Annex 1 – Project Details

The E-Commerce Integration Meta-Framework

1. The language

The ECIMF initiative proposes the use of UML-like modeling language to express relationships between the semantics and models of the e-commerce frameworks. This E-Commerce Integration Modeling Language ("ECIML"), to be defined as a result of the project, would be a concrete instance of the OMG's MOF meta-meta-model, at the same time re-using as many concepts from standard UML as possible. This puts it in the following relationship to the standard modeling approaches:

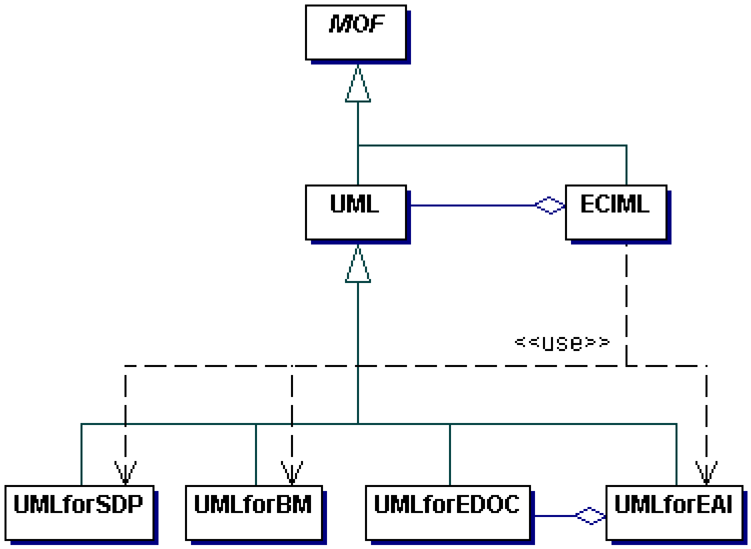


Figure 1 Relationship between the ECIML and other modeling standards.

We will build on the experiences of the projects like pUML (The Precise UML Group), using also the OMG's standards (e.g. CWM, standard UML 1.4 profiles, UML Profile for EAI and UML Profile for EDOC) when appropriate, in order to define a suitable meta-model.

One could use the standard UML for modeling these concepts, but we feel that in its current form it's too generic and lacks necessary precision, and though it's extensible, the way the extensions are specified is often implicit (e.g. stereotyping). In the ECIML meta-model they would be precisely defined. Some of these issues will be addressed in the next major revision of UML standard (2.0), at which point we will evaluate the possibility to use that standard as the sole basis for ECIML.

Consequently, one of the goals of this project will be to define a suitable set of modeling constructs to more adequately address the needs of meta-framework modeling and transformations.

2. The methodology

The proposed methodology for analysis and modeling of the transformations between the e-commerce frameworks follows the layered classification approach.

This approach means that in order to analyze the problem domain one has to split it into layers of abstraction, applying top-down technique to classify the entities and their mutual relationships:

- First, to identify the top-level entities and the contexts in which they occur.
- Then, to proceed to the next layer in which the interactions between the entities are analyzed.
- Then, to go to the lowest, the most detailed level to analyze the messages and data elements in communication between the entities.

Starting from the top-most level, the contexts in which the interactions occur are analyzed and collected, and these contexts affect the semantics of the interactions occurring at the lower layers.

The second dimension of the proposed approach conforms to the Meta-Model Architectures, as described in the MOF standard, introducing the meta-model, model and data layers.

The example classification layers are presented in the following picture, where the vertical dimension is the methodology abstraction layers, and the horizontal dimension is the model abstraction layers:

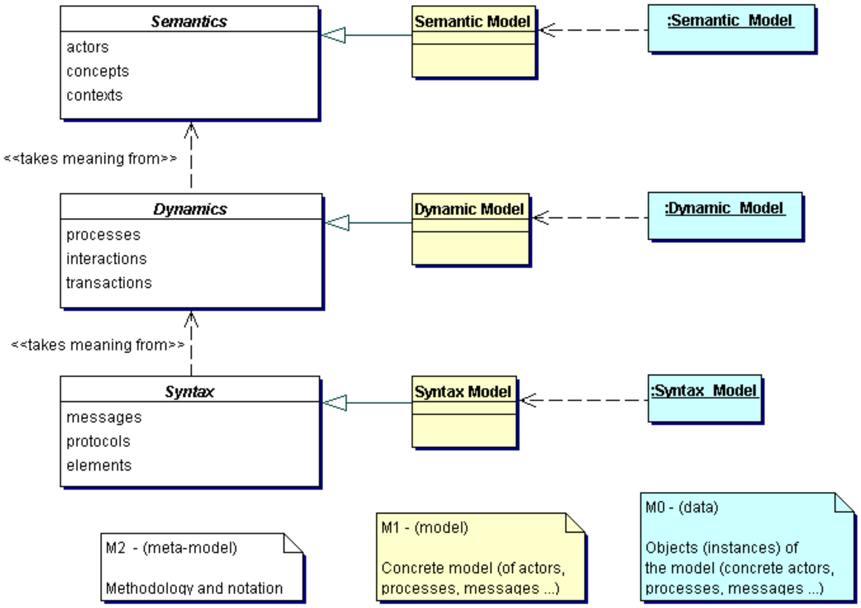


Figure 2 ECIMF methodology and the meta-model architecture.

In order to navigate through the framework models and concepts, a prototype tool named Conzilla is introduced, which in later stages will be augmented with other modules (like data format translating software, automatic generation of interfacing state machines, routing and packaging translators, etc).

The project will define a recommended methodology (named E-Commerce Integration Modeling Methodology – “ECIMM”) and base tools needed to prepare specific comparisons of concrete frameworks, which in the end should result in clear implementation guidelines for system integrators and software vendors on how to ensure interoperability and semantic alignment. This generic integration meta-framework will be expressed in the ECIML language, providing mapping and transformation descriptions/recipes that can be implemented by an ECIML-compliant agents/intermediaries. This ultimately should allow the frameworks to interoperate without extensive manual alignment by the framework experts.

