E-Commerce Integration Meta-Framework – General Methodology (ECIMF-GM)

CEN/ISSS/WS-EC/ECIMF

Draft, version 0.2
July 11, 2001

1. The methodology

The proposed methodology for analysis and modeling of the transformations between the e-commerce frameworks follows a layered 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, and how these contexts affect the semantic properties of the concepts,
- 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 instance (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:

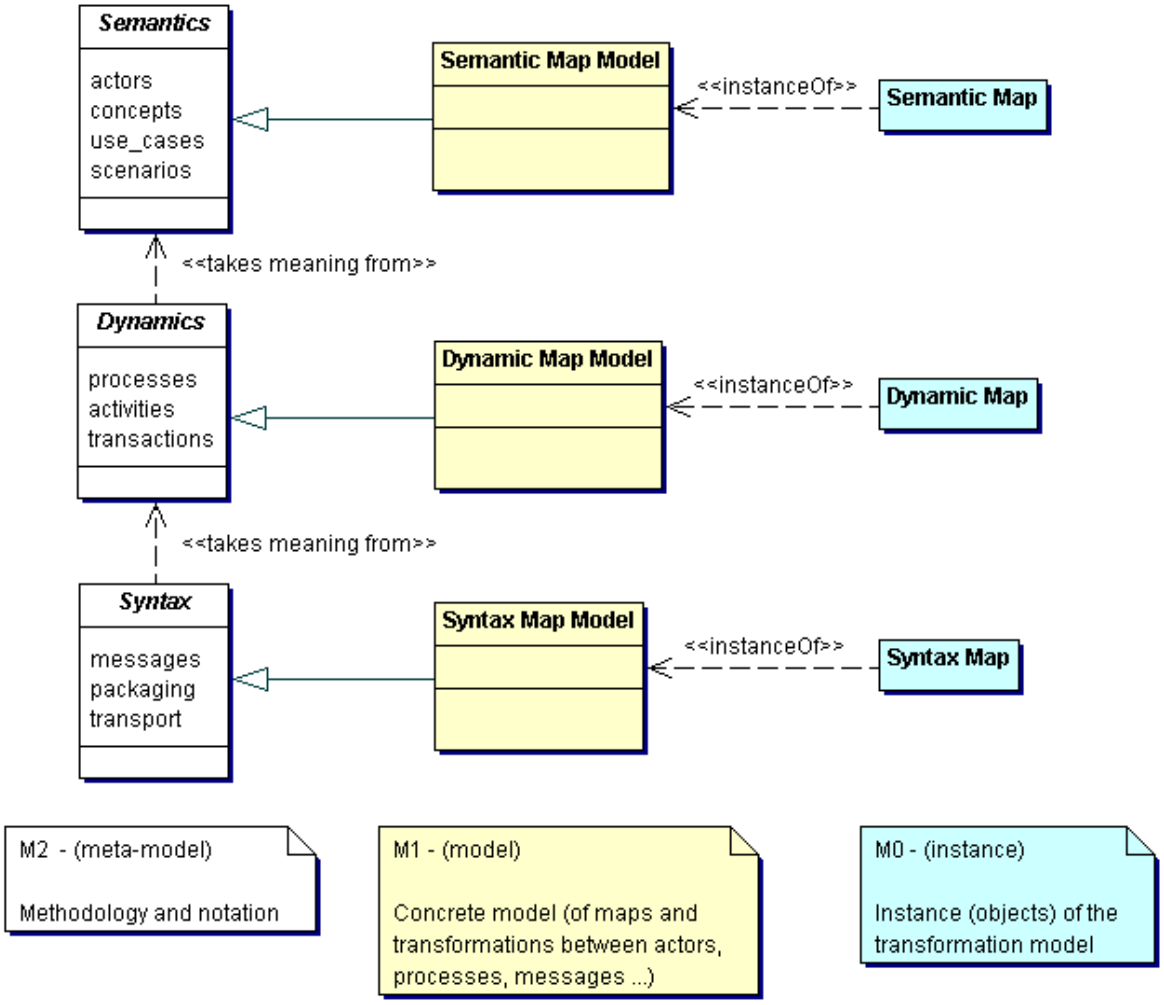


Figure 1 ECIMF methodology and the meta-model architecture.

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 consists of a recommended methodology (named E-Commerce Integration Modeling Methodology – "ECIMM"), presented in this document, and base tools needed to prepare specific comparisons of concrete frameworks (presented in the ECIMF-POC document), 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.

