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

CEN/ISSS/WS-EC/ECIMF

Draft, version 0.3
November 28, 2001

1. Overview

The ECIMF project deliverables consist of a recommended methodology, presented in this document, the technical specification (described in the ECIMF-TS document) and base tools needed to prepare specific comparisons of concrete frameworks (presented in the ECIMF-POC document, where you can also find the case studies).

The results of following the ECIMF methodology should be clear implementation guidelines for system integrators and software vendors on how to ensure interoperability and semantic alignment between incompatible e-commerce systems. This generic integration rules will be expressed in the ECIML language, providing mapping and transformation descriptions/recipes that can be implemented by ECIMF-compliant agents/intermediaries. This ultimately should allow the e-commerce frameworks to interoperate without extensive manual alignment by the framework experts, and will make the integration logic more understandable and maintainable.

1.1. Layered approach

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 establish the scope of the integration task in terms of a business context – based on the economic aspects of the partners' interactions,
- Then, to identify the top-level entities and the contexts in which they occur (the data model), and how these contexts affect the semantic properties of the concepts,
- Then, to proceed to the next layer in which the interactions (conversation patterns) between the partners are analyzed.
- Then, to go to the lowest, the most detailed level to analyze the messages and data elements in communication between the partners.

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. This means that ECIMF will be used to define:

- The modeling notation: a set of modeling concepts with their graphical and XML representation to model the transformations[1],
- The models: concrete transformations between concrete frameworks
- And the model instances of transformations, as realized by an ECIMF-compliant runtime.

[1] Since the modeling elements regard multiple layers of the ECIMF approach, hence the name "meta-framework", because they will be used to define interoperability frameworks.

Figure 1 presents the ECIMF layers, and how they are applied to define the interoperability model between two incompatible frameworks.

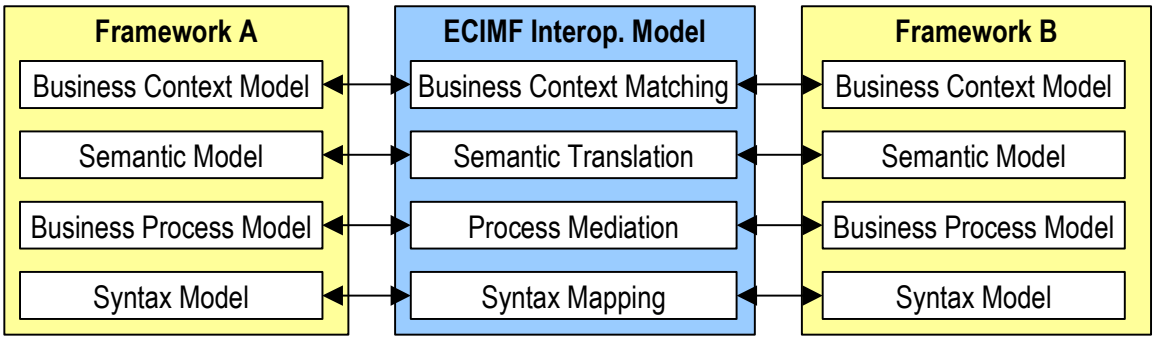


Figure 1 ECIMF methodology – interoperability layers.

Each of these layers is described in detail in the section 2.

1.2. Conceptual navigation – ECIMF Navigator

In order to navigate through the framework models and concepts, a prototype tool named Conzilla is introduced, which in later stages of the project will be augmented with other modules (like data format translating software, automatic generation of interfacing state machines, routing and packaging translators, etc). This extended toolset is called ECIMF Navigator, and its intended use is presented on the Figure 2.