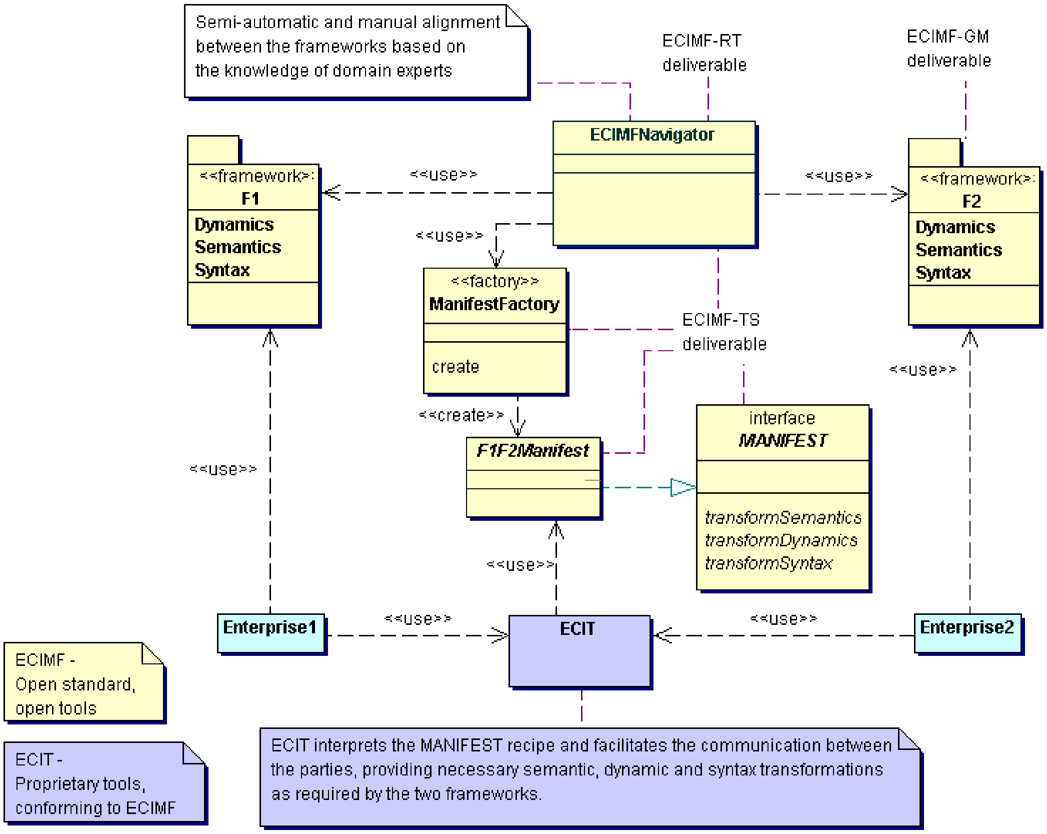

Semi-automatic and manual alignment between the frameworks based on the knowledge of domain experts
ECIMF-RT deliverable
ECIMF-GM deliverable
ECIMFNavigator
<<use>>
<<use>>
<<framework>> F1
Dynamics
Semantics
Syntax
<<framework>> F2
Dynamics
Semantics
Syntax
<<use>>
<<factory>> ManifestFactory
create
ECIMF-TS deliverable
<<use>>
<<create>>
F1F2Manifest
interface MANIFEST
transformSemantics
transformDynamics
transformSyntax
<<use>>
<<use>>
Enterprise1
<<use>>
ECIT
<<use>>
Enterprise2
ECIMF - Open standard, open tools
ECIT - Proprietary tools, conforming to ECIMF
ECIT interprets the MANIFEST recipe and facilitates the communication between the parties, providing necessary semantic, dynamic and syntax transformations as required by the two frameworks.

Figure 3 The ECIMF concept of frameworks transformation and alignment.

The meta-framework definitions/recipes for interoperability are named “MANIFEST”. The language to be used in these definitions will be called E-Commerce Integration Modeling Language (“ECIML”), and will be based on XML representation of UML-like meta-models, rules and definitions.

The following diagram describes how the ECIMF approach is used in order to align the two different frameworks:

