:

```
Add MFSMessageField and BMSField classes as subclasses of TDLangElement
class.
```

Disposition: **Resolved**

OMG Issue No: 5244

Title: Update Sample XMI in Section 7.3.11

Source:

IBM (Shyh-Mei Ho shyhmei@us.ibm.com)

Summary:

```
Fill in additional information in sample.

This description has 1 problem:
Fill in additional information in sample.
```

Resolution:

```
Update sample to include missing information.
```

Revised Text:

Update xmi example with a complete xmi listing.

Disposition: **Resolved**

OMG Issue No: 5246

Title: Missing request format Y9

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Figure 8-10 repeats the <>s Y3 and Y8 from Figure 8 -9, but not the <> Y9.

Resolution:

Show Y9 on Figure 8-10.

Revised Text:

In section 8.3.9, replace Figure 8-10 with the NewFigure 8-10 (below)

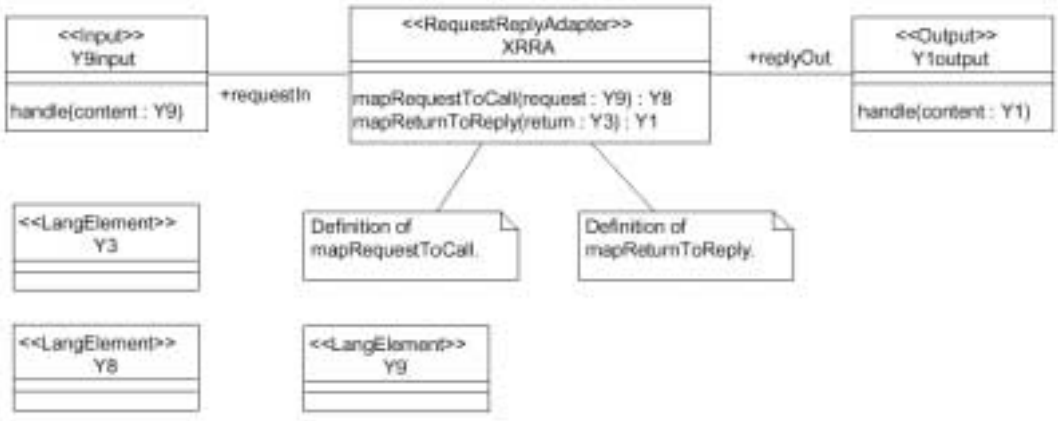


Figure 8-10 Class diagram from prototypical request/reply adapter

Disposition: Resolved

OMG Issue No: 5247

Title: Sources and Sinks are called Operators in the profile but not in the metamodel

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections 8.3.10 and 8.3.11 describe sources and sinks as kinds of operators. However, in Section 6.3.6, the corresponding metamodel elements are NOT defined as subclasses of EAIOperator, but rather are directly subclasses of FCMSource and FCMSink. Indeed, EAI sources and sinks cannot be operators, if they are really to serve the role of FCMSources and FCMSinks (which is to provide the internal view within an FCMCommand of the external terminals of that FCMCommand), since operators are FCMFunctions, and FCMSources and FCMSinks are not.

Issue Raiser's Recommendation
Do not describe sources and sinks as operators. Move the description of sources and sinks out of Section 8.3 on operators.

Resolution:

Issues 5245 & 5247 relate to parallel inconsistencies in the profile definitions of Adapters and Sources & Sinks. The isssue raiser recommends the same solution for each - to move them to their own sections at the same level as Operators - as they are in the metamodel definition.
I agree with this proposal that we regard the metamodel as the master and rearrange the profile definition accordingly. Each new section involves a considerable amount of moving & renumbering sub-sections, Figures, Tables & lists in section 8.3. To ease the task of the editor, i have merged the edits for these 2 issues (5245 & 5247) into one series (below). I have also amended the Parent stereotypes in the table of mappings to match the metamodel, in line with the issues raised. These particular details are not mentioned by the issue raiser, but they are implied by his correction.

Revised Text:

1. Create a new section '8.4 Adapters'
2. Create a new section '8.5 Sources and Sinks'
3. Renumber current sections 8.4 through 8.6 (and their sub-sections) 8.6 through 8.8
4. Move the current sub-sections 8.3.6 through 8.6.9 under the new section 8.4, renumbering as 8.4.1 to 8.4.4
5. Move the current sub-sections 8.3.10 and 8.3.11 under the new section 8.5, renumbering as 8.5.1 & 8.5.2
6. Renumber Figures 8-14 through 8-28 to Figures 8-7 through 8-21 respectively
7. Renumber Figures 8-7 through 8-13 to Figures 8-22 through 8-28 respectively
8. Add new sub-sections '8.8.3 Adapters' and '8.8.4 Sources and Sinks' under (new) section 8.8 (was 8.6)

9. Renumber existing sub-sections 8.8.3 & 8.8.4 to 8.8.5 & 8.8.6 (current after the renumbering in step 3 above)

10. Add a new table 'Table 8-5 Mapping of Adapters' in (new) sub-section 8.8.3 with the same format as Table 8-4.

11. Move the 4 rows relating to Adapters (Source, Target, Call & Request/Reply) from Table 8-4 to Table 8-5

12. Remove the references in the Parent column of this table (should be null).

13. Replace the references, in the Description & Constraints column, to sub-sections 8.3.6 through 8.3.9 with references to sub-sections 8.4.1 through 8.4.4 respectively.

14. Create a new sub-heading 'Mapping Constraints' in (new) sub section 8.8.3 below table 8-5.

15. Move constraints 15 through 19, and their respective sub-sub-headings, from section 8.8.2 to under the Mapping Constraints sub-heading in (new) 8.8.4, re-numbering them 35 through 39

16. Add a new table 'Table 8-6 Mapping of Sources' and Sinks in (new) subsection 8.8.4 with ther same format as Table 8-4.

17. Move the 4 rows relating to Sources & Sinks (Source, QSource, Sink, QSink) from Table 8-4 to Table 8-6.

18. Remove the references in the Parent column of the Source and Sink rows of this table (should be null).

19. Replace the references, in the Description & Constraints column, to sub-sections 8.3.10 and 8.3.11 with references to sub-sections 8.5.1 through 8.5.2. respectively.

20. Create a new sub-heading 'Mapping Constraints' in (new) sub section 8.8.4 below table 8-6.

21. Move sub-sub-heading 'EAISource, EAIQueuedSource, EAISink, EAIQueuedSink' and associated text ' There are no further constraints' from section 8.8.2 to under the Mapping Constraints sub-heading in (new) 8.8.4.

22. Renumber the remaining constraints 20 through 39 in sub-section 8.8.3 to15 through 34

23. Renumber (old) Tables 8-5 & 8-6 to Tables 8-7 & 8-8 (note there is another table, currently unnumbered - if this has not been raised as an issue elsewhere, number it Table 8-9).

Disposition: **Resolved**

OMG Issue No: 5248

Title: Diagram the queue for queued sources and sinks

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 8.3.10 (Sources and Queued Sources), 8.3.11 (Sinks and Queued Sinks)

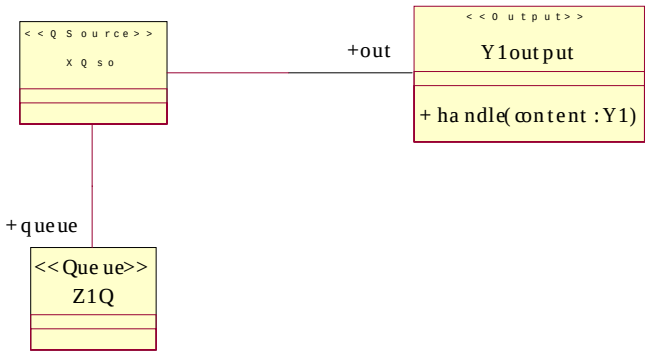
Description:
The constraints in Sections 8.3.10 and 8.3.11 require that queued sources and sinks have "a directed association to a queue resource". However, this is not shown in Figure 8-12 ("Class diagram for prototypical queued source") and there does not seem to be a sample diagram for a queued sink at all.

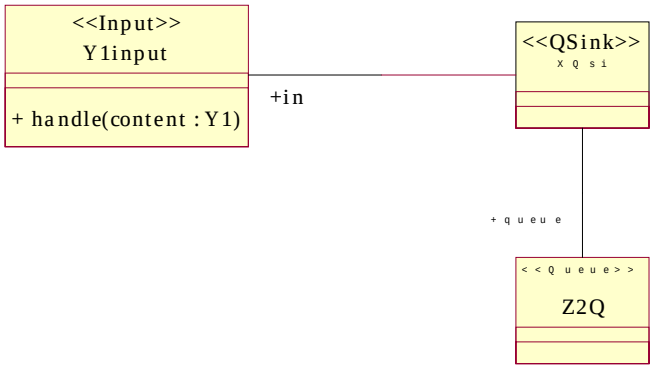
Resolution:

Draw the queue resource in Figure 8-12 and include a diagram with an example of a queued sink.

Revised Text:

Diagrams:





Disposition: **Resolved**

OMG Issue No: 5249

Title: Typographical errors in Figure 8-14 on aggregators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.12 (Aggregators): In Figure 8-14, the name of the operation "aggregateCompleted" is inconsistent with the text and the metamodel and the operation name "aggregateToAggregate" is incorrect in the leftmost note.

Resolution:

Change Figure 8-14.

Revised Text:

Change "aggregateCompleted" to "aggregateComplete" in the operation definition.
Change "aggregateToAggregate" to "addToAggregate" in the note in Figure 8-14.

Disposition: **Resolved**

OMG Issue No: 5250

Title: Insufficiency of the metamodel mapping for aggregators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.12 (Aggregators):
The metemodel for EAIAggregator in Section 6.4.1.6 allows a DIFFERENT aggregateComplete and addToAggregate condition for EACH aggregate being formed. However, Section 8.3.12 only provides for the specification of a one aggregateComplete and one addToAggregate operation for the entire aggregator. (These operations take a specific aggregate as an argument, but the BEHAVIOR of the operation will be the same for all aggregates.)

Resolution:

Define a new <<MessageAggregation>> stereotype. A class with this stereotype must have aggregateComplete and addToAggregate operations. Such a class maps to the EAIMessageAggregation metaclass (see Section 6.4.1.6). Require that a class with the stereotype <<Aggregator>> have associations with one or more <<MessageAggregation>> classes. (Note that multiple message aggregations can be achieved both by having an association with a multiplicity at the message aggregation end of greater than 1 or by having multiple associations with different message aggregation classes with different operator specifications).

(Also change mapping constraints 20 and 21 in Section 8.6.2 to be consistent with this.)