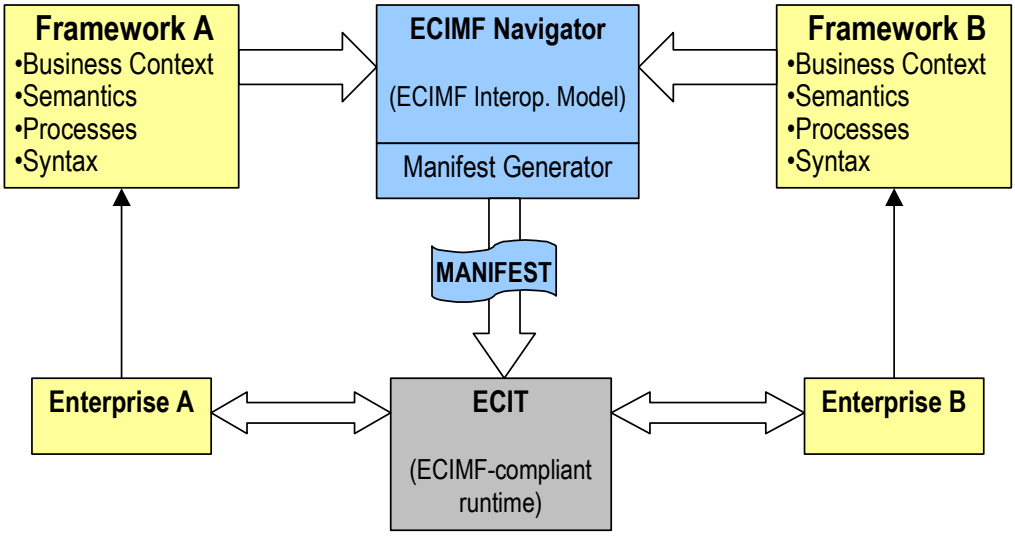


Figure 2 The ECIMF concept of frameworks transformation and alignment.

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

Conzilla is being developed as an open source project. See http://ww.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.

1.3. Top-down, iterative process

The ECIMF uses a classic top-down approach for solving the interoperability issues, but combined with an iterative process of refining the higher level models based on the additional information gathered in the process of modeling the lower levels.

This process is described in detail in the Framework Integration Guidelines section.

1.4. The modeling notation

The ECIMF project proposes to use an extended UML modeling notation (a UML profile) 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, will 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:

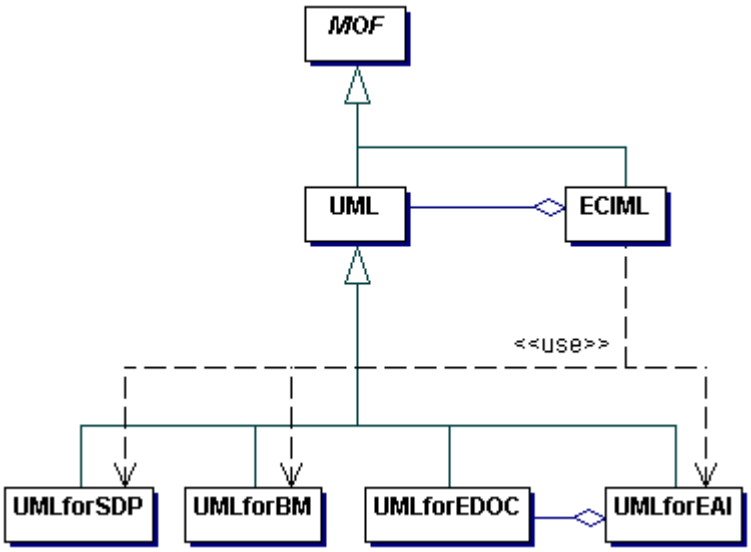


Figure 3 Relationship between the ECIML and other modeling standards.

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-N090R10.

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.

2. Methodology

As mentioned in the overview section, the ECIMF methodology addresses the following four layers of interoperability:

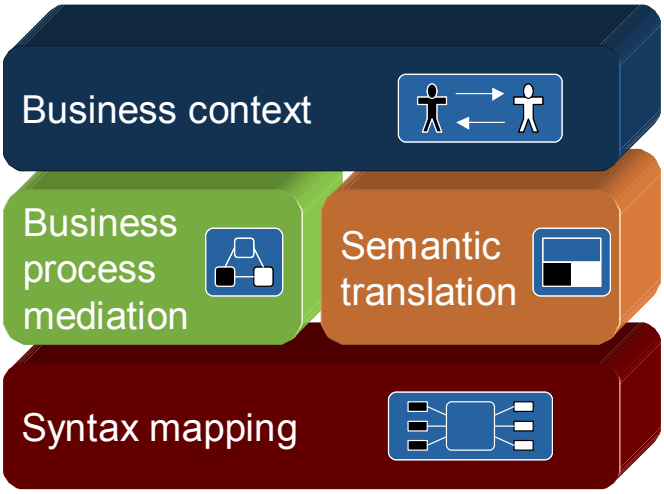


Figure 4 ECIMF layers of integration

- **Business Context Matching:** this stage deals with setting up the scope of the integration task – we assume that preparing a complete integration specification for all possible interactions might not be feasible (even if it were possible at all), so the task needs to be limited to the scope needed for solving a concrete business case. This case is identified, the models for each party are prepared, and then it needs to be determined if they match, i.e. if the business partners try to achieve the same business goals.
- **Business Process Mediation:** in this step the necessary mediation logic is defined, by introducing an intermediary agent that can transform conversation flow from one framework to that of the other, while preserving the business semantics (e.g. the transaction and legal boundaries).
- **Semantic Translation:** in this step the key concepts and their semantic correspondence is established, so that they can be appropriately transformed whenever they occur in contexts of each of the frameworks (which is also known as "semantic calibration" [CID52]).
- **Syntax mapping:** in this step the mapping between data elements in messages is defined, based on the already established semantic correspondence and translation rules defined in the first step. Also, the transport protocol and packaging translation is specified.