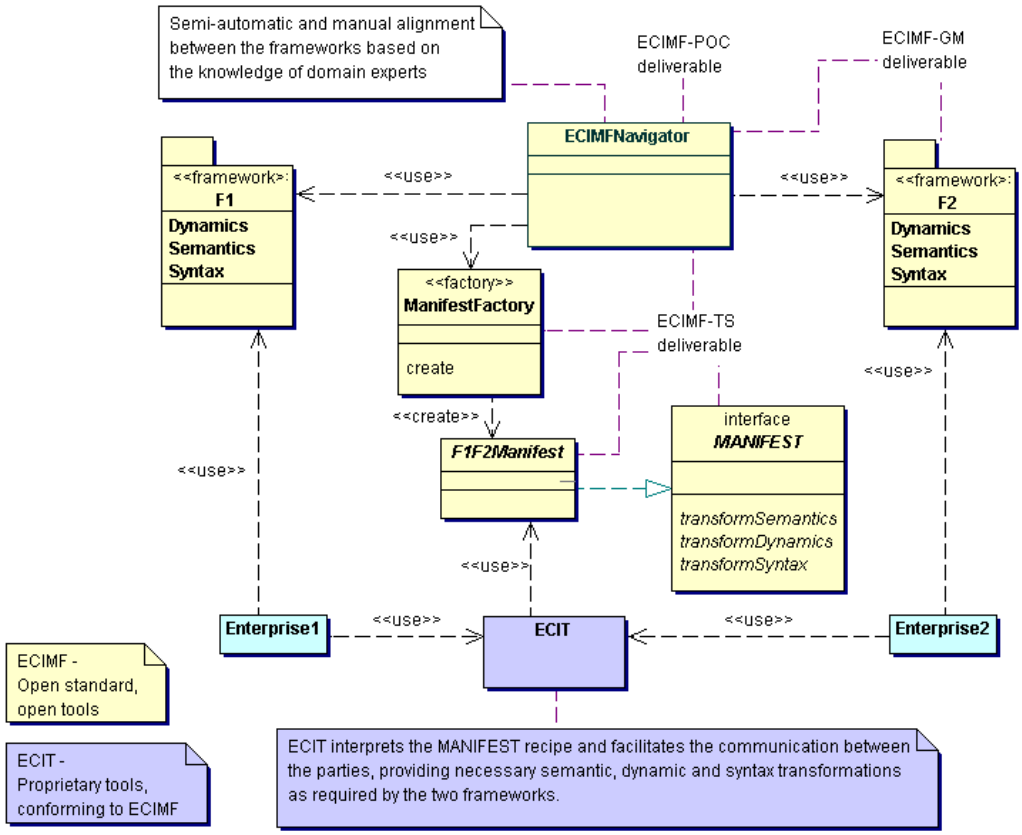


Figure 2 The ECIMF concept of frameworks transformation and alignment.

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 extended UML models, rules and definitions.

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

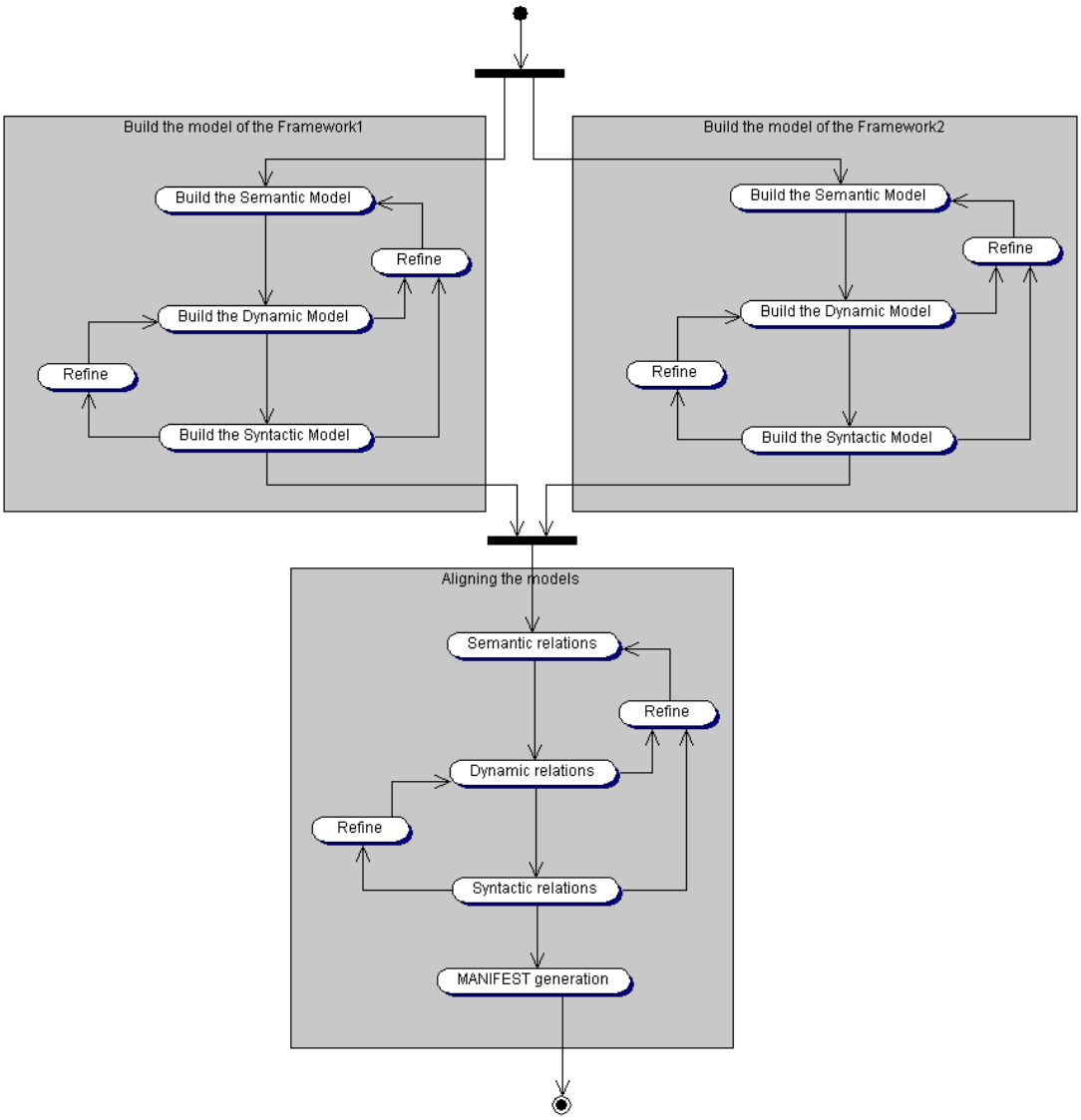


Figure 3 The process of modeling and alignment between two e-commerce frameworks.