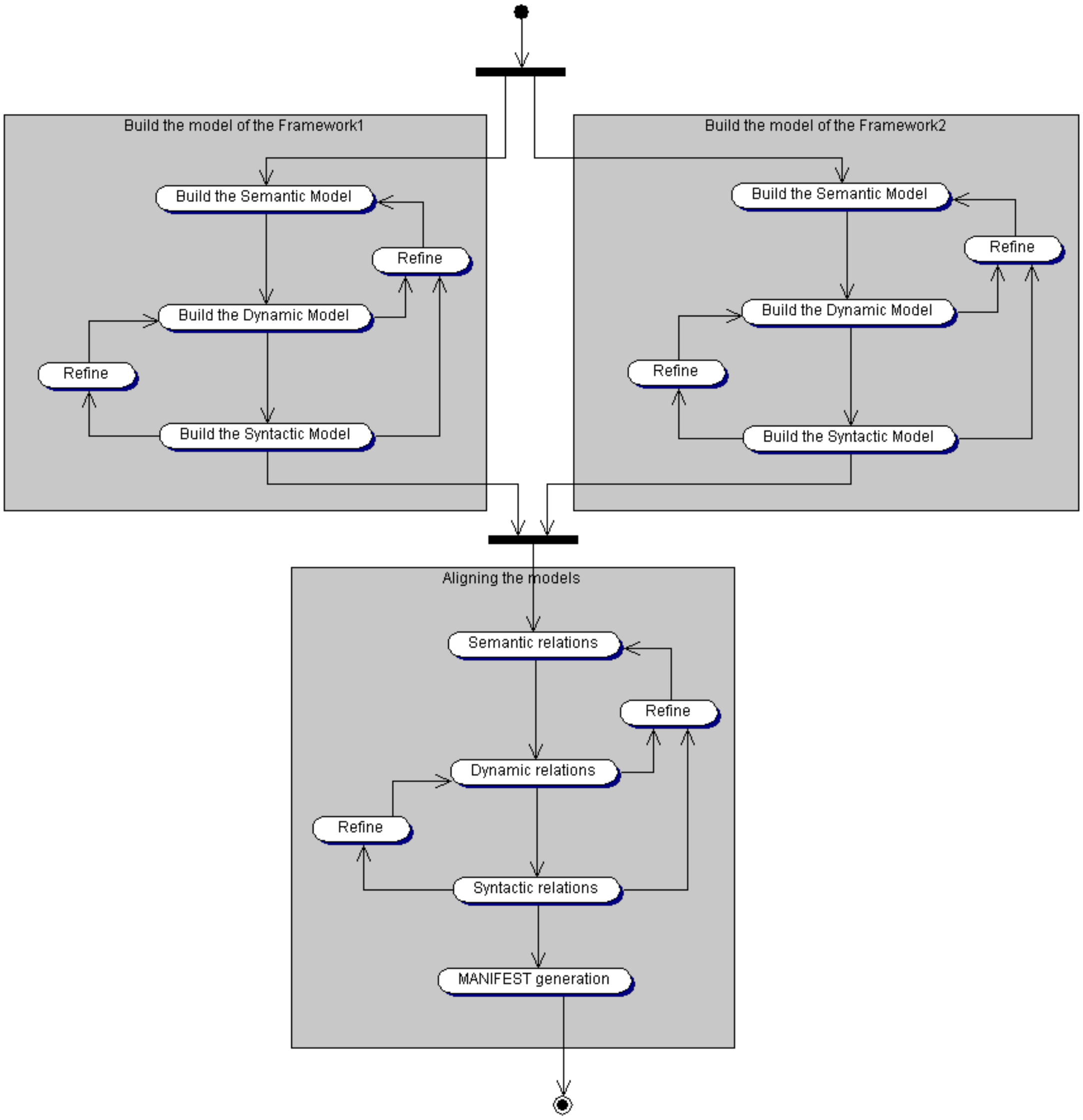


Figure 4 The process of modeling and alignment between two e-commerce frameworks.

3. MANIFEST recipes

A MANIFEST recipe described with ECIML will be identified by a unique ID, and stored in the repository from which an ECIML-compliant agent can retrieve it. The agent, based on the transformations specified in the MANIFEST recipe, will create necessary processing structures to align the message handling and interactions between the agents belonging to different frameworks. It is expected that the repository will be able to also store commonly used templates for inter-framework alignment, so that less experienced or knowledgeable users can leverage the accumulated expertise of framework experts, and by making relatively minor adjustments re-use the templates as their own MANIFEST recipes.

The specifics of the repository need to be further discussed. Initially we suggest possibility of using either ebXML or UDDI to store the MANIFEST recipes.

It is yet to be defined what kind of language will be used to describe the transformations between the models. The following is a short list of the requirements that need to be satisfied:

- Preferably Open Source implementations available
- Highly portable
- Well-known: this is needed in order to ease the adoption
- Strongly typed: the transformations need to be precisely defined, and it's preferred that most logical errors would be discovered during the parsing/compilation, not at the runtime.
- High level (additional tools for manipulation of complex programmatic structures, database and directory access, etc…)

The candidates that we consider at this stage are Java, XSLT and Python.

4. The Toolkit

The intention of the E-Commerce Integration Toolkit ("ECIT") is to offer a simplified and affordable solution to conform to the existing and upcoming standards without the burden of having to know all the complex technologies behind them.

We will aim to provide a simple implementation of the ECIT and make it available on an Open Source basis. However, in order to fully leverage the ECIMF approach, we expect the software vendors to follow our initiative and provide complete implementations as proprietary products – still, compatible with the open standard.

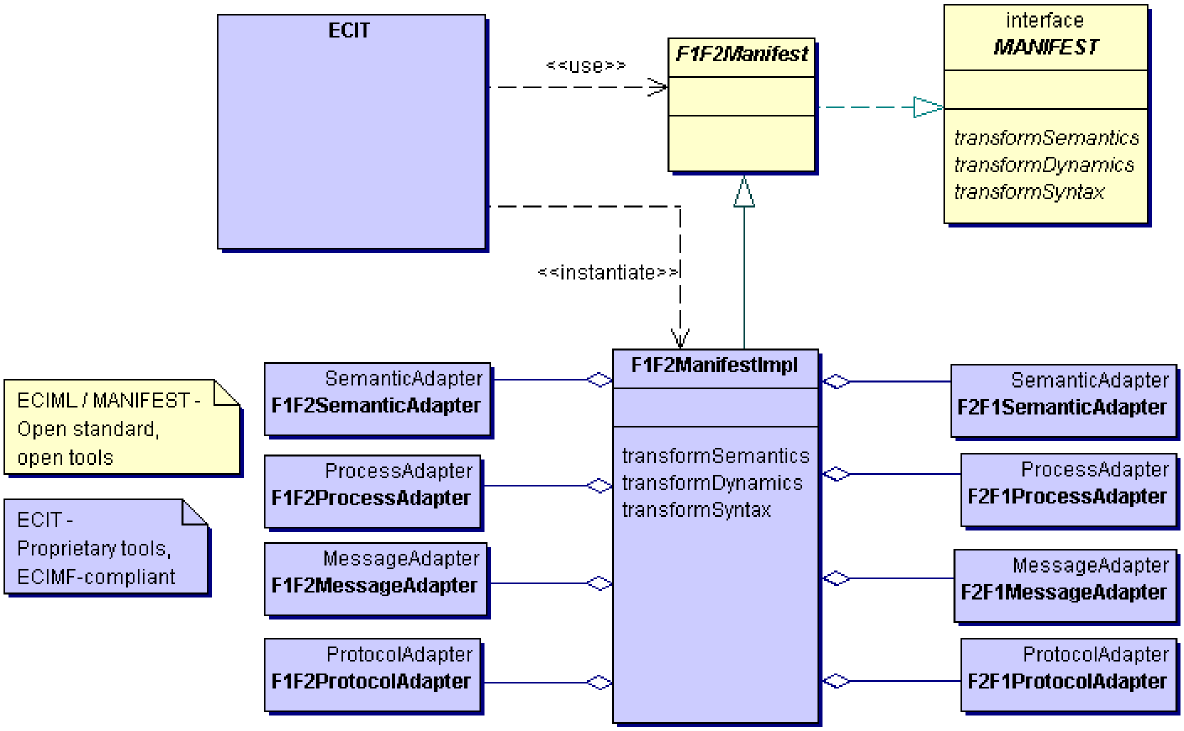


Figure 5 Example of ECIT (ECIML-compliant agent) facilitating message exchange.