The following sections describe in detail each of these areas of interoperability.

2.1. Business Context Matching

2.1.1. Importance of the business context

- IT infrastructure exists to support business goals
  - IT systems don't exist in a void
  - IT systems play specific roles in the business
- Business context is therefore crucial

    - Information is useful only when considered in the right business context
    - Business context determines the meaning of data and information exchange
- Business flow before technical flow
- REA is often used as the underlying meta-model

2.1.2. Resource-Event-Agent modeling framework

REA Enterprise Ontology has been created by William E. McCarthy, mainly for modeling of accounting systems. However, it proved so useful and intuitive for better understanding of business processes that it became one of the major modeling frameworks for both traditional enterprises and e-commerce systems. Recently, it has been extended to provide concepts useful for understanding the processing aspects (processes, recipes) in addition to the economic aspects (economic exchanges). Please see http://www.msu.edu/user/mccarth4/ for more information.

Some of the REA concepts have been used to model the Business Requirements in UN/CEFACT Modeling Methodology ("UMM", formally known as TMWG N090), and the Business Process Analysis Worksheets in ebXML, and it's use is currently a subject of further study in the Business Collaboration Patterns and Monitored Commitments team of the E-Business Transitionary Working Group (eBTWG) - the successor to ebXML.

2.1.2.1. Economic exchange as a central concept

- REA ontology focuses on the idea of economic exchange of resources as the basis of business and trading. In REA models, economic agents exchange economic resources in series of events, which fulfill mutual obligations (called Commitments), as specified in an Agreement between the business partners. See also the detailed definitions in the ECIMF-TS document.
- Economic exchange models define collaborations between partners involved in the process, and these collaborations naturally map to business document exchanges (both in paper and in electronic form).

2.1.2.2. Value-chain models (REA Enterprise Scripts)

- REA process diagrams show the high-level flows of economic resources in the enterprise, related to the economic events and collaborations between the agents involved in the exchanges. They are sometimes referred to as value-chain diagrams.
- The resource flows between processes in the value-chain diagrams represent the collective unbalanced stock-flows, consumed and produced by the events belonging to given processes.
- Value-chain model (also known as *REA Enterprise Script*) is a series of processes, consisting of exchanges, where collaborations between agents are realized with recipes (groups of ordered tasks).

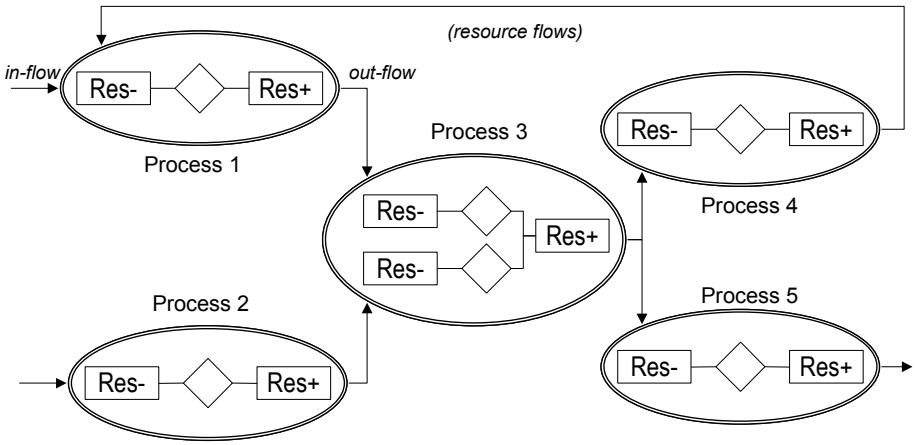


Figure 5 Enterprise value-chain, seen as series of exchanges.