2. The modeling notation

The ECIMF project proposes the use of extended UML modeling notation 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:

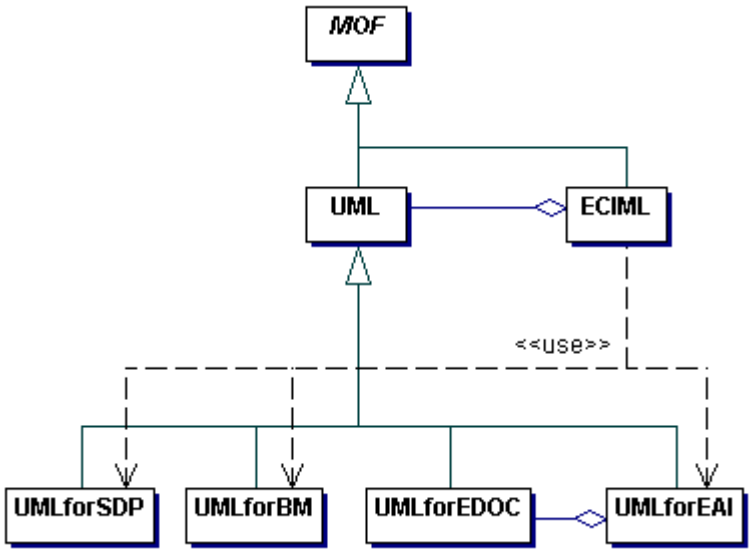


Figure 4 Relationship between the ECIML and other modeling standards.

In other words, the ECIML will be yet another profile of UML 1.4. 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. We will also reuse as much as possible the specialized concepts developed by the UN/CEFACT Unified Modeling Methodology (UMM), as described in TMWG-N090R9.1.

One could use the standard UML for modeling the interoperability concepts, but we feel that in its current form it is 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 these concepts 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.

3. MANIFEST recipes

A MANIFEST document consists of a set of interoperability recipes, based on the transformation model prepared using ECIML notation and then expressed in a serialized (XML) format. The MANIFEST-s 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 should also be possible for ECIML-compliant modeling tools to re-use already existing MANIFEST recipes to adjust the interoperability model to specific needs. It is expected that some publicaly available repository will store the 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. Conzilla – the prototype tool for navigating the standards manifold.

Conzilla is the name of a software tool that has been in development from the year 1998, 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 implementation 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 uses an extension of Conzilla as a prototype tool for browsing and comparing different e-commerce framework models. One of the goals of the ECIMF project is to extend this tool by necessary backend(s) for producing abstract machine-readable interoperability guides (MANIFEST recipes), expressed in ECIML language.

5. The Toolkit

The project aims to provide a simple implementation of the E-Commerce Integration Toolkit ("ECIT"), consisting of the ECIMF Navigator (extended Conzilla) and a basic implementation of ECIML-compliant agent, and make these 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.

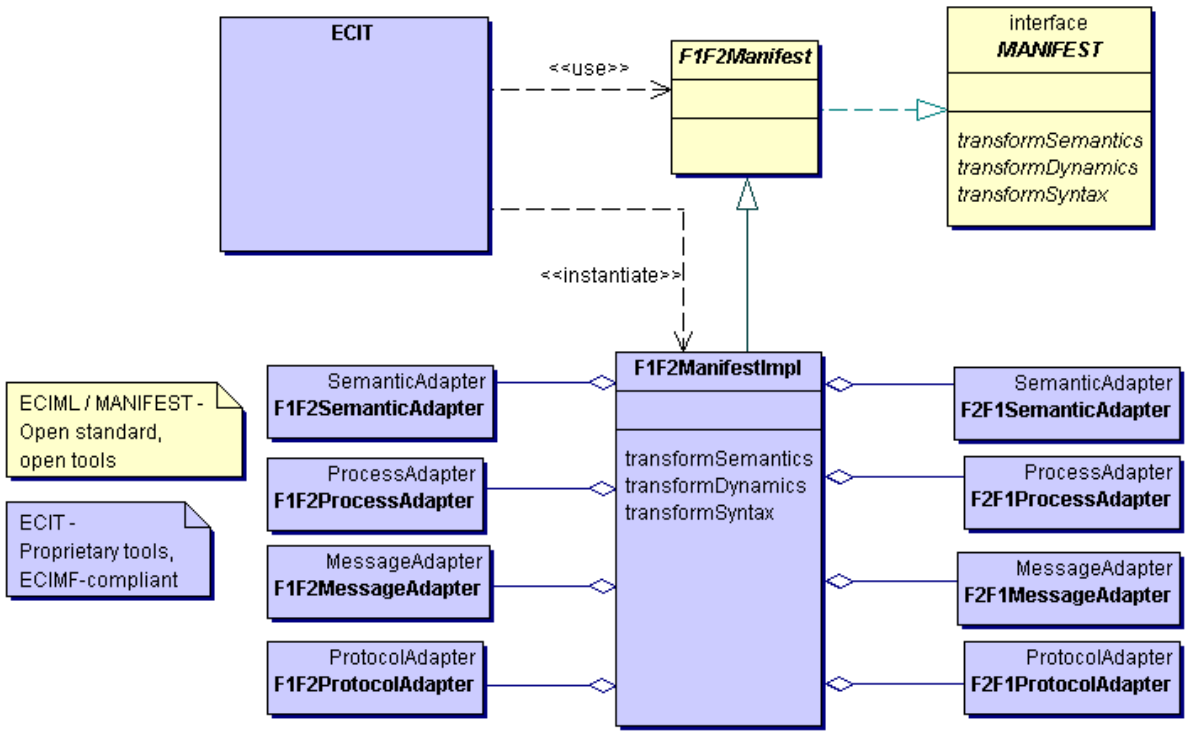


Figure 5 Example of ECIT (ECIML-compliant agent) facilitating message exchange.

Annex 1 – Example ECIMF model.

This example presents step by step how a meta-framework recipe for interoperability could be prepared, between hypothetical e-commerce frameworks Framework1 and Framework2. Please refer also to the Annex 2 for definitions of the concepts presented in the example frameworks.