5. Example

This example presents step by step how a meta-framework recipe for interoperability could be prepared, between hypothetical e-commerce frameworks Framework1 and Framework2.

Note: the diagrams have been prepared using a generally available UML modeling tool. Some of the concepts could not be presented appropriately (e.g. lack of notation constructs, or wrong constraints applied).

First, a formal model of both frameworks needs to be built based on the available models, natural language descriptions and domain expert knowledge of the frameworks. This model is built using the ECIMF approach. The scope of the model depends on the scope of the integration task at hand, i.e. it doesn't necessarily have to be a complete model. However, the modeling and the analysis follow the structured, layered approach:

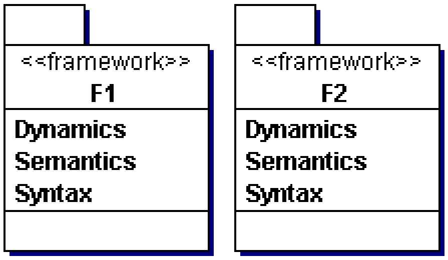


Figure 6 Modeling the frameworks

Then, using the ECIMF Navigator or a similar tool, the framework experts calibrate and align the concepts common to both frameworks.

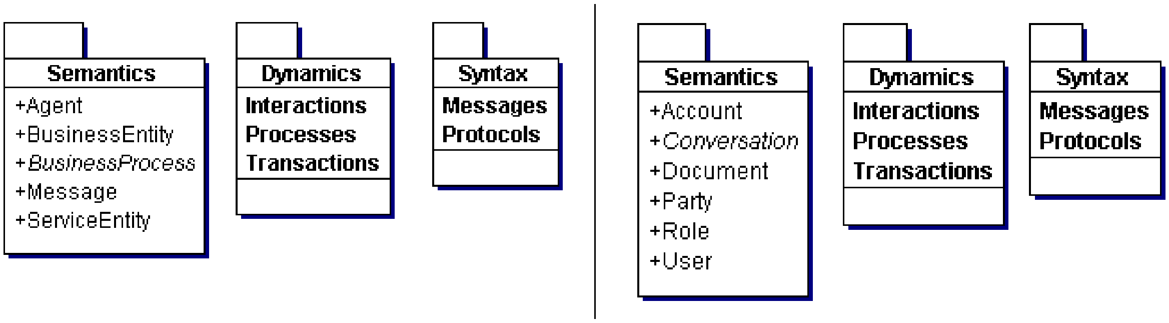


Figure 7 The top-most layers of the Framework1 and Framework2 models.

Let's look closer at this example. The figure 8 presents the Semantics elements of both frameworks in a more detailed fashion. We notice several similarities here. They are marked in the following pictures using the same colors and stereotypes for the corresponding concepts:

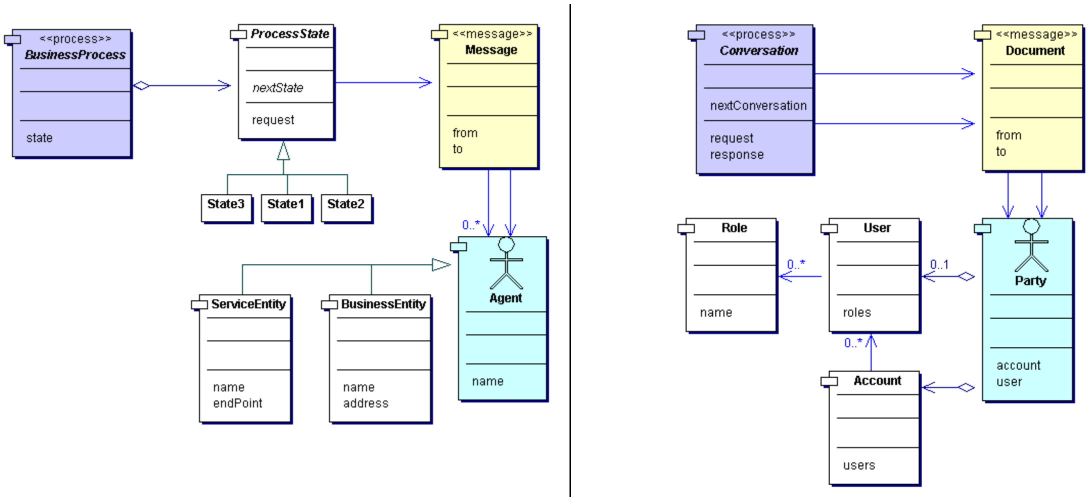


Figure 8 Comparing the corresponding semantic elements.

This is an important step that will affect many other modeling decisions during later stages. The ability to find the corresponding concepts is the basic premise for any successful attempt at interoperability.