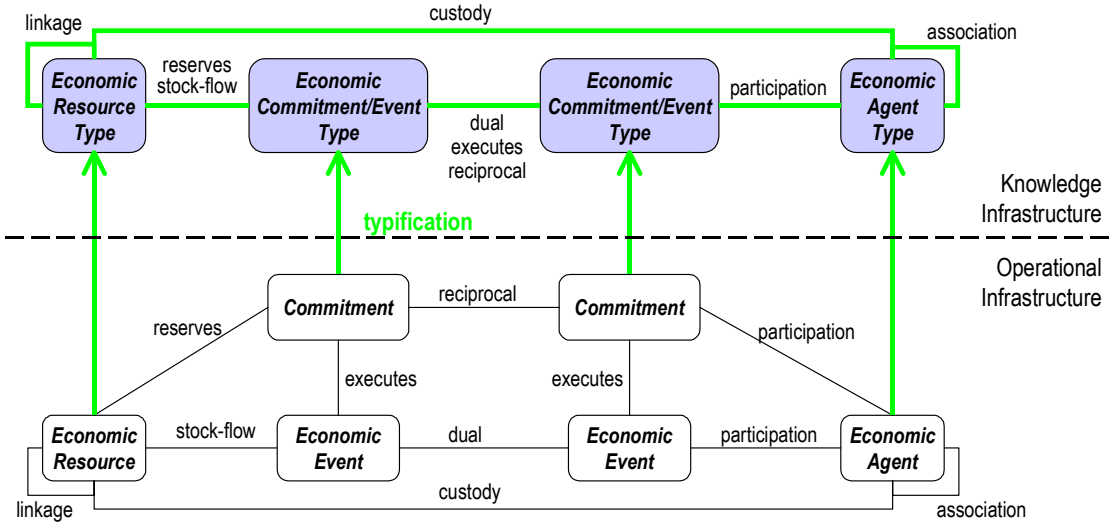


Figure 6 REA meta-model of economic exchanges (simplified).

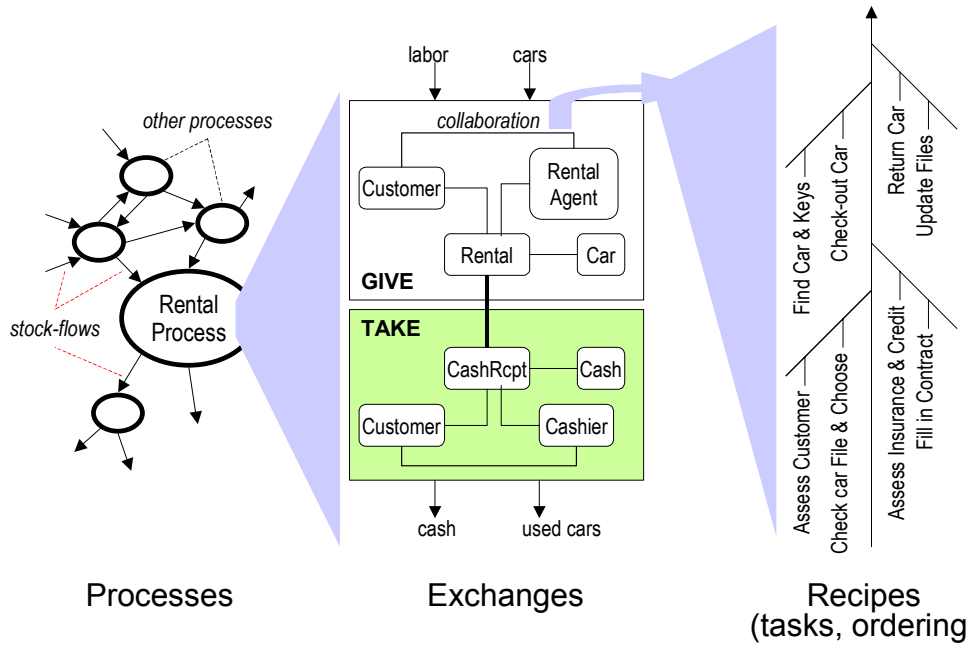


Figure 7 Overview of the processes, exchanges and recipes.

You will find the detailed description of this meta-model in the ECIMF technical specification document (ECIMF-TS).

2.1.3. Business Context Matching rules

2.1.3.1. Rationale

- Traditional trading partners' agreements
  - Both partners need to agree on:
    - The type of resources exchanged
    - The timing (event sequences/dependencies)
    - The persons/organizations/roles involved
  - Each of the partners needs to follow the commitments under legal consequences
- Conclusion: in the traditional business, partners achieve common understanding through negotiations, and their results and conditions are then recorded in a formal written contract. In electronic business some standards support creation of electronic TPA's (Trading Partner Agreements). Their formation is a special case of establishing the Business Context Matching described here.

2.1.3.2. Matching Rules

Business partners involved in an integration scenario need to consider first whether their business goals and expectations match, before they start solving the technical infrastructure problems. For that purpose, they can create two (or more) business context models, one for each party involved in the integration scenario. The interoperability of the e-commerce scenario, as implemented by two different partners, requires that these models match.

There are several requirements that the models have to meet for them to be considered matching:

#1: Complementary roles
Parties need to play complementary roles (e.g. buyer/seller)

#2: Matching resources
The resources expected in the exchanges need to match to the ones expected by the other partner (e.g. the provided resources could be subtypes of resources requested)

#3: Satisfied timing constraints
The timing constraints on events (commitment specification) need to be mutually satisfiable (e.g. down payment vs. final payment, payment within 24 hours, shipment within 1 week, etc...)

#4: Transaction preservation
The sequence of expected business transactions needs to be the same (even though the individual business activities and resulting conversation patterns may differ). This is especially important for those transactions, which result in legal consequences.