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. In some cases, the frameworks already have more or less comprehensive models available (as is the case with e.g. RosettaNet and e-Speak). These model are then re-structured to match the ECIMF layers. 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:

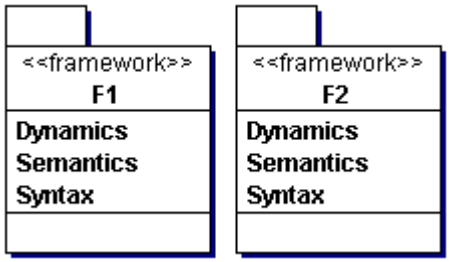


Figure 6 Modeling the frameworks

Then, using the ECIMF Navigator or a similar tool, the framework experts calibrate and align the semantics of the concepts common to both frameworks. In other words, they try to establish precise correspondence between the concepts, their common, unique and conflicting properties. This knowledge will then drive the integration efforts on the business process level and the syntax level.

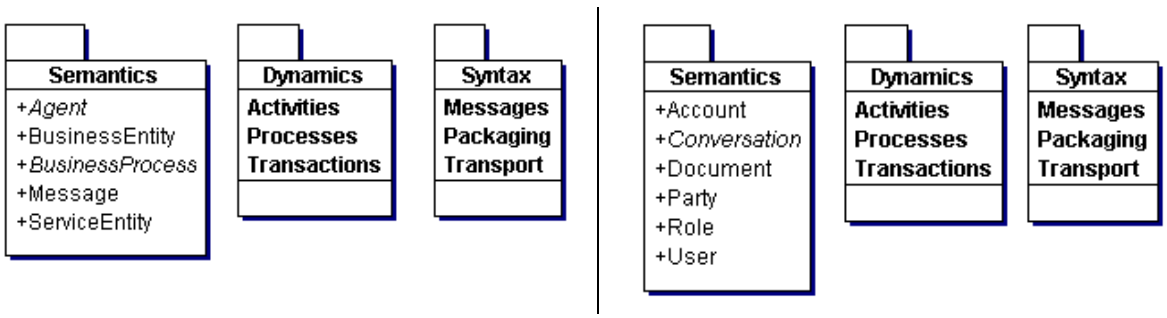


Figure 7 The top-most layers of the Framework1 and Framework2 models.

Let's look closer at this example. The figure 8 presents the semantic 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 (please refer to Annex 2 for more detailed descriptions):

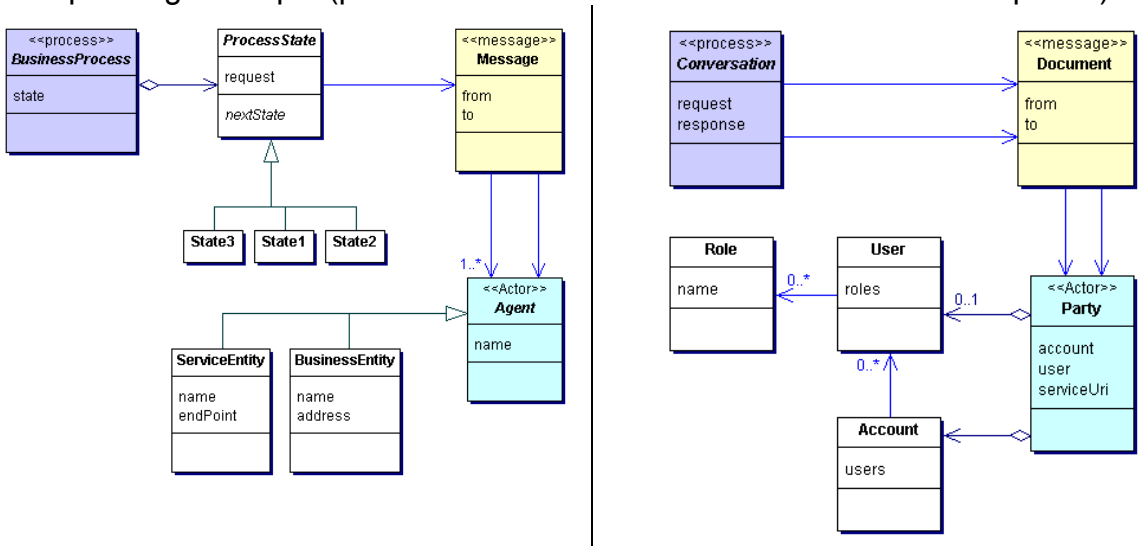


Figure 8 Comparing the corresponding semantic elements.

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:

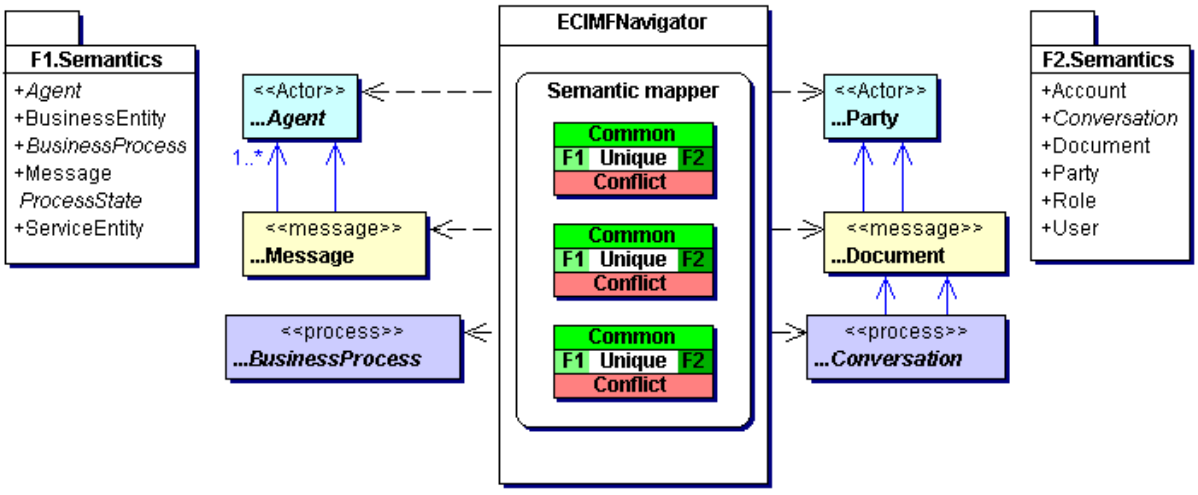


Figure 9 The ECIMF Navigator compares the semantic elements of the frameworks.