When using the ECIMF Navigator tool, we could imagine this step to look like the following figure:

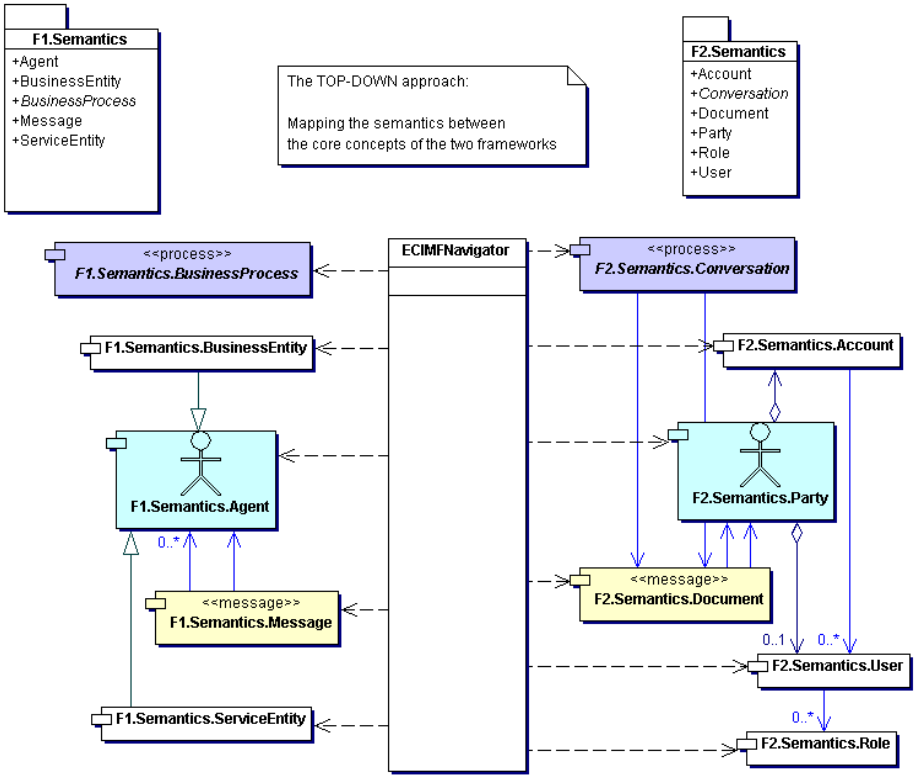
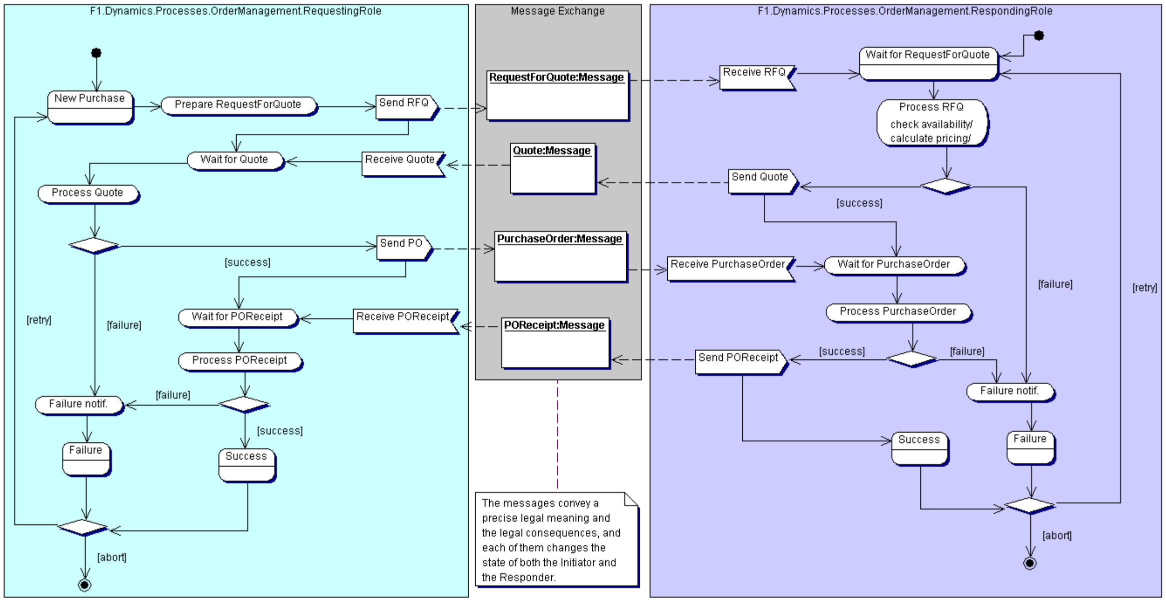


Figure 9 The ECIMF Navigator compares the semantic elements of the frameworks.

Then the modeling process proceeds to the next layer, where the framework integrator concentrates on the specific business scenarios that need to be integrated.

So, in the first step the framework integrator prepares a formal model of activities for e.g. Order Management business process for the Framework1. This is presented in the Figure 10. We use here the standard UML Activity Diagram notation, as it has been found to be flexible enough (see Annex 2 for comparative study of the notations).



The messages convey a precise legal meaning and the legal consequences, and each of them changes the state of both the Initiator and the Responder.

Figure 10 Framework1 business process of OrderManagement.

Then, using similar approach, the system integrator models the corresponding OrderManagement process in the Framework2 that leads to the same business consequences as the one in Framework1.

As the following picture shows, that process is different from the corresponding process in Framework1. The result is presented in Figure 11.