Revised Text:

In Section 8.3.12, *replace* the following paragraph:

> An aggregator operator is indicated by the <<Aggregator>> stereotype. On receipt of a message at its input terminal, if there are no existing message aggregates, the aggregator creates one and adds the message to it. On receipt of a subsequent message, the aggregator examines each existing aggregate, evaluating the addToAggregate condition (which will depend on the message header or body contents). If an aggregate exists for which addToAggregate evaluates to true, then the message is added to it.

with:

> An aggregator operator is indicated by the <<Aggregator>> stereotype. It creates aggregate messages based on one or more message aggregation specification, each of which is modeled by an associated class with the <<MessageAggregation>> stereotype. (Note that an aggregator can create multiple aggregates either by having an association with a multiplicity of greater than one with the *same* message aggregation class, in which case all aggregates share the same specification, or by having multiple associations with *different* message aggregation classes.)

On receipt of a message at it's input terminal, the aggregator operator adds the message to each aggregate for which the addToAggregate condition (which will depend on the message header or body contents) evaluates to true.

Each time a message is added to an aggregate, the aggregateComplete condition is evaluated for that aggregate. If it evaluates to true, then a message is constructed from the messages it holds and is sent on the output terminal. The mapping from the messages contained in the aggregate to the message sent is specified by the aggregate operation.

and *replace* the following paragraph:

If the aggregateComplete condition does not evaluate to true, then no message is sent.

with:

If no aggregateComplete condition evaluates to true, then no message is sent.

Add the following constraints:

The aggregator class must have associations with one or more classes with the stereotype <<MessageAggregation>>.

A class stereotyped <<MessageAggregation>> must have addToAggregate, aggregationComplete and aggregate operations.

Update Figure 8-14 to show the <<MessageAggregation>> classes.

In Section 8.6.2, *replace* the following items:

20. The aggregateComplete condition of the operator corresponds to the aggregateComplete operation in the corresponding class.
21. The addToAggregate condition of the operator corresponds to the addToAggregate operation in the corresponding class.
22. The aggregationMapping of the operator corresponds to the aggregate operation in the corresponding class.

with

20. The aggregateComplete condition of each EAIMessageAggregation of the operator corresponds to the aggregateComplete operation in the corresponding <<MessageAggregation>> class.

21. The addToAggregate condition of each EAIMessageAggregation of the operator corresponds to the addToAggregate operation in the corresponding <<MessageAggregation>> class.

22. The aggregationMapping of each EAIMessageAggregation of the operator corresponds to the aggregate operation in the corresponding <<MessageAggregation>> class.

Disposition: **Resolved**

OMG Issue No: 5251

Title: Incorrect constraint for aggregators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

The second constraint in Section 8.3.12 (Aggregators) states: "The content format of in and out must match the format of the parameter and result, respectively, of the transform operation." However, aggregators do not have "transform" operations.

Resolution:

Change the constraint to describe the types required for the parameters and results of the addToAggregate, aggregateComplete and aggregate operations required on an aggregator.

Revised Text:

Remove the following constraint from Section 8.3.12:

> The content format of in and out must match the format of the parameter and result, respectively, of the transform operation.

Add the following constraints at the *end* of the section:

> The addToAggregate operation of each message aggregation class must have two arguments, the first of which matches the content format of the in terminal of the aggregator operator and the second of which is a sequence of this content format, and a result of type Boolean.
>
> The aggregationComplete operation of each message aggregation class must have a single argument whose type is a sequence of the message content format of the in terminal of the aggregator operator and a result of type Boolean.
>
> The aggregate operation of each message aggregation class must have a single argument whose type is a sequence of the message content format of the in terminal of the aggregator operator and a result whose type matches the content format of the out terminal of the aggregator operator.

Disposition: **Resolved**

OMG Issue No: 5252

Title: Incorrect notation for message arrows in Figure 8-24

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.18.2 (Collaboration diagrams)
The arrows for synchronous and asynchronous messages shown in Figure 8-24 do not use the correct UML 1.4 notation.

Resolution:

Use the correct UML notation in Figure 8-24: an arrow with a filled, solid arrowhead for synchronous and an arrow with a stick arrowhead for asynchronous (see Section 3.72.2.1 of formal/01-09-67).

Revised Text:

Update Figure 8-24 as recommended. [Note that, using Rose, one needs to select a "synchronization" of "simple" to get the stick arrowhead (the standard notation for asynchronous) and "procedure call" to get the solid arrowhead (the standard notation for synchronous).]

Disposition: **Resolved**

OMG Issue No: 5345

Title: Modeling Approach: Phrasing of delivery

Source:

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

Summary:

End 3.0: slightly wrong to say it's delivered as a metamodel and profile - there are several.

Resolution:

Clarify the introductory wording to chapter 3.

Revised Text:

Old: The EAI specification is delivered as a complete MOF-based metamodel and a UML profile.

New: The EAI specification is delivered as a complete MOF-based metamodel and a UML profile, which actually consists of two profiles, one for collaboration modeling and one for activity modeling.

Disposition: **Resolved**

OMG Issue No: 5346

Title: Metamodel: Use UML profile for MOF

Source:

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

Summary:

Section 3.1 Should refer to UML Profile for MOF (part of EDOC). Packages are structural to the resultant model and should not be scoped by what can fit onto one diagram.

Resolution:

Update section 3.

Revised Text:

Old: Packages are limited in size so that only one class diagram per package is required

New: At the lowest level, packages are limited in size, and only one class diagram per package is required

Disposition: **Resolved**

OMG Issue No: 5348

Title: Compliance/Visualization: Clarification of visualization requirement

Source:

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

Summary:

Section 4.2-4.3
Unclear what visualization is required above UML.
Hence point out that any UML compliant tool will also be EAI Profile compliant?
Requiring that tools enforce constraints goes further than MOF and is hard when the constraints have not been formally specified in either the spec nor the XMI files.
According to 5.1.1 Activity and Collaboration representations are alternatives. This is not reflected in section 4.

Resolution:

Update sections 4.2.2 and 4.3.2 as below.

In section 4.2.1 and 4.3.1, delete: Furthermore it checks the well-formedness constraints that the Profile defines.

In section 4.4 delete: It also checks the well-formedness constraints defined by the metamodel.

Section 4 does give examples of compliance with separate compliance with the Collaboration or Activity representations.

Revised Text:

Sections 4.2.2 and 4.3.2:

Old: A compliant implementation supports the UML notation for the packages extended by the Collaboration Profile and for the EAI extensions to those packages.
A compliant implementation supports the UML notation for the packages extended by the Activity Profile and for the EAI extensions to those packages.

New: An implementation satisfies the Visualization compliance point if it supports the UML notation for the packages extended by the Collaboration Profile and for the EAI extensions to those packages.

An implementation satisfies the Visualization compliance point if it supports the UML notation for the packages extended by the Activity Profile and for the EAI extensions to those packages.

Disposition: Resolved

OMG Issue No: 5349

Title: Need to qualify profile names with EAI prefix

Source:

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

Summary:

```
Section: 4.3 It's unclear to just refer to 'UML Activity Profile' and
'UML Collaborations Profile' without the 'for EAI' qualification.
```

Resolution:

Update sections 4.2.1 and 4.3.1

Revised Text:

Old: A compliant implementation supports the UML XMI exchange mechanism for the UML packages extended by the Collaboration Profile.
A compliant implementation supports the UML XMI exchange mechanism for the UML packages extended by the Activity Profile.

New: A compliant implementation supports the UML XMI exchange mechanism for the UML packages extended by the Collaboration Profile for EAI.
A compliant implementation supports the UML XMI exchange mechanism for the UML packages extended by the Activity Profile for EAI.

Disposition: **Resolved**

OMG Issue No: 5350

Title: Compliance/metamodels: Clarify status of CAM

Source:

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

Summary:

```
Section 4.4 does not include the CAM metamodel - is this not normative?
```

Resolution:

Update Section 4.4

Revised Text:

```
TDLang and Type Descriptor models need to be included as normative
models.  IMS Transaction Message, IMS MFS, and CICS BMS models are non-
normative.
```

Disposition: **Resolved**

OMG Issue No: 5351

Title: Clarify relationship between EAI, FCM and ECA

Source:

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

Summary:

Section 2.3 of EDOC explains that architectures will be defined at the E/CCA level and thereby mapped to business processes etc. CCA components will then be 'mapped down' to various technology choices with FCM (and hence EAI) being one of them. EDOC contains only a proof of concept mapping for Business Process to FCM and not CCA to FCM.

This section also states that "Normative mappings from ECA to these models in the subject of future RFPs." It would seem that the current EAI RFP does not provide such a normative mapping (which I find disappointing though to be fair it was not a RFP requirement), and it should be made clear that this means one still has neither a development lifecycle nor a mechanism for either developing nor even recording the refinement from ECA (Enterprise/business architectures) to EIA technology. Just defining a correspondence between concepts or a means of representing EAI artefacts as CCA Components (6.5) does not achieve that. In particular it does not show how an arbitrary CCA design (possibly with defined constraints) can map to a EAI technology implementation. Without this, it is hard to evaluate the adequacy of the EAI proposal.

FCM "is a low-level metamodel focused on the middleware machinery for executing message flows. Higher levels of abstraction can be built upon the FCM for integrating a whole range of technologies and runtime environments:" (examples include Message Brokering). FCM allows the definition of hierarchic decompositions and the mapping of flows to FCMComponents. EAI actually extends FCM rather than creating a higher level of grouping/abstraction

Discussion:

The resolution to issue 4854 provides further detail on the relationship between CCA and EAI. The relationship between FCM and EAI is detailed in section 6.1

Changed text

See resolution to issue 4854

Disposition: **Resolved**

OMG Issue No: 5352

Title: Compliance: Consistency of statements about CAM compliance

Source:

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

Summary:

```
Section 5.1.2: states that the language metamodels in section 14 are
non-normative; however they are the basis of compliance points in
section 4!
```

Resolution:

Remove compliance points related to Section 15.

Revised Text:

```
Language models in section 14 are certainly normative.  What aren't
normative are the interface metamodels in section 15.
```

Disposition: **Resolved**

OMG Issue No: 5353

Title: CWM transformations

Source:

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

Summary:

```
Section 5.1.4: the spec does not permit the use of CWM transformations
due to the inconsistent way of modeling information resources
```

Resolution:

```
The purpose of the CWM model and CAM models are distinct and different.
They were not designed with integration of the two models in mind.
However, for the sake of integrating any existing non-normative CWM
COBOL models with CAM's normative COBOL model, a converged model of the
two COBOL models will be produced to ease the transition to CAM's COBOL
model.  The proposed integrated model will be non-normative and is
suggested as a temporary solution.
```

Revised Text:

Old Text:

> The transformation details are left to the implementation, and this includes the case where a transformation tool is based on XMI and the CWM.