If the above conditions are met, we can declare that the parties follow the same business model to achieve common business goals, and that the differences lie only in the technical infrastructure they use to implement their business model. If any of the above requirements is not met, there is no sufficient business foundation for these parties to cooperate, even in non-electronic form.

A successful completion of this step means that we have established a common business context for both parties. We have also identified the events that need to occur, and the collaborations between agents that support these events. This in turn determines the transactional boundaries for each activity.

(NOTE: this section definitely needs more substance…)

This business context model will help us to make decisions in cases when a strict one-to-one mapping on the technical infrastructure level is not possible. It will also help us to decide what kind of compensating actions are needed in case of failures.

2.2. Business Process Mediation *(to be completed)*

2.2.1. Business Process Models

The elements of Business Process models describe the major steps in the interaction scenario that need to be performed in order to successfully execute the mutual commitments. In this step we identify the business transaction boundaries, and the activities that need to be performed in order to fulfill them, or what kind of activities are needed to rollback (or compensate) for failed transactions.

A *business process* (according to [REA],[ebXML],[UMM]) consists of a sequence of *business activities* performed by one business partner alone, and *business interface activities* performed by two or more business partners. In the ECIMF methodology we will be interested primarily in aligning the *business interface activities*, although in most cases understanding both types of activities is needed in order to understand the business process constraints. These activities realize the collaborations between the involved business Agents, and they also support the economic exchanges identified in the Business Context models. Further, we will use the term BusinessActivity to mean the business interface activity.

In this model, each collaboration task is further decomposed into *business activities*, which may involve one or more *business transactions*, which in turn are executed with help of *business documents* and *business signals*.

2.2.1.1. Business Process Meta-model

Here are more detailed descriptions of each of the modeling elements:

- *BusinessProcess*: contains one or more economic exchanges, which in turn contain two or more BusinessCollaborationTasks each.
- *BusinessCollaborationTask*: a logically related group of BusinessActivities, which realizes the collaboration between two Agents in a given Event.
- *BusinessActivity*: a business communication (initiated by a requesting or responding business Agent). BusinessActivities may lead to changes in state of one or both parties.
- *BusinessTransaction*: a set of BusinessDocuments and BusinessSignals exchanges between two parties that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded (possibly some additional compensating actions need to be taken as well).
- *BusinessDocument*: a message sent between partners as a part of information exchange, which contains business data (payload).
- *BusinessSignal*: a message that is transmitted asynchronously back to the partner that initiated the transfer of business process execution control (by sending a BusinessDocument), which doesn't contain any business data, but instead just signifies acknowledgement or error condition.

*(NOTE: probably this meta-model needs to be harmonized with UMM or eBTWG, but there is also a need to provide a **simplified** version…)*

2.2.1.2. Business Process Models

Business processes are most often modeled using UML activity diagrams (or similar notation), where each diagram represents one of the collaborations. This view relates to the Business Context view in the following way:

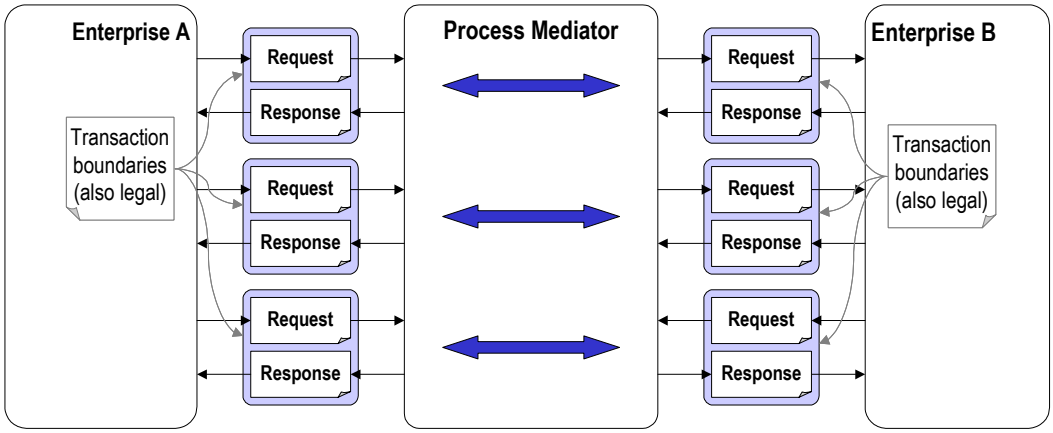
- The collaboration links between Agents correspond 1:1 to BusinessCollaborationTasks. This means that for the typical economic exchanges

there will always be two BusinessCollaborationTasks – one for the "give" part, and one for the "take" part of the exchange.