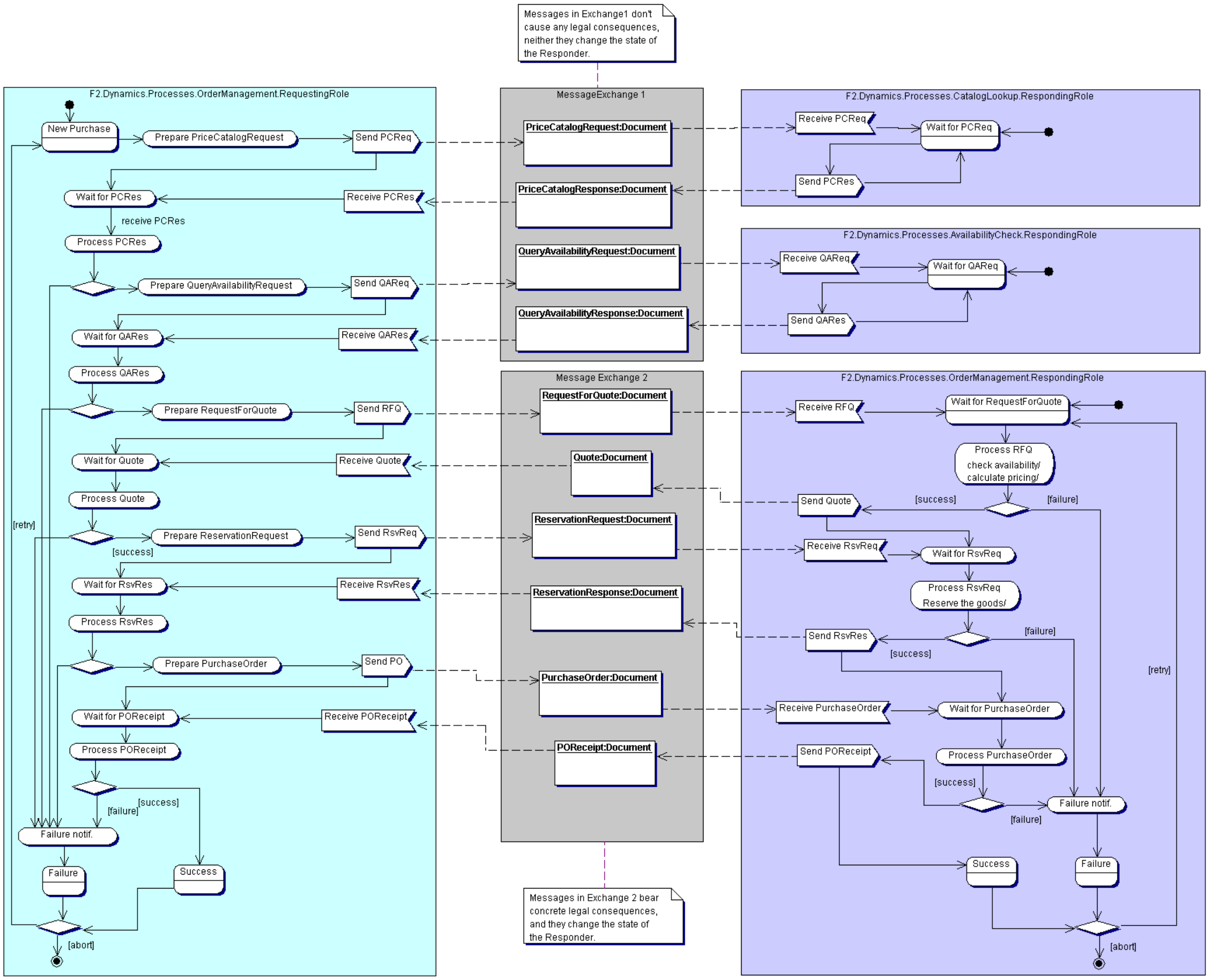
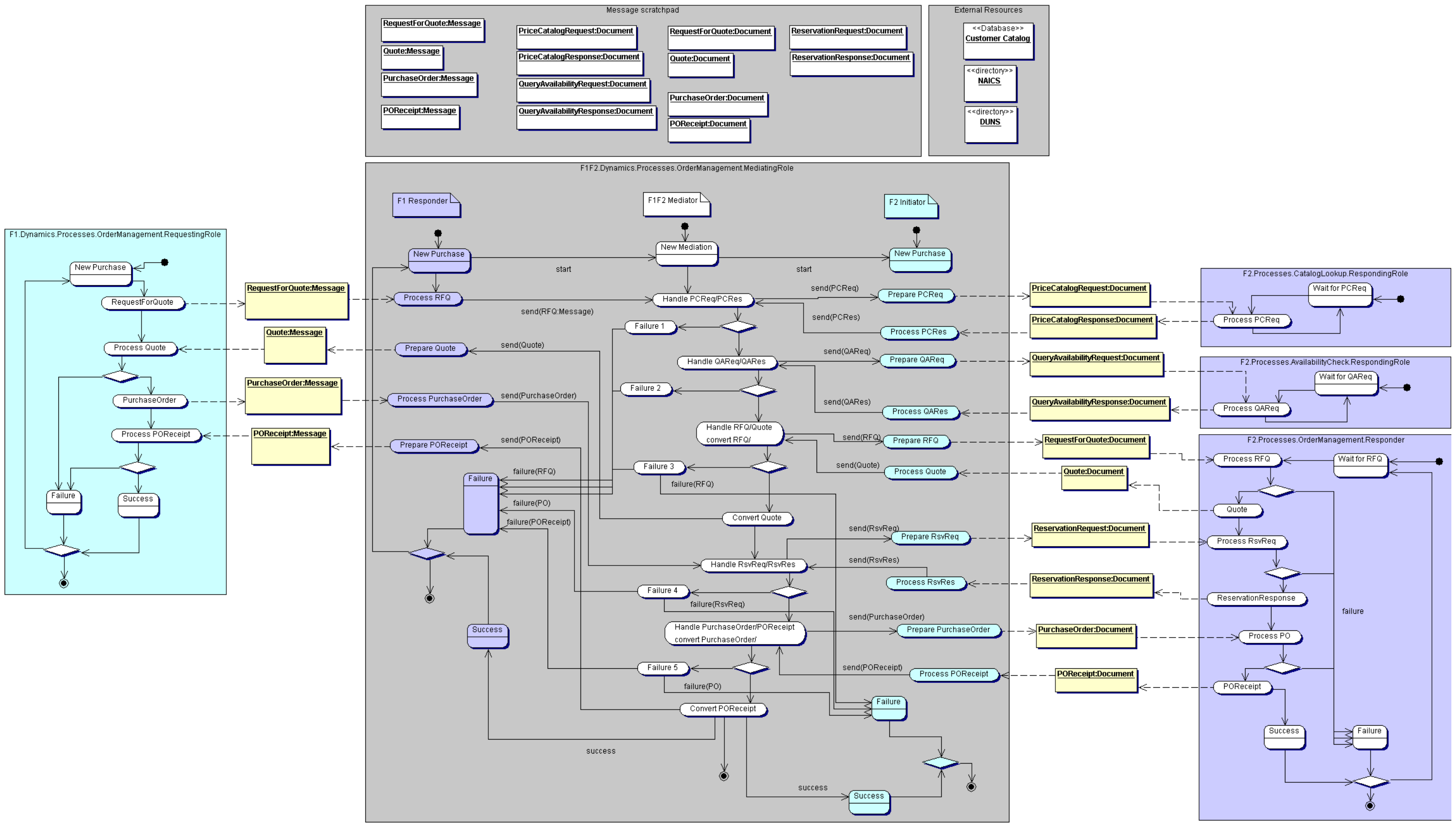


Figure 11 Framework2 business process of OrderManagement.