New Text:

> The transformation details are left to the implementation, and this includes the case where a transformation tool is based on XMI and the CWM, which is an alternative to the use of CAM with different representations.

Disposition: **Resolved**

OMG Issue No: 5354

Title: Update reference to EDOC

Source:

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

Summary:

Section 5.1.5, Section 6: update to adopted version of EDOC (ad/01-08-19 for the convenience document including errata).

Resolution:

Update references

Revised Text:

Section 5.1.5, paragraph 2:
Before

(see OMG document ad/01-06-09,

After

(see OMG document ad/01-08-19,

Section 6.1, paragraph 2:

Change

ad/2001-06-09

To

ad/01-08-19

Disposition: Resolved

OMG Issue No: 5355

Title: MOF compliance

Source:

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

Summary:

Section 5.1.6: response re extending MOF doesn't address the requirement to conform to it (e.g. the metamodels should be MOF compliant - which they're not quite: they do not comply with the UML Profile for MOF in EDOC).

Resolution:

Update section 5.1.6.

Revised Text:

Old: No extensions to the OMG MOF are proposed.

New: No extensions to the OMG MOF are proposed. The EAI integration metamodel and the EAI Common Application Metamodel are based on MOF.

Disposition: **Resolved**

OMG Issue No: 5356

Title: IBM CWM products

Source:

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

Summary:

Section 5.4.3 para 2: why the mention of IBM CWM products?

Discussion:

Resolution:

Remove mention of IBM CWM product from text.

Disposition: **Resolved**

OMG Issue No: 5358

Title: Related activities: Relationship to ebXML and BPML

Source:

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

Summary:

Section 5.5: ebXML is at a different level to and encompasses many of the other B2Bstandards mentioned.
What's the relationship of EAI to BPML? And Workflow Process Definition?

Resolution:

Update section 5.5

Revised Text:

Old: Much trading is inherently event based, and so streams, messages, publications, sources, targets, filters, transformations and other operations are natural modeling elements for the intra-enterprise systems that are needed to support both internal and public electronic trading.

B-to-B modeling is dealt with in ebXML, which is based on a particular approach to B-to-B implementation. However, there are other approaches, including web services (SOAP, WSDL, UDDI, the draft web services flow language - WSFL - and XLANG) at W3C and OASIS, RosettaNet, OBI, EDI, OAG BODs and several industry-specific formats and protocols. There continues to be a high volume of activity and a rapid rate of change.

New: Much trading is inherently event based, and so streams, messages, publications, sources, targets, filters, transformations and other operations are natural modeling elements for the intra-enterprise systems that are needed to support both internal and public electronic trading. Hence, EAI is important both to inter and intra-enterprise business processes.

B-to-B modeling is dealt with in ebXML, which is based on a particular approach to B-to-B implementation. There are other specifications at differing levels, including web services (SOAP, WSDL, UDDI, BPEL4WS) at W3C and OASIS, RosettaNet, OBI, EDI, OAG BODs and several industry-specific formats and protocols. BPML is a rival to BPEL4WS, which can be used to specify workflow and other intra-enterprise processes as well as inter-enterprise processes. There continues to be a high volume of activity and a rapid rate of change.

Disposition: Resolved

OMG Issue No: 5359

Title: Use 'EAI' qualify references to profiles

Source:

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

Summary:

Use of the term 'UML Collaboration Profile' is misleading and too general and should include 'EAI' somewhere

Resolution:

Update sections 4.2, 4.3 and 8.1.1

Revised Text:

Old:

4.2 Compliance with the UML Collaboration Profile

The UML Collaboration Profile is defined in Chapter 8.

4.3 Compliance with the UML Activity Profile

The UML Activity Profile is defined in the Activity Modeling chapter.

In 8.1.1: The collaboration profile makes use of UML class and collaboration diagrams to notate EAI models.

New:

4.2 Compliance with the UML Collaboration Profile for EAI

The UML Collaboration Profile for EAI is defined in Chapter 8.

4.3 Compliance with the UML Activity Profile for EAI

The UML Activity Profile for EAI is defined in Chapter 9.

In 8.1.1: The collaboration profile for EAI makes use of UML class and collaboration diagrams to notate EAI models

Disposition: Resolved

OMG Issue No: 5366

Title: Wording of FCMSource description

Source:

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

Summary:

Section 6.2.5: To say that "an FCMSouce implements an FCMOperation" is a misreading of the model in Figure 2 (though not helped by the poor association end name) which should be read as "FCMOperation plays the implements role in its association with FCMsource" i.e. it is the FCMOperation that implements the FCMSource.

Resolution:

Accept the issue but not the proposed resolution. Update text as below.

Revised Text:

Original text:

> In a composite node (i.e., a node created from an FCMComposition) the interface offered is defined by the FCMSource and FCMSink nodes contained within the FCMComposition. An FCMSource implements (see Figure 62 on page63) an FCMOperation.

Replace with:

> In a composite node (i.e., a node created from an FCMComposition) the interface offered is defined by the FCMSource and FCMSink nodes contained within the FCMComposition. The operation offered by the composite is recorded by the 'implements' association between FCMSource and FCMOperation.

Disposition: Resolved

OMG Issue No: 5367

Title: Use UML profile for MOF <<enumeration>> stereotype

Source:

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

Summary:

Section 6.3.2 Figure 6-7. This should use the UML Profile for MOF (defined in EDOC) to depict the metamodel (i.e. EAISyncMode should have stereotype <> and each value should be depicted as an attribute).

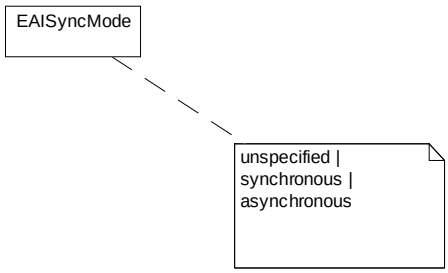
Resolution:

`Update figure.`

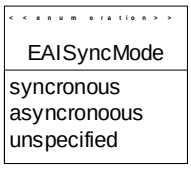
Revised Text:

Figure 6-7 (Fragment showing only) EAISyncMode

Before Change



After Change



Disposition: **Resolved**

OMG Issue No: 5368

Title: Clarify constraints on EAILink

Source:

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

Summary:

Section 6.3.2: Is an EAILink constrained to only link EAINodes?

Resolution:

Replace text of constraints section.

Revised Text:

Constraints

The source terminal of the EAILink is the same as the source terminal of its *controlLink*

context EAILink **inv**:

self.sourceTerminal = self.controlLink.targetTerminal

The target terminal of the EAILink is part of the *interface* of the *targetNode* of the controlLink

context EALink **inv**:

self.controlLink.targetNode.interface->exists(t | t=self.targetTerminal)

An EAILink connects two EAITerminals;

context EAILink

inv: self.sourceTerminal.oclIsKindOf(EAITerminal)

inv: self.targetTerminal.oclIsKindOf(EAITerminal)

An EAILink connects EAI operators, sources or sinks

context EAILink

inv: self.sourceNode.oclIsKindOf(EAIOperator) **or**

self.sourceNode.oclIsKindOf(EAISource)

inv: self.targetNode.oclIsKindOf(EAIOperator) **or**

self.targetNode.oclIsKindOf(EAISink)

Disposition: Resolved

OMG Issue No: 5369

Title: Constraints on EAITerminal

Source:

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

Summary:

Section 6.3.3: the third constraint in particular needs more description, especially to describe in terms of the metamodel the notion of a Terminal being on the 'exterior of a node'.

Resolution:

Clarify constraints.

Revised Text:

Replace existing text of section 6.3.3 as follows:

EAITerminal

EAITerminal

Definition

An EAITerminal is a specialization of FCMTerminal.

Constraints

EAITerminal can be connected to other instances of terminals only via instances of EAILink.

(any link that can have a source terminal which is an EAITerminal must be an EAILink, any link that can have a target terminal which is an EAITerminal must be an EAILink)

context FCMComposition

inv: self.connections->forall(c | if c.oclIsTypeOf(FCMTerminalToNodeLink) **then** c.sourceTerminal.oclIsKindOf(EAITerminal) **implies** c.oclIsKindOf(EAILink))

inv:self.connections->forall(c | **if** c.oclIsTypeOf(FCMTerminalToTerminalLink) **then** c.targetTerminal.oclIsKindOf(EAITerminal)

An EAITerminal is the representation (see Figure 66Default ¶ Font) of an FCMParameter that is of type EAIMessageContent.

Context EAITerminal

Inv: self.parameter.oclIIsKindOf(EAIMessageContent)

Disposition: **Resolved**

OMG Issue No: 5370

Title: Reword description of applicability of EAIMessageContent

Source:

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

Summary:

Section 6.3.4: reference to "MOM infrastructure" seems too technology-specific.

Resolution:

Revise Section 6.3.4

Revised Text:

Description of EAIMessagePart is moved to section 6.3.4.6 (see response to issue 5371). Change list item <1> to read:

> A message header, which contains metadata about the message rather than the application data . It is used to help determine required processing either by middleware or by metadata-aware applications.

Disposition: **Resolved**

OMG Issue No: 5371

Title: Clarify EAIParameter, EAIMessage

Source:

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

Summary:

Section 6.3.4 introduces a number of new classes (EAIParameter, EAIMessageContent etc) with no real description. (in particular the attributes of EAIMessageContent are a mystery).

Resolution:

Updates to Section 6.3.4

Revised Text:

Change title “6.3.4 EAIMessageContent” to “6.3.4 EAIMessageParameter”
Replace section 6.3.4 “Description” and “Constraints” with

Description

An EAIMessageParameter defines the data to be processed by an EAIOperation. It is associated with a single EAIMessage.

Change title **“6.3.4.1** EAIMessageElement Format Specification” to

“6.3.4.1 EAIMessageElement”

On Figure Diagram 6-9 change multiplicity of associationEnd ‘part’ from ‘1..n’ to ‘0..n’

(Comment: specification of message structure is stated to be optional (see Section 6.3.4.5 below)

Add sections:

6.3.4.5 EAIMessageContent

Description

Each message element (including the message header) conforms to a message format specification, which may be physically manifest in the message (as, for example, with an inline XML DTD) or may need to be inferred by the MOM infrastructure. In order to make this kind of distinction, EAIMessageContent has two properties;

domain which specifies the most generic message wireformat domain, and could be considered to encompass the domain of a generic parser. This is not restricted, but examples such as ‘XML’, ‘FixedFormat’, ‘Delimited’ would be valid

name within the domain specified above, this is the name of the message format to be processed. This information is intended to allow message format handling infrastructure to identify what type of message within a particular domain is being processed.