In addition to that, the BusinessProcess view enhances the understanding of the Business Context, because it allows us to correlate various Events that are dependent on each other even if they don't belong to the same economic exchange (e.g. consumption of resources, replenishment and sales tasks are dependent on each other, but they are not likely all to be part of the same BusinessCollaborationTask between two specific partners).

2.2.1.3. Business Collaboration Tasks and Business Transactions

- The BusinessCollaborationTasks support the execution of the BusinessEvents identified in the previous step. There should be as many Business Tasks as many collaboration links were in the Business Context models.
- BusinessEvents are realized by one or more BusinessTransactions. Consequently, BusinessCollaborationTasks consist of one or more BusinessTransactions
- BusinessCollaborationTasks are represented as UML activity diagrams, showing the activities of both collaborating agents. These diagrams usually contain two parts (swimlanes): one for the requesting (initiating) party, the other for the responding party. The diagrams should also contain the messages passed between the parties.



2.2.2. Business Process Mediation Model

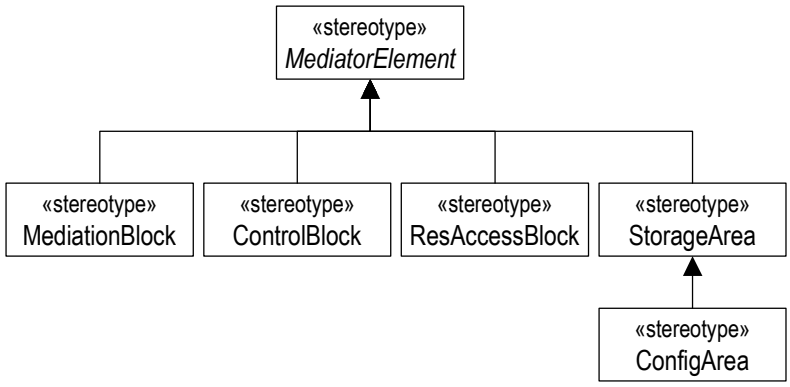
The mediation between two different conversation patterns (which may involve different low-level technical transactions) needs to be designed and managed in a Business Process Mediation model.

2.2.2.1. Business Process Mediation Meta-model

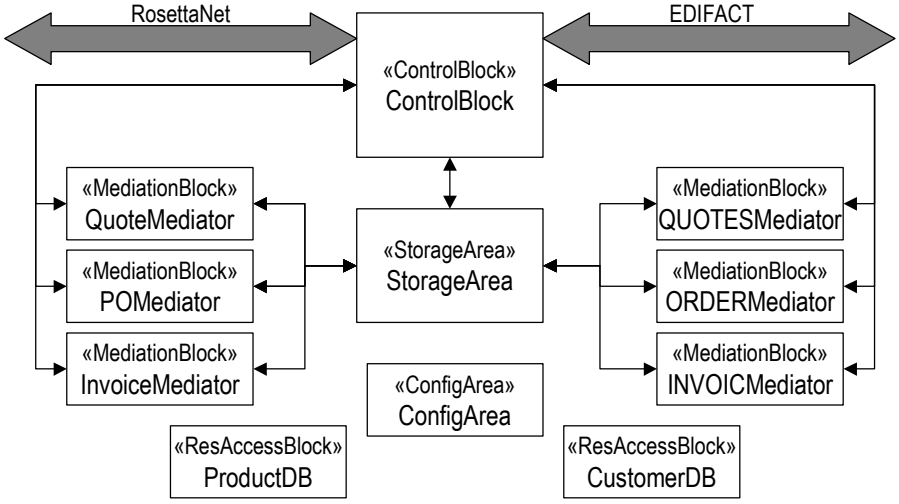
(NOTE: the working hypothesis is that the model elements will be responsible for reconciling concrete aspects of the conversations. The current idea of the internal structure of the model is as follows:

- *there will be mediation blocks handling the flow of each business transaction – totally the number of distinct business transactions on one side plus the number of distinct business transactions on the other side. These mediation blocks will be responsible for handling the details of conversations according to a given framework, within the boundaries of one specific transaction.*
- *there will be resource wrapper blocks, allowing for uniform access to external resources*
- *there will be one controlling block, responsible for managing the overall flow of transactions.*
- *there will be a common storage area, which any mediation block or the controlling block can access in order to store intermediate data – such as previous messages*
- *similar to that, there will be a configuration area accessible to all blocks, containing the configuration parameters.*

To summarize, the following diagram presents the meta-model:



And the diagram below presents a mediation model example:



Again, this is just a working hypothesis – any comments are much welcome!)

2.2.2.2. Checking the task alignment

(to be completed…)

2.2.2.3. *Creating the Mediation elements*

(to be completed…)

The process of building this part of the integration model is very closely related to the Semantic Translation, because very often a semantic correspondence needs to be established between the concepts, transactions, messages and information elements.