As the last step on this level of modeling, he proceeds to preparing the model of interactions for the ECIML-compliant agent (mediator). The mediating agent will play the role of Responding Party to the Requesting Party in the Framework 1, and the role of Requesting Party to the Responding Party in the Framework 2.

Note: at this stage, we concern ourselves only with binary collaborations. It is possible to present multi-party collaborations as series of binary collaborations.

In addition to that, the mediating process will use the information elements from the messages, as well as information available from the external resources, in order to fill in the values in the necessary data elements.


Message scratchpad
RequestForQuote:Message
Quote:Message
PurchaseOrder:Message
POReceipt:Message
PriceCatalogRequest:Document
PriceCatalogResponse:Document
QueryAvailabilityRequest:Document
QueryAvailabilityResponse:Document
RequestForQuote:Document
Quote:Document
PurchaseOrder:Document
POReceipt:Document
ReservationRequest:Document
ReservationResponse:Document
External Resources
<<Database>>
Customer Catalog
<<directory>>
NAICS
<<directory>>
DUNS
F1.Dynamics.Processes.OrderManagement.RequestingRole
New Purchase
RequestForQuote
Process Quote
PurchaseOrder
Process POReceipt
Failure
Success
RequestForQuote:Message
Quote:Message
PurchaseOrder:Message
POReceipt:Message
F1F2.Dynamics.Processes.OrderManagement.MediatingRole
F1 Responder
F1F2 Mediator
F2 Initiator
New Purchase
New Mediation
New Purchase
start
start
Process RFQ
send(RFQ:Message)
Handle PCReq/PCRes
send(PCReq)
Prepare PCReq
send(PCRes)
Process PCRes
Failure 1
Prepare Quote
send(Quote)
Handle QAReq/QARes
send(QAReq)
Prepare QAReq
Failure 2
send(QARes)
Process QARes
Process PurchaseOrder
send(PurchaseOrder)
Handle RFQ/Quote
convert RFQ/
send(RFQ)
Prepare RFQ
Prepare POReceipt
send(POReceipt)
send(Quote)
Process Quote
Failure 3
failure(RFQ)
failure(RFQ)
Failure
failure(PO)
failure(POReceipt)
Convert Quote
send(RsvReq)
Prepare RsvReq
Handle RsvReq/RsvRes
send(RsvRes)
Process RsvRes
Failure 4
failure(RsvReq)
send(PurchaseOrder)
Handle PurchaseOrder/POReceipt
convert PurchaseOrder/
Prepare PurchaseOrder
Success
Failure 5
send(POReceipt)
Process POReceipt
failure(PO)
Convert POReceipt
Failure
success
success
Success
PriceCatalogRequest:Document
PriceCatalogResponse:Document
QueryAvailabilityRequest:Document
QueryAvailabilityResponse:Document
RequestForQuote:Document
Quote:Document
ReservationRequest:Document
ReservationResponse:Document
PurchaseOrder:Document
POReceipt:Document
F2.Processes.CatalogLookup.RespondingRole
Wait for PCReq
Process PCReq
F2.Processes.AvailabilityCheck.RespondingRole
Wait for QAReq
Process QAReq
F2.Processes.OrderManagement.Responder
Process RFQ
Wait for RFQ
Quote
Process RsvReq
ReservationResponse
failure
Process PO
POReceipt
Success
Failure

Since preparing a complete meta-model might prove to be a very complex task, he concentrates on specific business scenarios that are required to interoperate.

The framework experts and integrators may use several strategies to approach this task (top-down analysis, best practices, already existing recipes, heuristics), gradually narrowing down the gap between the two frameworks. Finally, they end up with a sufficient (parameterized) meta-model of meaningful interactions between the two frameworks for the given business scenarios.

This model provides an abstract recipe for interoperability between Framework1 and Framework2 (within the given scope). The model can then be processed by an independently implemented ManifestFactory tool that will prepare a machine-readable abstract definition (F1F2Manifest), expressed in the ECIML, defining how to construct the adaptation implementation.

So, the whole process can be summarized by the diagram presented in Figure 13.