In addition to the basic attributes outlined above, EAIMessage may optionally specify further structure. It does this via an association with EAIMessagePart.

6.3.4.6 EAIMessagePart

EAIMessagePart may have two distinct elements:

A message header which contains metadata about the message rather than the application data itself. It is used to help determine processing either by middleware or by metadata-aware applications.

Message body, which contains the business content of the message.

The header and the body modeled via associations with EAIMessageElements.

6.3.4.7 EAIComposedMessagePart

EAIComoposedMessagePart is a subclass of EAIMessagePart which may itself contain messageparts.

Disposition: **Resolved**

OMG Issue No: 5372

Title: EAIMessagePart

Source:

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

Summary:

Section 6.3.4: It would make sense for EAIMessagePart.header to reference EAIHeader rather than its superclass EAIMessageElement: would it ever make sense for it to reference another type of EAIMEssageElement

Resolution:

The issue is a good one, but the suggestion is incorrect – not all headers need to subclass from EAIHeader, which is specifically provided to cope with replies and fault. Some headers (such as WS-Routing, etc) do not deal with these issues. Instead we insert explanatory text into EAIHeader.

Revised Text:

Inserted paragraph at the start of section '6.3.4.2 EAIHeader'

It is a common requirement for message processing to be able to specify a location to send any potential replies to, and to specify a location to which to send a message in the event of a message processing error. The information required to do this can be specified via a subclass of EAIHeader. In cases where the metadata contained in a header element does not concern replies or exceptions, it is not required for all headers in EAIMessageContent to be subclasses of EAIHeader.

Disposition: **Resolved**

OMG Issue No: 5373

Title: Constraints on EAIMessageElement

Source:

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

Summary:

Section 6.3.4: There should be constraints on the EAIMessageElements referred to by an EAIHeader (e.g. that they do allow navigation to Terminals and that they do not themselves have headers?) The derivations for the references to Terminals should be defined.

Resolution:

Association to EAITerminal has been removed (resolution to issue 4874), so the requirement to specify this derivation has gone away. Otherwise, I agree.

Revised Text:

Constaints

> The *exceptionTarget* and *replyTo* EAIMessageElement must not themselves be instances of the subclass EAIHeader
>
> **context** EAIHeader
>
> **inv:** replyTo->forall(rto | rto.oclIsKindOf(EAIHeader) = false)
>
> **inv:** exceptionTarget->forall(exc | exc.oclIsKindOf(EAIHeader) = false)

Disposition: Resolved

OMG Issue No: 5374

Title: How is EAIMessageContent.part used?

Source:

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

Summary:

Section 6.3.4 EAIExceptionNotice: Again refers to "MOM infrastructure". How if at all is the inherited reference EAIMessageContent.part used?

Resolution:

Update Section 6.3.4.3

Revised Text:

Replace 6.3.4.3 first sentence with;

> Messages of this form may be sent if an exception occurs during the processing of a message.

Add a new sentence at the end of the paragraph.

> In addition to these required message parts, the message may contain other message parts. These may be specified using the association to EAIMessagePart inherited from EAIMessageContent.

Disposition: **Resolved**

OMG Issue No: 5375

Title: Conflict with XML production of XML schema

Source:

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

Summary:

Section 6.3.4 XML Elements: XMI Production of XML Schema is now an adopted specification

Discussion:

It was agreed that this section was not intended to be normative, and that it should be removed to remove any potential for conflict.

Revised Text

<remove section 6.3.4 XML Message Elements>

Disposition: **Resolved**

OMG Issue No: 5376

Title: XML Message Elements

Source:

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

Summary:

Figure 6-12, though described as "showing a linkage" is in fact implicitly proposing a change to that specification through adding the new generalizations shown. This should

be made a lot clearer. IMO it is not an appropriate change since i t does not apply to other uses of that XML Schema metamodel and so EAI should introduce (one-way) associations instead of the generalizations that let (for example) a TDLangClassifier optionally refer to a XSDType.

Discussion:

See discussion for issue 5375.

Revised Text

See revised text for issue 5375

Disposition: **Resolved**

OMG Issue No: 5377

Title: Relationship to CWM XML Schema model

Source:

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

Summary:

Section 6.3.4 XML Elements: CWM also has a XML metamodel which might be more appropriate through its support for transformations. Justify not using it.

Discussion:

See discussion for issue 5375

Revised Text

See revised text for issue 5375

Disposition: **Resolved**

OMG Issue No: 5379

Title: EAIQueuedInputTerminal: Wording error on constraint

Source:

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

Summary:

Section 6.3.8, first constraint: an EAILink cannot be an instance of a Terminal. Presumably the target of the EAILinks must be instances of EAIQueuedInputTerminal. And there should be a similar constraint on "all links to an EAIQueuedInputTerminal"?

Resolution:

Insert the word ‘to’ between ‘be’ and ‘instances’

Revised Text:

Text before;

> All EAILinks from an EAIQueuedOutputTerminal must be instances of EAIQueuedInputTerminal.

Text after

> All EAILinks from an EAIQueuedOutputTerminal must be to instances of EAIQueuedInputTerminal.

Append the following at the end of section 6.3.8

context EAILink

inv: if self.sourceTerminal.oclIsKindOf(EAIQueuedOutputTerminal) **then**

self.targetTerminal.oclIsKindOf(EAIQueuedInputTerminal) **and**

self.synchronization=asynchronous **and**

self.sourceTerminal.targetQueues->includes(self.targetTerminal.inputQueue)

Disposition: Resolved

OMG Issue No: 5380

Title: Clarify the meaning of refinement relationships

Source:

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

Summary:

Section 6.3.9, refinement relationships: this is the first mention of refinement relationship and the topic needs some general introduction/context including how refinement is represented in the metamodel.

Resolution:

Revise Sections 6.3.9 and 6.3.2.

Revised Text:

Section 6.3.9, Refinement relationships
Before

Refinement relationships

An EAILink with synchronization of unspecified is refined by an EAILink with synchronization of either synchronous or asynchronous.

Where there is an instance of an EAILink with a synchronization of asynchronous linking a pair of FCMTerminals, this is refined by the substitution of EAIQueuedInputTerminal and EAIQueuedOutputTerminal for the FCMTerminals.

After

(text removed)

Section 6.3.2, Definition, paragraph 2

Before

Links may have their synchronization specified as synchronous, in which case a link between a pair of terminals implies a synchronous (call) invocation of the relevant FCMOperation, or asynchronous in which case a link between a pair of terminals implies an asynchronous invocation of the relevant FCMOperation (the FCMOperation which owns the parameter that the terminal represents).

After

Links may have their synchronization specified as

- **synchronous**, in which case a link between a pair of terminals implies a synchronous (call) invocation of the relevant FCMOperation;
- **asynchronous**, in which case a link between a pair of terminals implies an asynchronous invocation of the relevant FCMOperation;
- **unspecified**, in which case the invocation mechanism is left unspecified

Disposition: Resolved

OMG Issue No: 5381

Title: Operators: Wording change

Source:

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

Summary:

Section 6.3.10, para 2: should be "EAICompoundOperator".

Resolution:

Accept change precisely as worded.

Revised Text:

See above

Disposition: **Resolved**

OMG Issue No: 5382

Title: EAIPrimitiveOperator: Define derivations formally

Source:

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

Summary:

Section 6.3.10.1: the derivations should be more formally defined, especially "defines"; the semantics of this are also unclear (especially since EDOC does not describe FCMType).

Discussion:

The 'defines' association was erroneously labelled as derived in the original specification. The resolution to this issue is covered by the resolution to issue 4892.

Disposition: **Accepted**

Revised text

Covered by the revised text for issue 4892

OMG Issue No: 5383

Title: Relationship between EAIMessageFlow annotations and FCMComposition annotations

Source:

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

Summary:

Section 6.3.10.2.1: what's the difference between EAIMessageFlow.operatorAnnottions and the reference FCMComposition.annotations which it inherits?

Resolution:

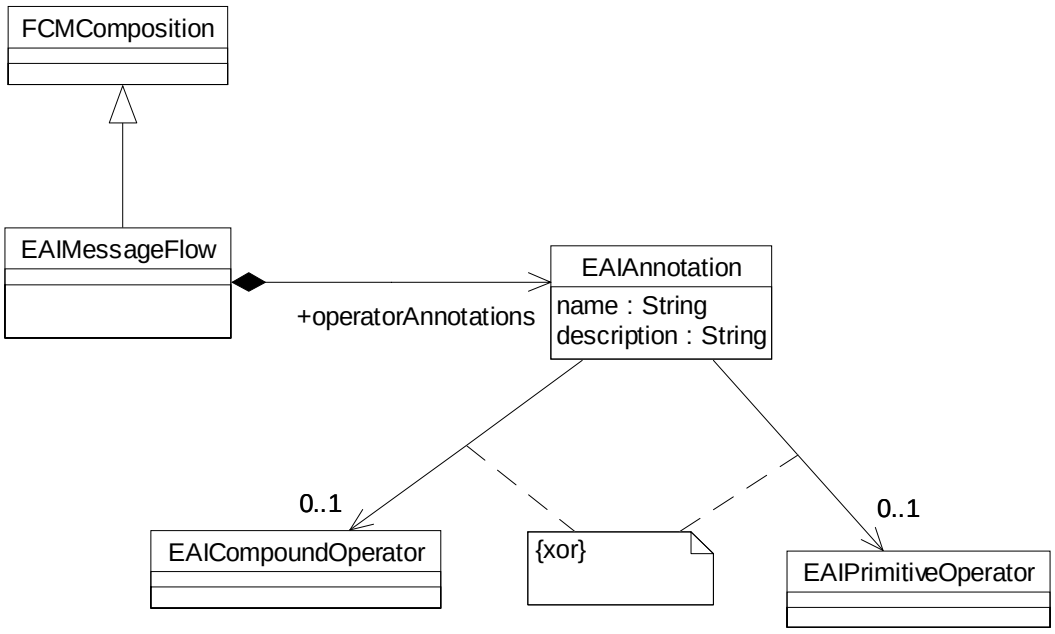
Explanation: EAIMessageFlow annotations are associated with EAI Operators, which are subclasses of FCMNode. FCMComposition annotations are associated with FCMComponent. Will remove inheritance from FCMAnnotation to prevent inheritance of the association to FCMComponent.

Revised Text:

Append sentence at the end of paragraph 1:

(this is in addition to the annotations associated with FCMComponent inherited from FCMComposition)

Updated Diagram



Disposition: **Resolved**

OMG Issue No: 5386

Title: Section 6.5.1.2, bottom p57

Source:

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

Summary:

Section 6.5.1.2, bottom p57: Implies that rather than EAI being just a low-level technology mapping for CCA, CCA components are required to provide the further detail of aspects such as transformations. Which means that EAI could be topped and tailed by CCA? A full example is needed.