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 the ECIMF-ProcessModeling document for comparative study of the notations).

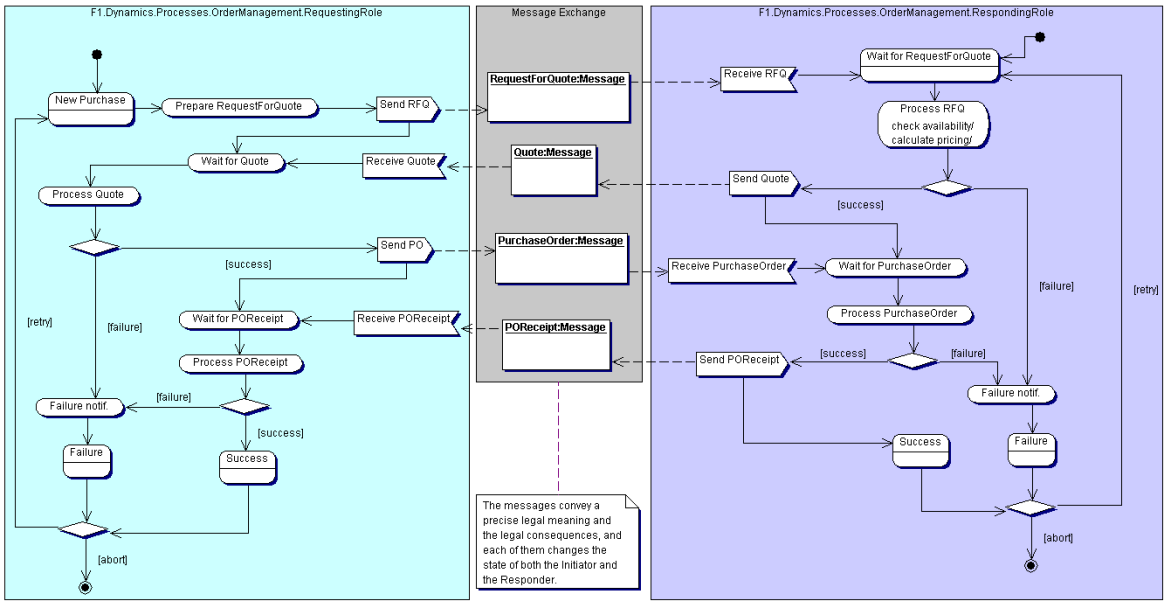


Figure 10 Framework1 business process of OrderManagement.

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 (or, rather a group of business processes) is different from the corresponding process in Framework1. The result is presented in Figure 11.

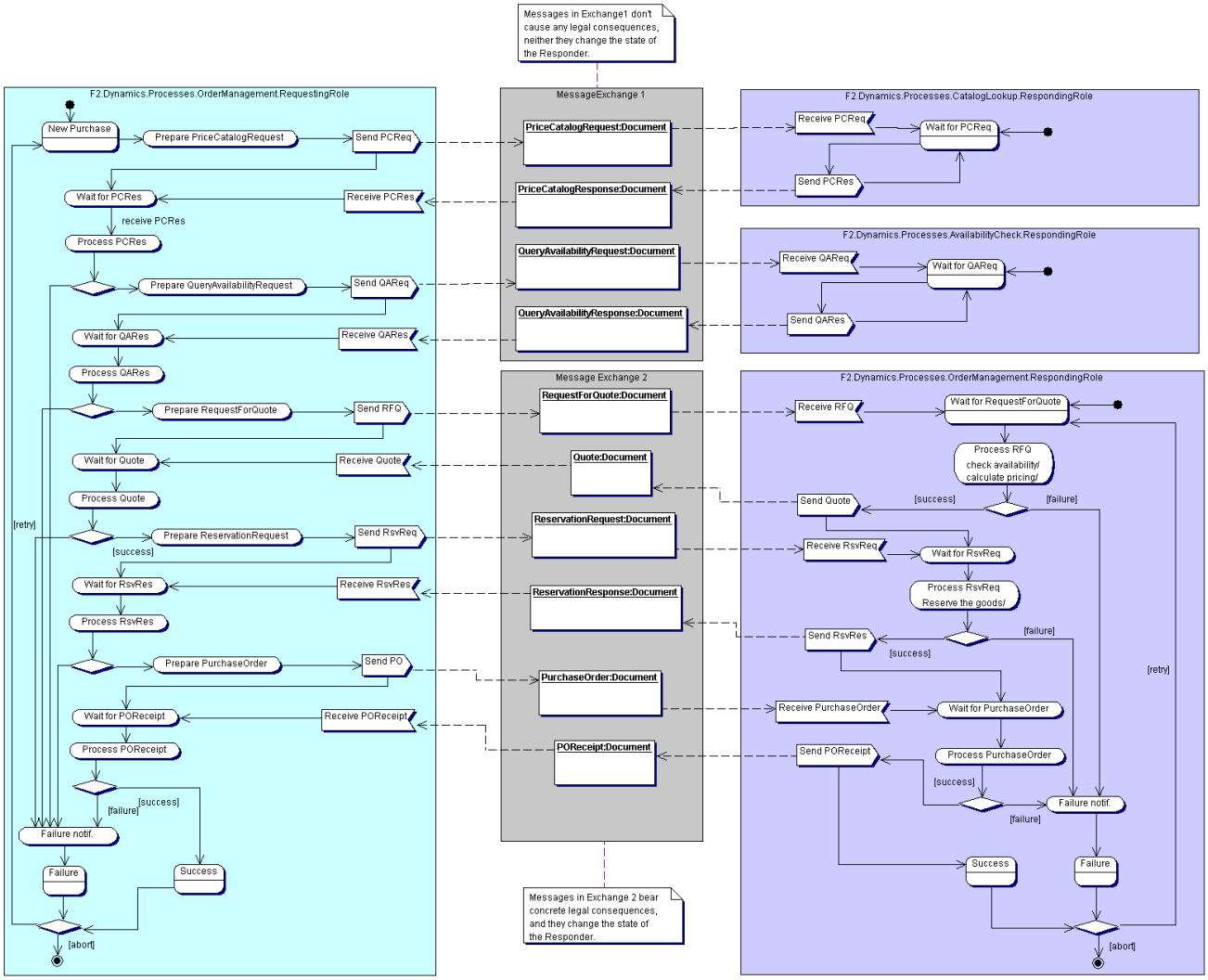


Figure 11 Framework2 business processes related to OrderManagement.