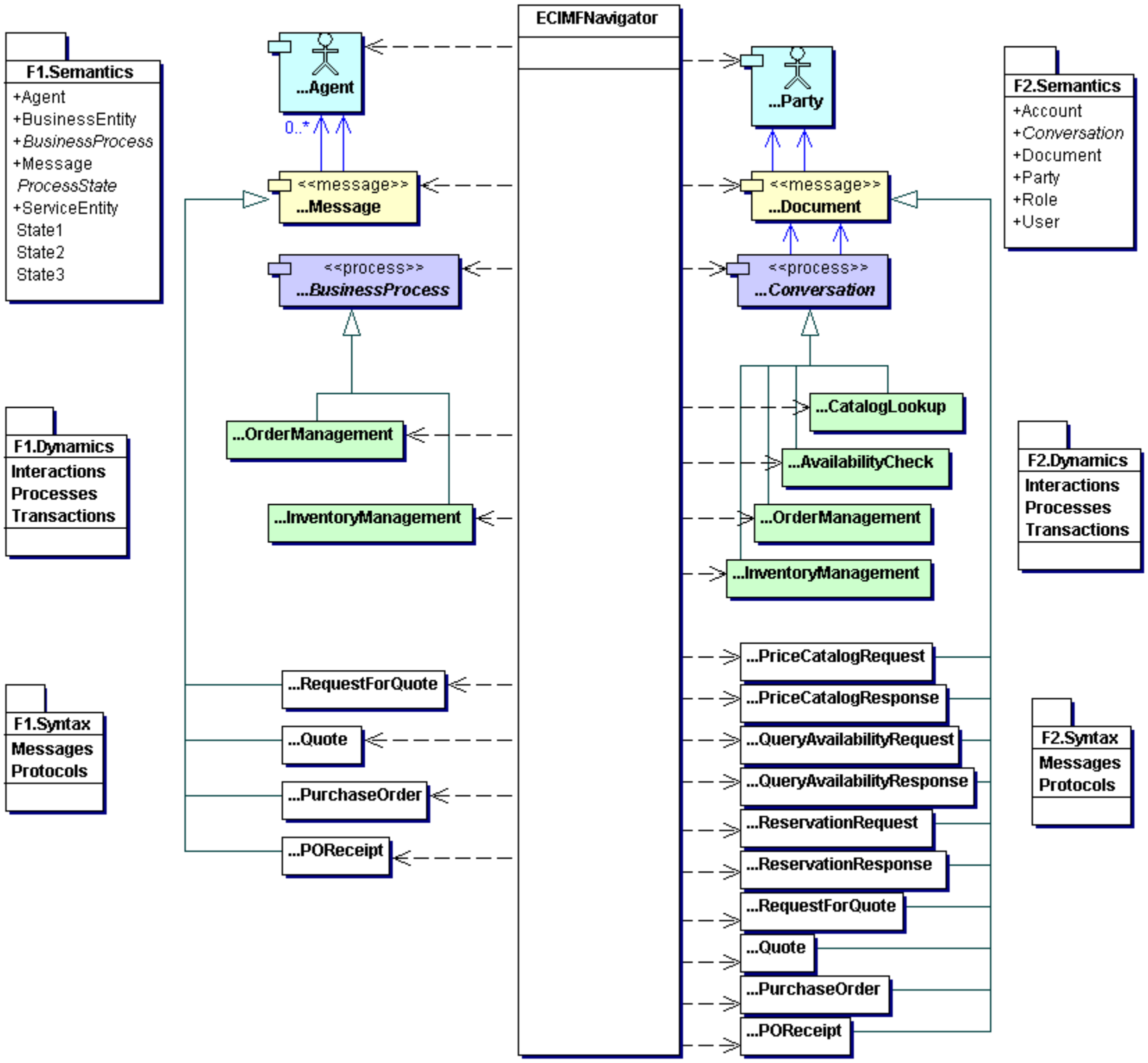


Figure 13 ECIMF Navigator aligns all layers of the frameworks.

In the next step, as previously presented in Figure 5, the ECIML-compliant agent setting up processing pipelines for messages, creating state machines to keep track of complex interactions, creating translation maps for message elements, reading parameters provided by the communicating parties, etc. This reference environment for execution of the MANIFEST recipe can be provided as a commercial product.

Finally, at this stage it is possible for the parties to successfully establish business interaction, even though they use different e-commerce frameworks to express their activities.

5.1. Conzilla – the prototype tool for navigating the standards manifold.

Conzilla is the name of a software tool that has been developed during the past 3 years by the Interactive Learning Environments (ILE) group at the Centre for user-oriented IT-design (CID) at the Royal Institute of Technology (KTH) in Stockholm, Sweden (http://cid.nada.kth.se/il). Conzilla is the first prototype of a *concept browser*, which is a new type of tool for the exploration and presentation of electronically stored information that has been invented by Ambjörn Naeve, a mathematician and researcher within the ILE group at CID. In contrast to most hyperlinked information systems, like e.g. the ordinary web (www), a concept browser supports a clear separation between *context* and *content*, and lets you navigate the different contexts (of a so called *knowledge manifold*), and view the content of a given concept within a clearly defined and displayed context. For a more detailed discussion of the ideas behind conceptual browsing see the report by Naeve: *Conceptual Navigation and Multiple Scale Narration in a Knowledge Manifold*, which is available in PDF format at http://cid.nada.kth.se/sv/pdf/cid_52.pdf.

The basic design principles for concept browsers can be expressed as follows:

- *separate* context from content.
- *describe* each context in terms of a concept map.
- *assign* an appropriate number of components as the content of a concept and/or a conceptual relationship.
- *label* the components with a standardized data description (meta-data) scheme.
- *filter* the components through different aspects.
- *transform* a content component which is a map into a context by contextualizing it.

When desiging concept maps it is important to use a conceptual modeling language that adheres to international standards. At CID, we make use of UML, which has emerged during the past 5 years as “the Esperanto of conceptual modeling”. As for meta-data we make use of the IMS-IEEE proposed standard for learning objects (http://www.imsproject.org).

Conzilla is being developed as an open source project. See www.conzilla.org for more information about the Conzilla project.

The ECIMF project will use Conzilla as a prototype tool for browsing and comparing different e-commerce framework models. One of the goals of the ECIMF project will be to extend this tool by necessary backend(s) producing abstract machine-readable interoperability guides (MANIFEST recipes), expressed in ECIML language

Annex 2 – Comparison of the modeling notations for Business Process and EAI modeling

Note: this annex, due to its size, is provided as a separate Word document.

[1] The ebXML project, http://www.ebxml.org/specdrafts/ .

[2] The e-Speak framework, Hewlett-Packard, both as a commercial product http://www.e-speak.hp.com, and an OpenSource free Java implementation of the complete framework at http://www.e-speak.net .

[3] The BizTalk framework, Microsoft, http://www.microsoft.com/biztalk/techinfo/BizTalkFramework20.doc , BizTalk repository at http://www.biztalk.org, and commercial product BizTalk Server http://www.microsoft.com/biztalk , which additionally contains the mapping and orchestration tools.

[4] The eCo Framework, CommerceOne, http://www.commerce.net/eco .

[5] RosettaNet, http://www.rosettanet.org .