Resolution:

This paragraph is just pointing out some potentials for specification of transformations. Technology mappings export whatever functionality is

relevant to each technology layer, as such it is reasonable that information from a CCA component may also map to EAI transformer implementations as is implied. The only change recommended is to include "EAI transformer implementations" in the list of possible implementation options.

Revised Text:

The transformation to be performed on the DataElement contents can be specified in a Property of the CCA ProcessComponent as an expression, script or transformation specification in any of the transformation languages available. Alternatively, the transformation can be delegated into usages of other technology-specific transformation processComponents in the internal Composition or into EAI transformer implementations.

Disposition: **Resolved**

OMG Issue No: 5387

Title: CAM: Introduce products in 'EAI' terms

Source:

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

Summary:

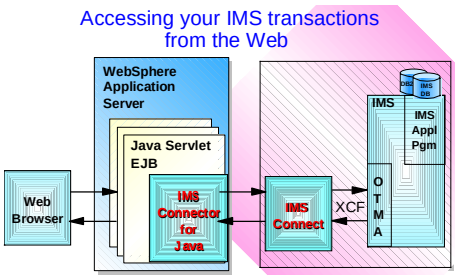
```
Section 7.2, last para: not clear without a clear understanding of the
various products and their role (in EAI terms). E.g. IMS Connect, OTMA.
It's also not clear how any connector built via the 'connector builder
tool' fits into the picture. A diagram might help
```

Resolution:

More text and a diagram in Section 7.2.

Revised Text:

IMS Connect and IMS OTMA are connector products that enable applications to interact with systems outside of the host machine. For example, IMS Connect allows IMS to exchange data with sources outside of S/390 environment over TCP/IP. IBM's WebSphere Application Developer Studio is an example of a 'connector builder tool.' Once the connector builder tool has generated a servlet and/or transformer code for the application, the code can be deployed on a web server such as IBM WebSphere Application Server to communicate with the backend application via connectors such as IMS Connect and IMS OTMA. Below is a picture to help explain.



Disposition: **Resolved**

OMG Issue No: 5389

Title: CAM Type descriptor metamodel: Introduce TDLangElement

Source:

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

Summary:

`Section 7.3.3/4: Uses TDLangElement without any introduction`

Resolution:

Swap order of presentation of TD and TDLang models.

Revised Text:

Interchange sections 7.3.3 and 7.3.4.

Disposition: **Resolved**

OMG Issue No: 5397

Title: Collaboration model: error in text associated with figure 8-1

Source:

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

Summary:

Section 8.2: the example does not have 2 input terminals as claimed

Resolution:

Change text in Section 8.2 to read “1 input terminal”

Revised Text:

1 input terminal

Disposition: **Resolved**

OMG Issue No: 5398

Title: Collaboration model: use UML operation specification

Source:

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

Summary:

P92 para 2: In UML, Operation already has a 'specification' property which should be used instead of attaching notes to the class.

Resolution:

Change text in section 8.1.1. to state that any definition of an operation used in operator specifications must be provided as part of the specification of that operation. If tools do not support the display of operations specifications on diagrams (as many don't) a UML note may be used in addition to repeat the definition on the diagram. Note that the specification of operations in examples used in this document will always be relayed by notes on the diagram.

Revised Text:

Disposition: **Resolved**

OMG Issue No: 5399

Title: Describe the required properties of terminal-operator associations

Source:

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

Summary:

Collaboration model: Describe the required properties of terminal-operator . Section 8: Does not describe the required (or otherwise) properties of the associations linking terminals and operators (multiplicity, navigability etc).

Resolution:

The associations are navigable only from operator to terminal, and have cardinality 1. These markings (which never change) may be omitted from the diagram (tool permitting) to avoid clutter. Any other properties are inconsistent with the profile.

Add text to this effect in Section 8.2

Revised Text:

Disposition: **Resolved**

OMG Issue No: 5401

Title: Explain underscores on names in collaboration diagrams

Source:

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

Summary:

Collaboration model: Explain underscores on names in collaboration diagrams. Fig 8-23: should explain the use of underscores at the start of names. And the use of the names to represent the values.
The figure seems to use names such as 'true' to the association ends being connected which should be explianed

Resolution:

Underscores are there because the tool used to create the diagrams did not allow objects on the same diagram to have the same name. There is already text explaining the naming scheme used (see text below Fig 8.23 and text in section 8.3.18.6, which provides the detailed constraints). This can be made clearer, perhaps, by making a forward reference to the latter from the former.

Revised Text:

Disposition: **Resolved**

OMG Issue No: 5402

Title: Collaboration model: Explain how terminals are wired together

Source:

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

Summary:

In general it's not clear how the terminals are wired together: what Association does the Link shown represent?

Resolution:

The UML 1.4. metamodel requires links to be connected to associations. In this case, the associations are redundant, but, of course, any UML tool strictly conforming to UML 1.4. should force the link to be associated with an association. To get around this, we propose that all EAI-UML models includes a class EAITerminal from which all Terminal classes inherit, which has an association to itself with cardinality 0..* on each end, and whose end names are left empty. All terminal to terminal links will be instances of this association.

This is only required when using tools that strictly enforce UML 1.4.

An explanation to this effect should be provided in section 8.3.18.2.

Revised Text:

Footnote: Underscores on names are used to ensure uniqueness, a requirement of the tool used.

Disposition: **Resolved**

OMG Issue No: 5405

Title: CAM Language Metamodels: Wording change

Source:

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

Summary:

```
Section 14: the metamodels start off by describing themselves in terms
of "The .. metamodel is a MOF Class instance at the M2 level". This
does not make sense. Possibly a MOF Model instance?
```

Resolution:

Update Section 14

Revised Text:

Update text to “Every CAM class is an instance of a MOF class at the M2 level."

Disposition: Resolved

OMG Issue No: 5409

Title: CAM: CsourceText clarification

Source:

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

Summary:

14.3.1.11: (CsourceText). The granularity of the text is not clear (e.g. a parameter or a CField can in theory have its own CsourceText instance).
Also the model has no obvious way of storing line numbers as claimed.

Resolution:

Update to 14.3.1.11

Revised Text:

The purpose of CSourceText is to provide the model creator with a place to store the entire source text under the "source" attribute in the class. Granularity of the content is decided by the granularity of the C source the modeler is generating.

Disposition: Resolved

Disposition: Unresolved

OMG Issue No: 4853

Title: Semantic information is poorly organized between Chapter 6 (EAI Integration Metamodel) and Chapter 8 (Collaboration Modeling)

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Chapter 6 describes the interchange metamodel and one would expect it to also provide the normative semantics of models constructed according to that metamodel. However, the specification of semantics is, in reality, spread between Chapter 6, Chapter 8 (which describes the collaboration modeling profile) and Chapter 9 (which describes the activity modeling profile). In practice, it is necessary to carefully read corresponding sections in both Chapters 6 and 8 (or 6 and 9) in order to understand the intended semantics. But, since the structure of the chapters is not parallel and since there are inconsistencies between the chapters [some of which will be identified in subsequent issues], the specification ends up being very difficult to use.

Recommendation: Structure Chapter 6 similarly to the specification of the UML 1.4 metamodel, but, perhaps, at a finer level of granularity. That is, for each major item (e.g., EAILink, EAITerminal, each kind of operator, etc.), organize the specification for that item under the following headings: o Metamodel: The metamodel diagram for the item. (Analogous to the UML metamodel "Abstract Syntax".) o Constraints: Textual and OCL descriptions of each of the applicable constraints. (Analogous to the UML metamodel "Well-Formedness Rules".) o Semantics: The COMPLETE specification of the semantics of the item. Chapter 8 should have a closely parallel structure to Chapter 6. For each major item, Chapter 8 should present: o Description of the profile notation/stereotypes and its mapping to the metamodel. o Textual and OCL descriptions of constraints associated with the stereotypes. (Note that, as part of the profile, these are constraints on the UML metamodel, as opposed to the constraints in Chapter 6, which are constraints on the EAI interchange metamodel.) o Descriptions of the mapping between the UML semantics and the metamodel semantics. Note that Chapter 8 should ONLY describe the MAPPING to the Chapter 6 metamodel and semantics, and not otherwise contain any normative semantics. (Similar comments also apply to Chapter 9, "Activity Modeling", and its relation to Chapter 6.)

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4860

Title: Errors in the FCM4EAI DTD

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

: There are errors (such as duplicate names and misspellings) in the FCM4EAI.dtd

Discussion:

Awaiting finalization of the rest of the issues

Disposition: **Unresolved**

OMG Issue No: 4873

Title: The "languageElement" association vs. the "message" association for EAIParamater

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

EAIParameter inherits a "languageElement" association with TDLangElement from FCMParameter (this association is part of the FCM specification and is shown in Figure 6-1). However, this association does seem not related in any way to the "message" association with EAIMessageContent. Indeed, an EAIMessageContent may be made up of several TDLangElements, so it is not clear which one of them might be considered to be "the" TDLangElement for the EAIParameter. This makes it unclear how the semantics of EAIParameter can be specialized from the FCM semantics for FCMParameter.

Recommendation:

Perhaps one could require that the languageElement for an EAIParameter to be, say, the languageElement of the body of the EAIMessagePart. But I don't think this really quite captures the right semantics (and, besides, this body is actually optional).

Instead, what is probably required is a change to the FCM to break the unfortunate cyclic dependency between the FCM (in the EDOC specification) and the CAM (in this specification). For instance, the FCM could define an abstract type descriptor class for the use as the type of an FCMParameter. TDLangElement could then be one possible descendant of this abstract type descriptor. But, for the purposes of EAIParameter, EAIMessageElement should also be a descendant, with the constraint that the type of an EAIParameter is always an EAIMessageElement (and the additional "message" association then being unnecessary).

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4959

Title: Unclear semantic description for EAIStream

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.2 (EAIStream) Description:
Section 6.4.1.2 states that the output behavior of an EAIStream is "abstracted via an 'emissionCondition' that determines under what circumstances a message is emitted from the stream." However, it is not clear exactly what this condition is really to be on or when it is invoked. Also, the next sentence says that "The message emitted may be _any_ element of the 'buffer'", but this is inconsistent with the previous paragraph, which states that "The streaming algorithm determines when to place messages from the _top_ of the buffer onto the 'out' terminal" (emphasis added). Recommendation: Define the semantics of EAIStream to be the following: When a message is received on the input terminal, the message is placed in the buffer at a place determined by the streaming algorithm associated with the EAIOperation invoked by the operator. The emissionCondition is then evaluated on the current state of the buffer. If the condition evaluates to true, then

the first ("top") element of the buffer is placed on the output terminal. Otherwise the operator produces no output (i.e., the output of the EAIOperation is "null").