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: in this example, 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 mediator 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 expected data elements.

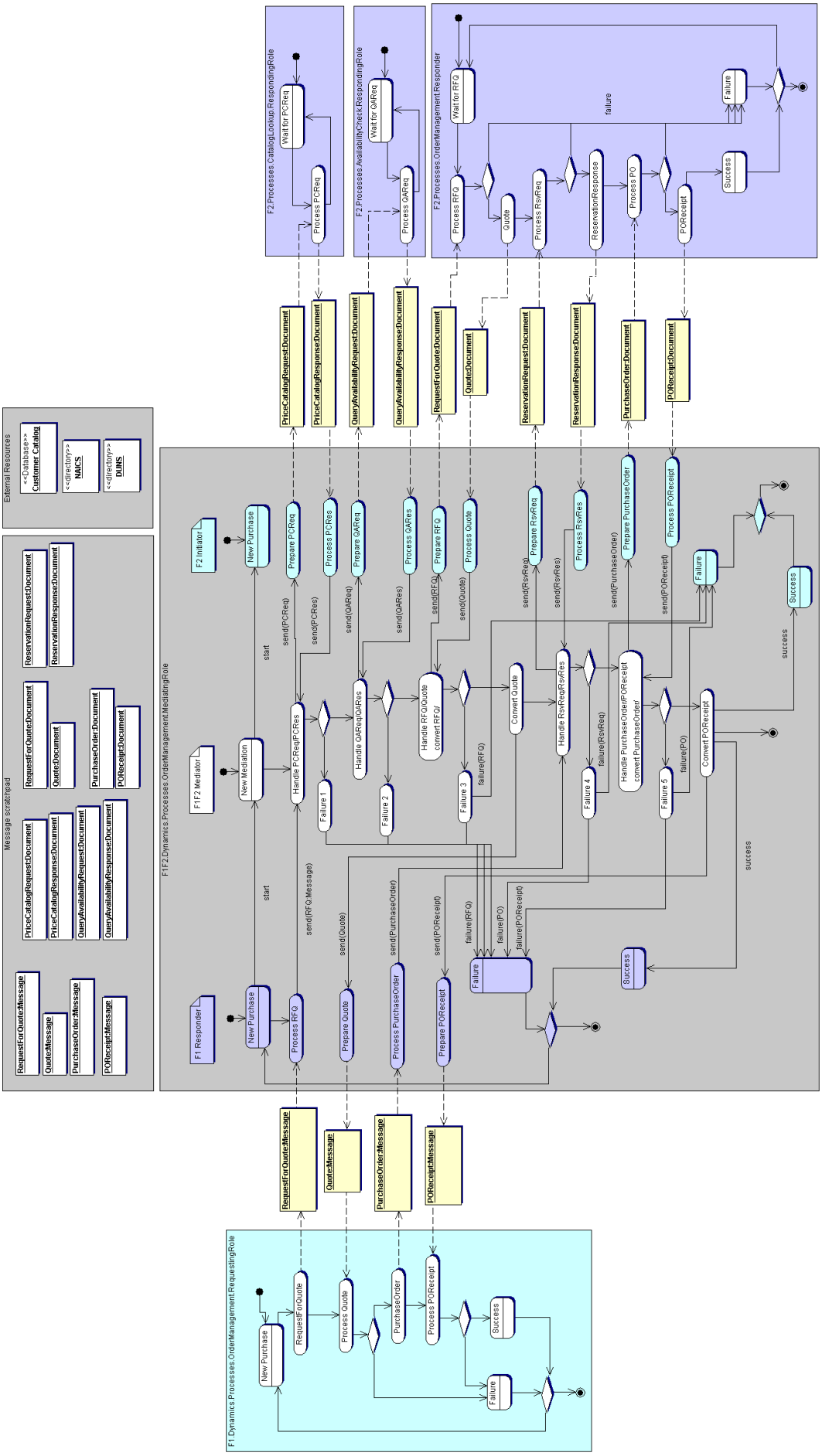


Figure 12 The process specification for ECIMF mediator.

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.

This step can be illustrated with the following figure, presenting the ECIMF Navigator tool used to define the process mediator.