2.3. Semantic Translation *(to be completed)*

Figure 8 presents the idea of the semantic translation and the reason why it's a required step in solving the interoperability puzzle. In general, the concepts underlying the foundations on which the IT infrastructures are built, differ between not only the industry sectors, or geographical regions, but even between each company within the same sector. This phenomenon – of different semantics, and different ontologies – causes many complex problems in the area of system integration, and in the area of e-commerce integration specifically.

One of the most common cases that require semantic translation to be performed is when each business party uses a different product catalogue (this situation is sometimes referred to as the "catalog integration", or "catalog merging" problem).

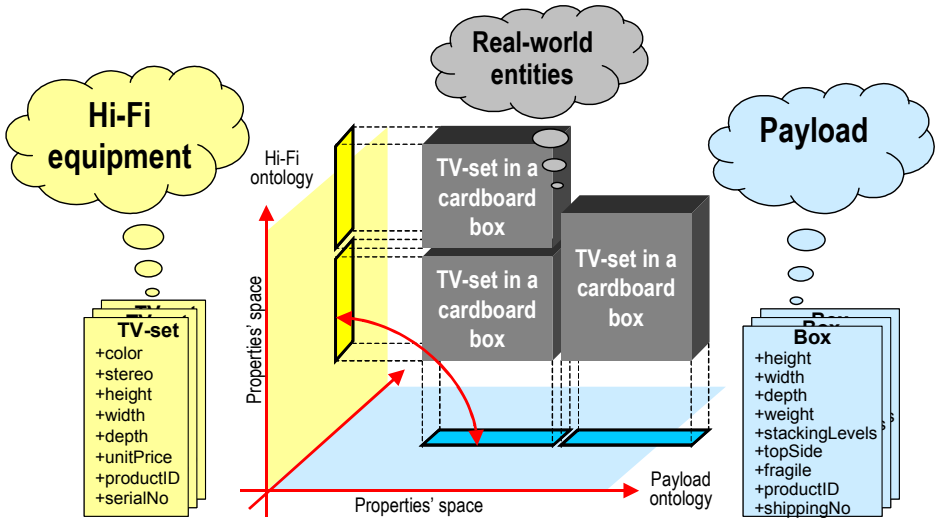


Figure 8 Mapping concepts from different ontologies.

In the example presented on Figure 8, a real-world entity - TV-set in a cardboard box - is represented very differently in two domain ontologies - the ontology of Hi-Fi equipment, and the transportation ontology. Although two representations may refer to the same real entity, in order to communicate that fact to the users of the other ontology we need to perform a semantic enrichment, in order to determine the proper classification of the concept in the other ontology.

What's even worse, we may discover (as is often the case) that the concepts overlap only partially, and the conditions under which they match the concepts from the other ontologies are defined by complex formulas, dependent potentially on several factors such as values from external resources, time, geographical region etc. In this case, the physical dimensions of the TV-set concept are confusingly homonymous to the dimension properties of the Box concept, but in the first

case they refer to the TV-set chassis, and in the second case they refer to the cardboard box dimensions. Furthermore, the Box dimensions might be allowed to take only certain discrete values (e.g. according to a normalized cardboard container types), so in order to determine their values based on the information available in the TV-set concept, it is necessary to access some external resource (a cardboard box catalogue).

2.3.1. Describing the semantic mapping

2.3.1.1. Semantic Translation meta-model

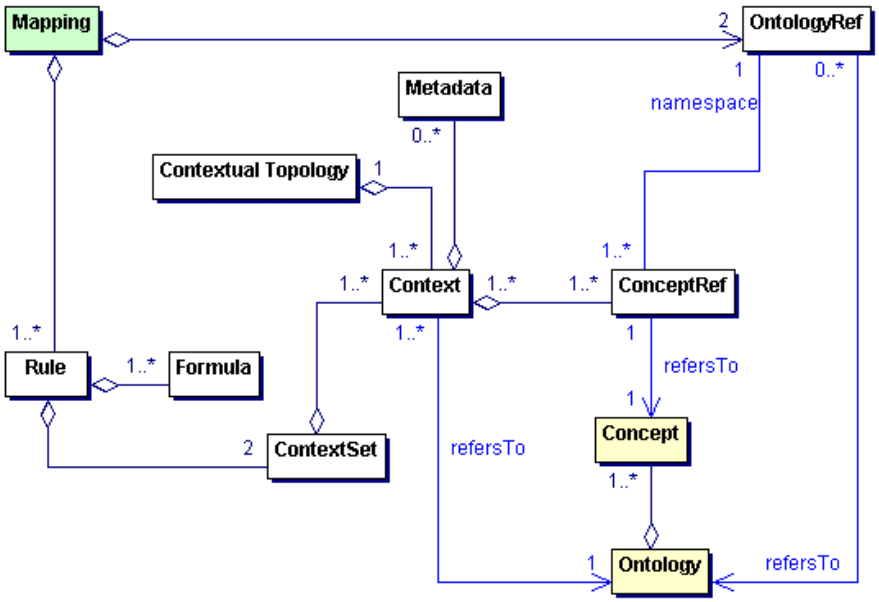


Figure 9 Semantic Translation meta-model

Figure 9 presents the meta-model for capturing the rules of semantic correspondence between concepts belonging to two different ontologies. This meta-model has been developed based on the principles of contextual navigation, which means that the proper understanding of a concept requires considering the context in which it occurs.