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4960

Title: Lack of constraints on the terminals of an EAIStream

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.2 (EAIStream) Description: It seems to be implicit in the discussion of an EAIStream in Section 6.4.1.2 that there such an operator has a single input terminal and a single output terminal. However, this is not explicited stated anywhere in the section.

Recommendation: Add a constraint that "An EAIStrea m has a single input terminal that is an EAITerminal named 'in' and a single output terminal that is an EAITerminal named 'out'." (Note that this also makes the similar constraint in Section 6.4.1.3 on the terminals of an EAIPostDater redundant and unnecessary.)

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4961

Title: Missing multiplicity for the "emissionCondition" of an EAIStream

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.2 (EAIStream): Figure 6-27 does not show any multiplicity for the "emissionCondition" of an EAIStream.

Recommendation: Show a multiplicity of "1..1".

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4962

Title: Inclusion of the dynamic state "buffer" in the metamodel for EAIStream

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.2 (EAIStream): Figure 6-27 shows a "buffer" association on EAIStream. However, this is part of the dynamic state of an EAI stream operator, not part of the specification of the operator. An instance of EAIStream is a SPECIFICATION of an EAI stream operator, not the operator itself, and therefore should not include the dynamic state of the operator.

Recommendation: Remove the "buffer" association from Figure 6-27.

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 4963

Title: Unclear semantic description for EAIPostDater

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.3 (EAIPostDater): Section 6.4.1.3 states that an "EAIPostDater" holds a message in its buffer "until its individual timing condition is met". Does this mean that, at the appropriate time, the EAIPostDater autonomously emits the message, with no other stimulus, or that the operator checks the timing conditions each time it receives an incoming message? Also, as a child of EAIStream, EAIPostDater inherits the "emissionCondition" of a stream. How does this effect the behavior of the EAIPostDater?

Recommendation: Define the semantics of EAIPostDater to be the following:

> When a message is received on the input terminal, an EAIPostDater acts like an EAIStream in placing the message in its buffer and, possibly, immediately emitting a message. In addition, if the new incoming message is not the one that is immediately emitted, the EAIPostDater evaluates the timerMapping to create a timing condition for the message. Further, whenever any timing condition is met for any message in the buffer, the EAIPostDater autonomously places that message on its output terminal.

(An alternative would be to have EAIPostDater NOT be a child of EAIStream, with its semantics defined in a stand-alone fashion without an emission condition.)

Discussion:

Work in progress

Disposition: Unresolved

OMG Issue No: 4964

Title: Inclusion of the dynamic state "buffer" and "timingCondition" in the metamodel for EAIPostDater

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.3 (EAIPostDater): Figure 6-28 shows "buffer" and "timingCondition" associations on EAIPostDater. However, this is part of the dynamic state of an EAI post-dater operator, not part of the specification of the operator. An instance of EAIPostDater is a SPECIFICATION of an EAI post-dater operator, not the operator itself, and therefore should not include the dynamic state of the operator.

Recommendation: Remove the "buffer" and "timingCondition" associations from Figure 6-28.

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 5226

Title: The semantics of Stream operators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.4 (Streams) describes the semantics of stream operators as follows: "Messages that arrive from the input terminal do not get passed on, but instead are stored in a buffer or some other appropriate data structure. The emit operation defines the algorithm used to decide when and in what order messages get emitted to the output terminal. Abstractly, one can imagine a loop that continually calls the emit operation. It returns a message to be put on the output terminal at each call. There may be a delay between its being called and its returning a message."

Some issues with this description are:

1. The concept of "a loop that continually calls the emit operation" does not clearly seem to reflect the semantics described in Section 6.4.1.2 (see Issue 4959 on the lack of clarity of the semantics in that section) and does not reflect the underlying Flow Composition Model semantics of the metamodel. 2.

2. Section 8.6.2 states that, for an EAIStream, "12. The emissionCondition of the operator maps to the emit operation in the corresponding class". However, in Section 8.3.4, the emit operation is not a condition (which would return a Boolean) but, rather, has a message content return type. 3.

3. How is the "buffer or some other appropriate data structure" specified? Section 6.4.1.2 (EAIStream) shows the buffer as a set of EAIMessageContent, but this is not appropriate for the metamodel and should rather be addressed as a model-level concern (see Issue 4962).

Recommendation: Consistent with the recommendations given for Issues 4959 and 4962:

1. Require an "insert" operation that takes a single argument of the input content type. This operation maps to the "streaming algorithm" of the EAIOperation of the EAIStream and is triggered when a message arrives on the input terminal. It defines where the incoming message content is placed in the stream operator's buffer.

2. Define the emit operation to have a Boolean result. This operation maps to the emissionCondition of the EAIStream. The emit operation determines whether the top element of the buffer is emitted or not.

3. State that a buffer data structure for a stream class may be explicitly modeled, in order to provide more precise specification of the insert and emit operations. However, this model is considered part of the specification of the EAIOperation and EAICondition (emissionCondition) of the stream, and is not otherwise mapped explicitly into the EAI metamodel.

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 5227

Title: The semantics of Post Dater operators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.5 (Post Daters) states that "As the definition for 'emit' is fixed for post daters, only a definition for 'setTimingCondition' should be provided." It is not clear that this description is consistent with the description of the semantics for EAIPostDater in Section 6.4.1.3, since the metamodel still requires the specification of an emissionCondition (inherited from EAIStream). (See also Issue 4693, "Unclear semantics description for EAIPostDater".)

Recommendation: Either allow an emit operation on a post dater, to permit the possibility of immediate emission, or change the metamodel to not require an emissionCondition on an EAIPostDater. (See also the recommendation for Issue 4693.)

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 5230

Title: The semantics of Stream operators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 8.3.5 (Post Daters) Description: In the Section 6.4.1.3, the timing condition for a message received by an EAIPostDater is described as possibly entailing "a derivation from the content of the input message by a 'timerMapping'." This seems to indicate that a "timerMapping" is a mapping from a message (or message content) to a timing condition. However, while the "setTimingCondition" operation, which reflects the timerMapping, specified for a Post Dater class in Section 8.3.5 has a message content argument, it produces no result.

Recommendation: To correctly reflect the timerMapping, it would seem that the setTimingCondition operation should instead be "createTimingCondition", returning a TimingCondition. A model could also include a specification of exactly what a TimingCondition is, if this is necessary for precision of specification (though this would not be mapped to the metamodel, except as part of the specification of the FCMMapping that is the timerMapping).

Discussion:

Work in progress

Disposition: **Unresolved**

OMG Issue No: 5253

Title: Errors in the text of constraints on compound operators

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

In the first paragraph of Section 8.3.18.6 (Constraints [on Compound Operators]), "<<Compound>>" should be "<<CompoundOperator>>" and "cardinality" should be "multiplicity".

Discussion:

Awaiting resolutions in EDOC

Disposition: **Unresolved**

OMG Issue No: 5343

Title: Incorrect MOF files

Source:

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

Summary:

The XMI files are not MOF-compliant and incomplete. For example datatypes have no type codes, some datatypes (e.g. Boolean) are defined as classes, several associations are not contained in a package, several AssociationEnds have no Multiplicity.

Discussion:

Awaiting finalization of the rest of the issues

Disposition: **Unresolved**

OMG Issue No: 5403

Title: Collaboration model: MessageContent core

Source:

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

Summary:

8.5.1: Table 2 does not show the base classes for the stereotypes. Fig 93: <<LangElement>> is shown applied to both Attributes and Classes, which does not seem good practice.

Discussion:

Work in progress

Disposition: **Unresolved**

Disposition: Deferred

OMG Issue No: {issue No. here}

Title: {title of the issue}

Source:

{Company submitting issue, Name of individual, and e-mail of individual}

Summary:

{Summary of the issue}

Discussion:

{Summary of why the issue was deferred}

Disposition: **Deferred**

Disposition: Transferred

OMG Issue No: 4865

Title: Use of Derived Associations

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.2.1 (Motivation): The last paragraph of this section (beginning "Note that these derived associations...") is not entirely clear to me. I believe that the intent is the following: Derived associations are included in the metamodel and result in corresponding generated elements in the DTD. However, derived associations can always be computed from other information in the metamodel. Therefore, a tool would not necessarily need to store derived associations internally, though it would effectively have to compute them if it generated XML for interchange.

Recommendation: Reword the paragraph along the lines of what I wrote above. Also, be sure, to include the appropriate constraint for every derived association to define how it can be computed.

Discussion:

This issue (and others relating to FCM derived associations) is transferred to EDOC as EDOC Issues 5441, 5442, 5443 and 5444. These issues require an explanation of the FCM mechanism for recursive composition. The result of this is that the content of section 6.2 of the EAI specification is now covered by revisions to the FCM Model described in the 'UML Profile for EDOC'. Section 6.2 is retained, but its complete text is replaced with a brief description referencing the revised sections of the "UML Profile for for EDOC" (document number adxx-xx-xx.

Revised Text

<Section 6.2 to be replaced with the following text;>

6.2 FCM support for recursive composition
The UML profile for EDOC provides support for the definition of 'composite nodes', whose function is defined by an FCMComposition. The FCMNodes in an FCMComposition may themselves be conposite. Terminals on a composite FCMNode have an association with either an FCMSource or an FCMSink in the FCMComposition that defined an FCMCompositeNode. This is detailed in section x.x of the 'UML profile for EDOC' document number ad/xx-xx-xx.

Disposition: **Transferred**

OMG Issue No: 4866

Title: The implementingComposition derived association

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.2.3 (Composite nodes): The constraints seem to imply that the implementingComposition association is computed by navigating from FCMCommand to its "performedBy" FCMComponent, then from that to the "instanceOf" FCMType, then from that to an FCMCompositionBinding, and, finally, from that to the FCMComposition. Unfortunately, the association between an FCMCompositionBinding and an FCMType is unidirectional and not navigable from the FCMType back to the FCMCompositionBinding (see Figure 6-1). Further, there may be multiple FCMCompositionBindings for any FCMType (each FCMCompositionBinding is between one FCMType and one FCMComposition, but the model allows more than one binding), so it is not possible to identify a unique, single implementingComposition for an FCMCommand anyway. (Note that this problem becomes immediately apparent if you try to write the constraint in OCL.)

Recommendation: If you really want to require each FCMCommand to have an optional "implementingComposition", then I don't think this can be a derived association. And even if you want to constrain the "implementingComposition" to be selected from SOME relevant composition binding, then you need to provide the context for the set of composition bindings to search (or you could use FCMCompositionBindings.allInstances, but this is ugly).