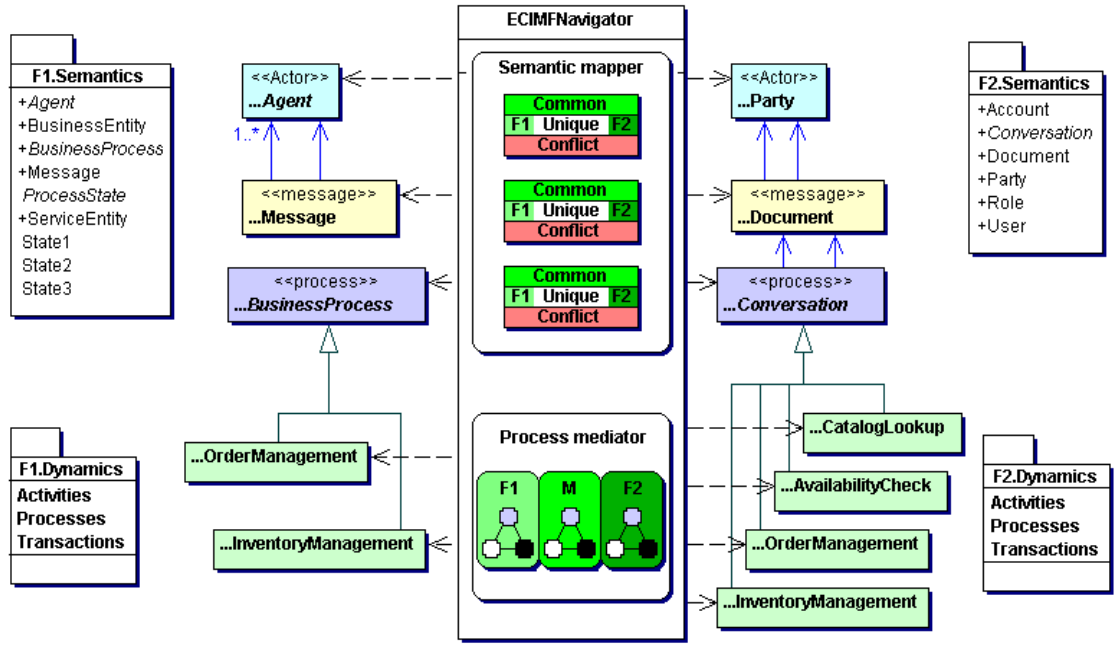


Figure 13 Using ECIMF Navigator to define the process mediator.

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) model of meaningful interactions between the two frameworks for the given business scenarios.

As the final step, the system integrator prepares mappings between syntax layers, i.e. mapping and transformations needed between data elements and message formats, packaging specifications and transport protocol configurations. Since at this stage he knows how both parties understand these messages and elements, and when they expect to receive or send specific data, it makes the task much easier.

So, the diagram presented in Figure 13 can summarize the whole process.

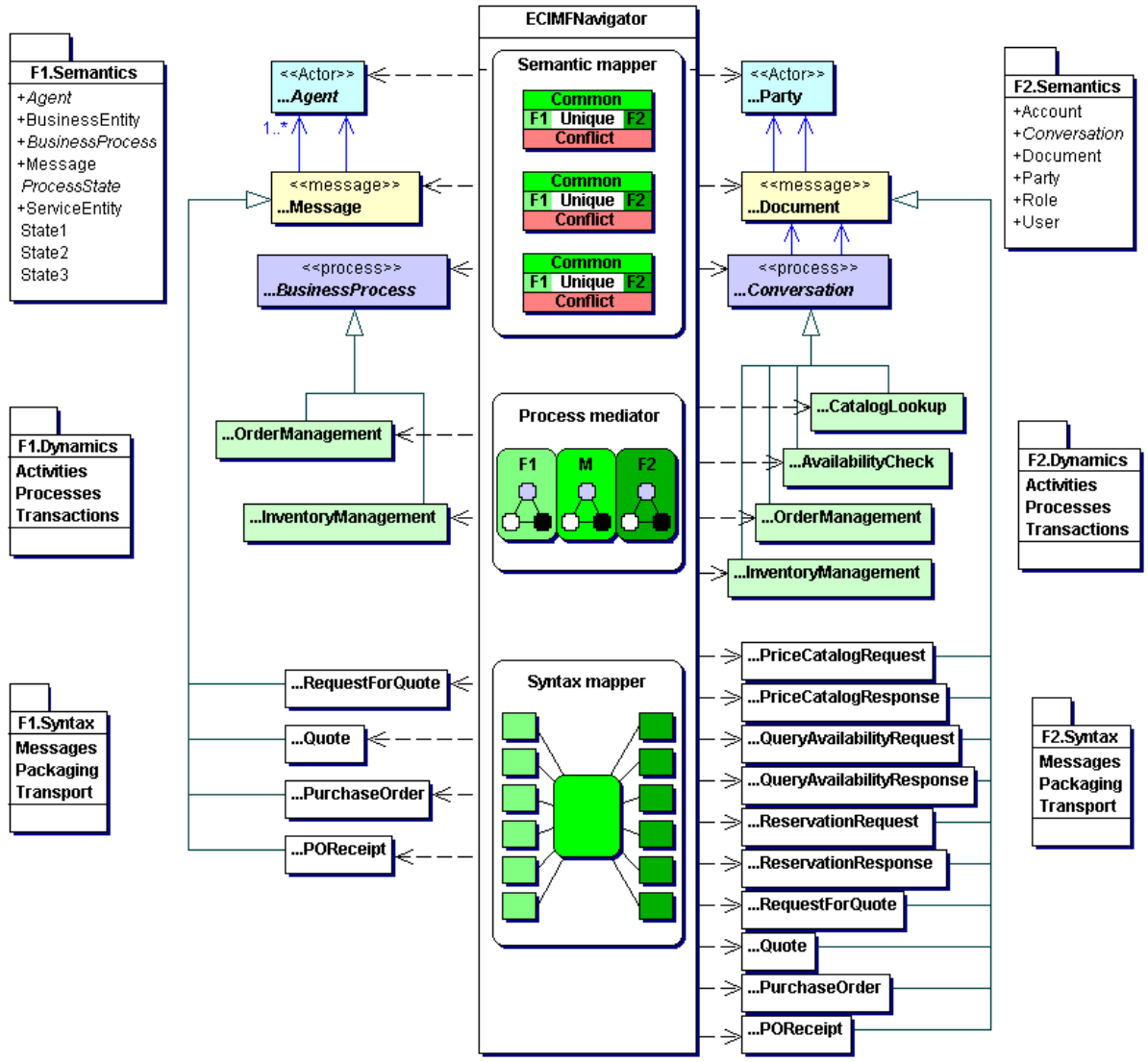


Figure 14 ECIMF Navigator aligns all layers of the frameworks.