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 4867

Title: The representation/parameter derived association

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.2.5 (EAITerminal) Description: The representation/parameter association is marked as <<derived>> in Figure 6-6, however no clear description is given on how it is derived.

Recommendation: I think the intent is that the terminal represents an FCMParameter of an operation associated with the FCMNode to which the terminal is attached. However, given the associations and navigabilities shown in Figure 6-2, this really cannot be done as a single constraint. Instead, it needs to be done as separate constraints on each kind of FCMNode:

FCMFunction: An FCMFunction has FCMTerminals that represent each of the parameters of the FCMOperation invoked by the FCMFunction.

> (self.interface->select(terminalKind = #in).parameter = self.invokes.inputs) and (self.interface->select(terminalKind = #out).parameter = self.invokes.outputs) and (self.interface->select(terminalKind = #fault).parameter = self.invokes.faults)

FCMSource: An FCMSource has a output terminals that represents the input parameters of the operation implemented by the FCMSource. (Note that a source has OUTPUT terminals, but these terminals represent the INPUT parameters wit hin the composition that implements the operation.)

> (self.interface->forAll(terminalKind = #out)) and (self.interface.parameter = self.implements.inputs)

FCMSink: An FCMSink has a single input terminal that represents a single output (or fault) parameter of the operation implemented by the FCMSource associated with the FCMSink. (Note that a source has an INPUT terminal, but this terminal represents an OUTPUT parameter within the composition that implements the operation.)

> (self.interface->size() = 1) and (self.interface.terminalKind = #in) and self.source.implements.outputs->union(self.source.implements.faults) ->includesAll(self.interface)

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5360

Title: FCM/Motivation

Source:

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

Summary:

Avoid introducing constraints on the FCM. Section 6.2.1 claims that no additional constraints are introduced to FCM. This is not true: such a constraint is introduced in 6.1.3 (it in effect constrains certain instances of FCMType to have only one FCMCompositionBinding, which is not required by FCM

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5361

Title: Why use FCMCommand?

Source:

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

Summary:

Section 6.2.1: the Motivation should state why FCMCommand has been chosen as the primary 'composite node' element from FCM (as opposed to FCMComponent for example which seems a more obvious match). According to EDOC "An FCMCommand is a special kind of FCMNode that represents the invocation of a particular FCMOperation on an FCMComponent. An FCMCommand can be thought of as being analogous to a programming language statement that invokes a method on an object".

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5362

Title: Wording of composite node description

Source:

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

Summary:

The initial statement "the composition method in the FCM is to construct an FCMCommand from an FCMComposition" would be better worded "the hierarchical composition method". FCM does not require that FCMCommands will themselves be defined through compositions.

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5363

Title: Composite nodes: Derivation of implementingComposition

Source:

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

Summary:

Figure 6-4: The diagram is misleading, and the actual derivation needed (which is hinted at under Constraint but should be more formally defined as in 6.2.4) would seem to rely on non-navigable references in the FCM metamodel. The new derived 'implementingComposition' association would not be based on the 'nodes' association shown between FCMComposition and FCMNode: this is already inherited by FCMCommand and shows where it is included into other 'larger' compositions. The new 'implementingComposition' reference to FCMComposition can, as far as I can see looking at FCM, only be derived from the following list of reference navigations: FCMCommand.performedBy (giving FCMComponent); FCMComponent.instanceOf (giving FCMType); FCMType.compositionBinding (giving FCMCompositionBinding - however this is not navigable!); FCMCompositionBinding.composition (finally giving FCMComposition).

An alternative route is to follow FCMCommand.invokes (giving FCMOperation); FCMOperation.type (giving FCMType though this is not navigable) and then navigating from FCMType as above. [One would hope that both navigation routes would give the same FCMType though this constraint is not documented in FCM, nor is any description provided there for FCMType!].

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5364

Title: Composite nodes and their contents

Source:

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

Summary:

Rename 'constraints' Section 6.2.4: the Constraints listed are not constraints but definitions of the derivation (which it is useful to have expressed).

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

OMG Issue No: 5365

Title: Define derived relationship between terminal and parameter

Source:

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

Summary:

Section 6.2.5: Should define the derived association via existing references.

Discussion:

See discussion for issue 4865

Disposition: **Transferred**

Disposition: Closed, no change

OMG Issue No: 4862

Title: EAIRouter output terminal type

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.6.1.7: EAIRouter has single output terminal that is connected to input terminals. This places a constraint on all connected input terminals to have the same type EAITerminal or EAIQueuedInputTerminal.

Discussion:

Proposed Resolution:

Reject. This limitation is inherent in the semantics of queued input and output terminals, which corresponds to communication mediated by a queue.

Disposition: Closed, no change

OMG Issue No: 4970

Title: Redundancy of EAIRouterUpdate/EAIBroadcaster with EAISubscriptionOperator

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section: 6.4.1.7.1 (EAIRouterUpdate and EAIBroadcaster): Particularly without the enclosing EAIRouter compound operator (see earlier issue on "The specification of EAIRouter and EAITimer as compound operators"), the EAIRouterUpdate/EAIBroadcaster operator pair update is pretty much redundant with the EAISubscriptionOperator/EAIPublicationOperator pair. Providing the simplified "subscription" model of EAIRouterUpdate does not seem worth the price of complicating what could be a very simple but still useful EAIBroadcaster concept.

Recommendation: Eliminate the EAIRouterUpdate operator and the concept of the EAIRoutingTable. Instead, define an EAIBroadcaster to simply be a primitive operator with a single input terminal and a single output terminal, with the semantics of copying

each message received at the input terminal to the output terminal. The EAIBroadcaster then provides a simple "hub" capability for providing fan-in/fan-out connection points in a message flow. (The name "EAIBroadcaster" is more appropriate than "EAIRouter" for this semantics.)

(Note that, if this recommendation is adopted, it makes moot the previous issue on "Inclusion of the dynamic state "routingTargets" for the EAIRoutingTable".)

Proposed Resolution:

Reject. This is a greater change to the submission than is necessary for finalization. If a modeler does not wish to use EAIRouterUpdate, it can simply be ignored. With the changes proposed in the resolution to Issue 4969, an EAIBroadcaster (renamed EAIRouter) can also be used, if desired, as a simple hub independently of EAIRouterUpdate.

Disposition: Closed, no change

OMG Issue No: 5342

Title: Incorrect filenames

Source:

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

Summary:

End Section 1.2: the filenames for DTD and XMI zip files are not correct

Discussion:

Linda Heaton of the OMG editorial staff confirms that the files are still at ad/2002-80-25 as shown in section 1.2.

Disposition: Closed, no change

OMG Issue No: 5347

Title: Compliance/Overview: use consistent XMI and MOF levels

Source:

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

Summary:

Section 4.1: XMI 1.2 is based on MOF 1.4 so to include it with MOF 1.3 and UML 1.3 is inconsistent.

Discussion:

XMI 2.1 does not explicitly reference MOF 1.3. The submitted XMI files are based on MOF 1.3. Hence, the submission is correct as it stands.

Disposition: **Closed, no change**

OMG Issue No: 5378

Title: EAIQueue: Show association with EAIMessage

Source:

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

Summary:

Section 6.3.7: I would expect Figure 15 to show the association with EAIMessage (which is needed to implement the constraint).

Discussion:

Reject. This is operational information.

Should we remove maxdepth?

Disposition: **Closed, no change**

OMG Issue No: 5391

Title: CAM InstanceTDBase: add a derived association

Source:

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

Summary:

```
Section 7.3.4.8: a 'parent' derived association should be defined to
encapsulate the navigation described. Similarly in 7.3.4.11
```

Discussion:

```
Navigation is already provided by the normative TDLang model classes.
```

Disposition: **Closed, no change**

OMG Issue No: 5400

Title: Use of containment in UML Collaboration Diagrams

Source:

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

Summary:

Collaboration model: Use of containment in UML Collaboration Diagrams. Section 8.3.18.2:
UML Collaboration diagrams do not have the notion of containment.
Also it's not clear how this notation, if supported, would map to the UML metamodel

Discussion:

UML does allow containment in Collaboration Diagrams. See, for example, p3.130 of the UML 1.4 spec. Rather containment is shown in the class diagram of Figure 85.

Disposition: Closed, no change

OMG Issue No: 5404

Title: Activity Model: Describe how this relates to the EDOC process profile

Source:

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

Summary:

Section 9.3.3 of the UML Profiles for EAI defines how the Activity Model Profile maps to the EAI Metamodel that is defined in section 6.

Section 6 defines how the EAI Metamodel is derived from the FCM of the UML Profile for EDOC.

Discussion:

I propose no changes to the document to repsond to this issue, since the document already answers the question.

I did consider whether the organizarion of the material in the document makes it more difficult than necessary to follow the chain necessary to answer the queston, but on reflection, i see no way to improve it without major disruption that would make the document less readable overall.

Disposition: Closed, no change

OMG Issue No: 5406

Title: CAM: COBOL Metamodel: Naming consistency

Source:

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

Summary:

Naming consistency: COBOLNumericType and COBOLNumericEditedType. Fig 14-1: Inconsistency e.g. COBOLNumericType.currencySymbol:char compared to COBOLNumericEditedType.currencySign:String
And random use of 'name' sometimes derived sometimes not.
It is not sensible to have VariableLengthArray as a subclass of FixedLengthArray (or vice versa - they are alternatives).

Discussion:

Naming consistency: COBOLNumericType and COBOLNumericEditedType. Fig 14-1: Inconsistency e.g. COBOLNumericType.currencySymbol:char compared to COBOLNumericEditedType.currencySign:String

Response: Resolved in issue 5239

And random use of 'name' sometimes derived sometimes not.

Response: Use of derived is not random. Attributes are only marked derived if they are inherited from a parent class.

It is not sensible to have VariableLengthArray as a subclass of FixedLengthArray (or vice versa - they are alternatives).

Response: Both fixed and variable length arrays share a maximum upper bound, while variable length arrays can have a minimum upper bound set by a Occurs Depending On clause such that the size of the array does not reach the maximum upper bound. Because both classes share the same maximum upper bound property on class, one can be considered a specialization of the other, i.e., a subclass.

Disposition: Closed, no change

Disposition: Duplicate/merged

OMG Issue No: 4864

Title: Lack of use of the MOF Profile

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

If the EAI Integration Metamodel is intended to be the basis for directly generating the FCM4EAI DTD (which it should be), then it should be presented with diagrams using the UML Profile for MOF, which was adopted as part of the UML for EDOC submission.

Disposition: **See issue 5367 for disposition**

OMG Issue No: 4884

Title: The "Refinement relationships" in the section on queued sources

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Section 6.3.9 (EAIQueuedSource and EAIQueuedSink):
There is a heading "Refinement relationships" in Section 6.3.9 that don't seem to have anything to do with queued sources and sinks (which is the topic ofthe section). Indeed, it is not clear what these statements are supposed to be about at all.

Recommendation: Remove these statements unless they can be clarified.

Disposition: **See issue 5380 for disposition**

OMG Issue No: 4977

Title: Missing message content class for timer conditions

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections: 6.4.1.10.1 (EAITimeSetOperator) Description: Section 6.4.1.10.1 states that an EAITimeSetOperator "processes a _message_ (EAIMessageTimerCondition)..."

(emphasis added). However, in Figure 6-43, EAIMessageTimerCondition is defined as a child of FCMCondition, not EAIMessageContent. Further, under the Constraints heading it is stated that "No more than one EAIMessageTimerCondition can apply to any single message in the timeSetConditions." But, as shown in Figure 6-42, the timeSetConditions are themselves EAIMessageTimerConditions, not messages, so it is not at clear what the constraint means. EAIMessageTimerCondition seems to be part of the dynamic state of a time-set operator, not its specification. What is needed instead really is a message format for representing a timer condition.

Recommendation: Replace the EAIMessageTimerCondition with an EAITimerConditionFormat class that is a child of EAIMessageContent and has "timerCondition" and "correlationCondition" associations with FCMCondition.

Disposition: **See issue 4976 for disposition**

OMG Issue No: 5245

Title: Adapters are called Operators in the profile but not in the metamodel

Source:

InteliData Technologies Corporation (Mr. Ed Seidewitz, eseidewitz@intelidata.com)

Summary:

Sections 8.3.6 to 8.3.9 describe adapters as kinds of operators. However, in Sections 6.3.11.1 through 6.3.11.4, the corresponding metamodel elements are NOT defined as subclasses of EAIOperator, but rather are directly subclasses of FCMFunction. (Actually, in Section 6.3.11.4, EAIRequestReplyAdapter actually is diagrammed as a subclass of EAIPrimitiveOperator, but it is described in the text as being a subclass of FCMCommand and should probably really be a subclass of FCMFunction like the other adapters -- see Issue 4859). This would seem to indicate that adapters are NOT operators, since not all their terminals are EAITerminals.

Issue Raiser's Recommendation:

Do not describe adapters as operators. Move the description of adapters out of Section 8.3 on operators

Disposition: **See issue 5247 for disposition**

OMG Issue No: 5351

Title: Clarify relationship between EAI, FCM and ECA

Source:

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

Summary:

Section 2.3 of EDOC explains that architectures will be defined at the E/CCA level and thereby mapped to business processes etc. CCA components will then be 'mapped down' to various technology choices with FCM (and hence EAI) being one of them. EDOC contains only a proof of concept mapping for Business Process to FCM and not CCA to FCM.

This section also states that "Normative mappings from ECA to these models in the subject of future RFPs." It would seem that the current EAI RFP does not provide such a normative mapping (which I find disappointing though to be fair it was not a RFP requirement), and it should be made clear that this means one still has neither a development lifecycle nor a mechanism for either developing nor even recording the refinement from ECA (Enterprise/business architectures) to EIA technology. Just defining a correspondence between concepts or a means of representing EAI artefacts as CCA Components (6.5) does not achieve that. In particular it does not show how an arbitrary CCA design (possibly with defined constraints) can map to a EAI technology implementation. Without this, it is hard to evaluate the adequacy of the EAI proposal.

FCM "is a low-level metamodel focused on the middleware machinery for executing message flows. Higher levels of abstraction can be built upon the FCM for integrating a whole range of technologies and runtime environments:" (examples include Message Brokering). FCM allows the definition of hierarchic decompositions and the mapping of flows to FCMComponents. EAI actually extends FCM rather than creating a higher level of grouping/abstraction

Discussion:

The resolution to issue 4854 provides further detail on the relationship between CCA and EAI. The relationship between FCM and EAI is detailed in section 6.1

Disposition: See issue 4854 for disposition

OMG Issue No: 5357

Title: CAM/CWM alignment

Source:

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

Summary:

```
Does not allow the use of CWM transformations.  Inconsistent use of
data vs programming constructs (e.g. for C++) operations etc.
```

Section 5.4.3 while comprehensive is not at all convincing and smacks of NIH. What is someone wanting to manage the mapping of EAI to databases supposed to do? These are not at all isolated universes. If CAM and CWM are both needed to meet different perspectives then there should be a mapping and moreover a common core. It's like saying in a UML context that there should be no relationship between state charts and class diagrams since they address different perspectives.

In fact data warehousing is just an example of application integration, and CWM even supports event-based communication (in the Warehouse Process submodel).

Disposition: **See issue 5353 for disposition**

OMG Issue No: 5384

Title: Derivation of promoted terminal

Source:

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

Summary:

Section 6.3.10.2.2: this really needs an example to help explain this and detail for the derivation of promotedTerminal.

Discussion:

Disposition:

Duplicate of 4897

Revised Text
<Remove Section 6.3.10.2.2>

OMG Issue No: 5385

Title: What is a 'CCA Component Library'?

Source:

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

Summary:

What is a 'CCA Component Library'? This is not described in EDOC What is the motivation for this? How is the mapping formally represented? Why the different concepts? When would one define compositions via CCA and when via FCM/EAI Integration? Can CCA be thought of as a higher level architectural

view on the FCM/EAI Integration model? If so is that not more important for the RFP scope?

Disposition: **See issue 4854 for disposition**

OMG Issue No: 5388

Title: CAM Type Descriptor Stereotypes

Source:

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

Summary:

`Figure 7-5: <<Enumeration>> stereotype missing from two of the classes.`

Disposition: **See issue 5237 for disposition**

OMG Issue No: 5390

Title: CAM Type descriptor stereotypes: Heading change

Source:

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

Summary:

Section 7.3.4.13 The heading is wrong - it should refer to 'enumerated types' not 'stereotypes'.

Disposition: **See issue 5237 for disposition**

OMG Issue No: 5392

Title: CAM Type descriptor formulas

Source:

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

Summary:

`Section 7.3.5, p7-12: need to define "level-1 data structure" and "level-1 parent".`

Disposition: **See issue 5237 for disposition**

OMG Issue No: 5393

Title: CAM TDLang Metamodel diagram changes

Source:

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

Summary:

Figure 7-6: the associations are shown as derived (the '/') which is not correct.
Figure 7-6: the composition should be shown as {ordered}.

Disposition: **See issue 5237 for disposition**

OMG Issue No: 5394

Title: CAM TDLangModelElement: Classifier or Element

Source:

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

Summary:

Section 7.3.8.4: Each instance of TDModelElement will represent EITHER a Classifier or an Element - not a combination (though an Element will in turn refer to its Classifier).

I think more explanation/example is needed for the difference between TDClassifier and TDLangElement (which does not have any concrete examples) and why mappings are not made at the Classifier as opposed to the Element level.

Disposition: **See issue 5237 for disposition**

OMG Issue No: 5395

Title: CAM: Title of section 7.3.9

Source:

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

Summary:

Section 7.3.9: it's confusing to imply this is a separate metamodel - it just describes how the 2 previous metamodels are used together

Disposition: **See issue 5243 for disposition**

OMG Issue No: 5396

Title: CAM: Sample serialisation: Problems with XMI

Source:

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

Summary:

Section 7.3.11: Lots of problems with the XMI: it is not valid for showing the relationship between the TDLangElements and the TDLangClassifiers (in fact the XMI represents no relationship at all between them!): also it's wrong to show the SimpleInstanceTDs nested within the COBOLComposedType, since there is no Composition (in fact no direct relationship at all in the metamodel) between them. Also defaultFloatType is not an attribute of SimpleInstanceTD but of PlatformCompilerInfo. And the SimpleInstanceTDs do not have the mandatory 'sharedType' reference, which would have been useful to see expressed, and none of the TDs have the mandatory 'platformInfo' reference.

Typos: the COBOLComposedType element is incorrectly terminated on the first line (just need to remove the "/") and 'AggregateInstanceTDBase' should be just 'AggregateInstanceTD'

The XMI should, I believe, be as below. An instance diagram would help undestanding!:

```
   <COBOLElement xmi.id='CE-1'name="NAME" instanceTDBase='AIT-1'
tdLangSharedType='CCT-1'/>
   <COBOLComposedType xmi.id='CCT-1'>
      <TDLangComposedType.tdLangElement>
           <COBOLElement xmi.id='CE-2' name="FIRST"
instanceTDBase='SIT-1'
tdLangSharedType='CT-1'/>
           <COBOLElement xmi.id='CE-3' name="LAST" instanceTDBase='SIT-
2'
tdLangSharedType='CT-1'/>
      </TDLangComposedType.tdLangElement>
   </COBOLComposedType>
   <COBOLAlphaNumericType xmi.id='CT-1' name="PICX10"
pictureString="PIC X10"/>

   <AggregateInstanceTD xmi.id='AI-1' languageInstance='CE-1'
platformInfo='PC'/>
   <SimpleInstanceTD xmi.id='SIT-1' languageInstance='CE-2'/
sharedType='ST-1' platformInfo='PC'/>
   <SimpleInstanceTD xmi.id='SIT-2' languageInstance='CE-3'
sharedType='ST-1' platformInfo='PC'/>
   <StringTD xmi.id='ST-1' nickname='COBOL PIC X10' width=10
addrUnit=byte
encoding='ASCII'….>
   <PlatformCompilerInfo xmi.id='PC' …../>
```

The above shows no top-level container. This lack of a packaging structure seems to be an omission from the metamodel.

Disposition: **See issue 5244 for disposition**

OMG Issue No: 5407

Title: CAM: C Derivation diagram

Source:

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

Summary:

Fig 14-9: Cderived has 2 references 'derives'. In any case it is not a sensible role name.

Disposition: **See issue 5240 for disposition**

OMG Issue No: 5408

Title: CAM: C User Types

Source:

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

Summary:

Fig 14-12: CunsignedLong should not be a subtype of CunsignedInt since it's a larger set: it should inherit from Clong. Likewise CunsignedLongLong should inherit from ClongLong.
Moreover Long should not inherit from Cint and CWChar not from CCHar. Finally it's not clear what the dependency arrows mean.
Response: Whether unsigned datatypes should inherit from their signed counterpart or from unsigned datatypes is arbitrary. The current model allows the unsigned property of all numbers to be shared under CUnsignedInt. Dependency issue resolved in issue 5240

Disposition: **See issue 5240 for disposition**