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

The example syntax of the MANIFEST document could look like the sample below:

```
<?xml version='1.0'?>
<Manifest>
   <Process name='Procurement'>
      <Framework id='A' name='WidgetsLtd'>
         <BusinessProcessDefinition>
            ... (here it follows)...
         </BusinessProcessDefinition>
      </Framework>
      <Framework id='B' name='ebXML'>
         <BusinessProcessDefinition location='uddi:...'/>
      </Framework>
      <MappingRules>
         <SemanticMapping>
```

```
            <MapElement from='A' to='B'>
               <Element in='A'>Actor<Element>
               <Element in='B'>Party</Element>
               <Element in='B'>User</Element>
               <Common> ... </Common>
               <Unique in='A'> ... </Unique>
               <Unique in='B'> ... </Unique>
               <Conflict> ... </Conflict>
            </MapElement>
            <MapElement from='A' to='B'>
               <Element in='A'>BusinesEntity<Element>
               <Element in='B'>Party</Element>
               <Element in='B'>User</Element>
               <Common> ... </Common>
               <Unique in='A'> ... </Unique>
               <Unique in='B'> ... </Unique>
               <Conflict> ... </Conflict>
            </MapElement>
         </SemanticMapping>
         <ProcessMediating> ... </ProcessMediating>
         <SyntaxMapping> ... </SyntaxMapping>
      </MappingRules>
   </Process>
</Manifest>
```

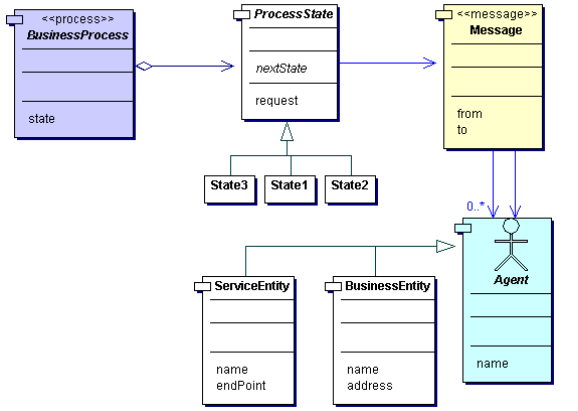
In the next step, as presented previously in the Figure 5, the ECIML-compliant agent receives the F1F2Manifest and instantiates the necessary adapters. This may involve setting up processing pipelines for messages, creating state machines to keep track of complex interactions, creating translation maps 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.

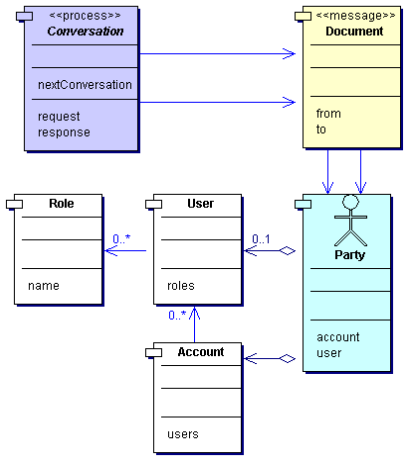
Annex 2 – Semantics of the example frameworks

NOTE: Both frameworks are purely hypothetical. The authors used in preparation of the example the concepts present in RosettaNet and e-Speak, so that the Framework 1 resembles simplified RosettaNet model, and the Framework 2 resembles simplified e-Speak model. However, the example frameworks are by no means representative for the real models, and in fact conflict with them. This example will be replaced at later stage with the real models of business processes from two selected frameworks. Meanwhile, readers are encouraged to refer to the original documentation for more details: RNIF and SFS2.0 specifications respectively.

1. Models of the central semantic concepts



Framework 1 Semantics



Framework 2 Semantics

2. Semantic correspondence

Framework 1	Common	Framework 2
Agent: Abstract entity involved in message exchange, either as an initiator, or a responder. The same Agent may play each role in the course of a business process. **Specialization:** • BusinessEntity: human or organizational actor involved in the business process • ServiceEntity: application-level service end-point involved in the message exchange	An entity involved in data exchange, either as an initiator, or a responder. The same entity may play each role in the course of related data exchanges. **F1 unique:** see specialization **F2 unique**: see identification	**Party:** Concrete entity involved in document exchange, either as an initiator, or a responder. The same Party may play each role in the course of a conversation. **Identification:** Party always represents human or organizational actor, playing specified role in the conversation. It is identified by a set of ID-s, related to Account, User, and network service end-point identifiers.
Message: An XML instance conforming to specific schema, consisting of service-related information (needed for transport, packaging and routing), plus optionally the business data payload.	An XML document instance conforming to specific schema. The document contains references to the entities involved in the data exchange. Documents always contain the business payload data.	**Document:** A business-related structured data, expressed as XML instance conforming to specific schema. Documents are sent and received by Parties. Exactly two Party ID-s uniquely identify the business

Framework 1	Common	Framework 2
Messages are sent and received by ServiceAgents, and contain the ServiceEntity identification as well as the BusinessEntity identification. Messages can be sent to multiple recipients. Messages can be exchanged using any MIME-compatible protocol (SMTP and HTTP).	**F1 unique**: message may contain both transport-level information and business level, or just the former. It may be sent to multiple recipients. **F2 unique**: ability to use MS-MQ binary format	actors as well as the service end-points. Documents are packaged into a transport-specific envelope (e.g. MIME for SMTP and HTTP, binary for MS-MessageQueueing).
BusinessProcess: Business process consists of several **ProcessStates** that specify the request (message), which caused entering this state, and the specification for events (messages) to transition to the next state. Each state is implicitly related to processing of the request message, and sending of 0, 1 or 2 response message(s). Business process may involve multiple agents.	Orchestration of data exchange is structured into request/response dialogs, which result from processing activities and state changes in entities. **F1 unique**: both multiple and binary collaborations. Business process is explicitly stateful. **F2 unique**: only binary collaborations. Conversation is stateless. Parties keep implicit state (not in the scope of Conversation specification), and its violation results in specific documents being sent.	**Conversation:** Conversation contains a specification of states, transitions and activities related to the documents being exchanged by parties. Conversations are always specified as interactions between exactly two parties. Multiple collaborations are expressed as series of binary collaborations. Document exchanges form request/reply dialogs, always with 1 request resulting in 1 reply document.