Furthermore, the translation rules (mappings) only refer to the original ontologies and concepts, which means that the original definitions, constraints, relationships and axioms are not recorded in the translation rules, but are only represented by unique identifiers (references). The reason for this is that especially in the e-commerce scenarios these source ontologies are usually completely separate, and maintained by separate organizations. These two concepts (Ontology and Concept) are accordingly marked as “external” in the list below.

- *Ontology*: the original full domain ontology (external)
- *Concept*: concepts defined in the original Ontology (external)
- *Mapping*: a top-level container for the semantic mapping rules, applicable to a pair of ontologies, as specified by the *OntologyRef*-s.
  (The *Mapping* is marked green in the diagram as the starting point for reading the whole meta-model.)

- *OntologyRef*: a URN uniquely identifying the referred ontology (possibly allowing to access it remotely).
- *ConceptRef*: a namespaced reference to individual *Concept*-s defined in the original *Ontology*. A URN, which possibly allows to access remotely the concept definition in the original ontology.
- *Context*: built on the basis of the original *Ontology* (refersTo), consists of related concepts represented by *ConceptRef*-s, which are considered relevant to the given transformation rule (the exact and full relationship of the *Concept*-s is defined in the original ontology - *Context* captures just the fact that they are related for the purpose of mapping).
- *ContextSet*: a group of one or more *Context*-s referring to the same *Ontology*.
- *Rule*: a rule that defines how to translate between the concepts in a *ContextSet* from one ontology, to the corresponding concepts in a *ContextSet* from the other ontology. A *Rule* consists of exactly two *ContextSet*-s, each one referring to respectively one of the ontologies, and a set of *Formula*-s, which define the valid transformations on these *ContextSet*-s.
- *Formula*: a formal expression defining how translation is performed between concepts from the source *ContextSet* to those in the target *ContextSet*.

The reason for defining the *ContextSet*, in addition to *Context*, is that probably we would like to use concepts from several contexts belonging to a single *Ontology*, and map them to several contexts in the other. But at the same time there is a requirement to state explicitly that we always map between exactly two different ontologies.

2.3.1.2. Algorithms for discovering the semantic correspondence

(Many exist, none ideal or fully automatic. There is a need to use several in parallel, plus heuristics…)

2.3.1.3. The Formula language

(Needs to be more complex than first-order logic. Probably a full-fledged programming language, e.g. XSLT, JavaScript, XQuery, etc.)

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, JavaScript, XSLT, XQuery and Python.

2.3.2. Example model

Below is an example of (part of) the model built with the Semantic Translation meta-model.

(NOTE: for now the Formula language is unspecified, and in this example a JavaScript-alike language was used).

```
Rule:rule1
| | |
| | +-- ContextSet:set1 {Ontology 1}
| |       \Context:Party
| |       \Context:Address
| |       \Context:PartyIdentification
| |       \Context:Name
| +--ContextSet:set2 {Ontology 2}
|         \Context:Agent
|         \Context:Location
|         \Context:Name
|         \...
\Formula:formula1
|   \body: "set2.Name = set1.Name"
\Formula:formula2
|   \body: "set2.Location.Address.Street1 =
                  set1.Address.Street;
            set2.Location.Address.Street2 =
                  concat(set1.Address.Zip, set1.Address.City);"
\Formula3:Formula ...
        .....
```

(NOTE2: There is also a working hypothesis that one could use a rule of thumb to treat the ebXML aggregate core components as Contexts, and most primitive core components as concepts - but this needs further research, and discussions with the eBTWG community.)

2.4. Syntax Mapping (to be completed)

2.4.1. Data element mapping

(using the semantic mapping rules. Syntax mapping is often preformed with XSLT, plus optionally the straightforward wrappers for non-XML formats)

2.4.2. Message format mapping

(see above. Additionally, it needs to ensure the well-fomedness and validity of messages according to the format specifications.)

2.4.3. Message packaging mapping

(ebXML CPP/CPA ?)

2.4.4. Transport protocol mapping

(ebXML CPP/CPA ?)

2.5. MANIFEST recipes

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 ECIMF models, rules and definitions.

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 publicly 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.

3. The ECIMF-compliant runtime 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.

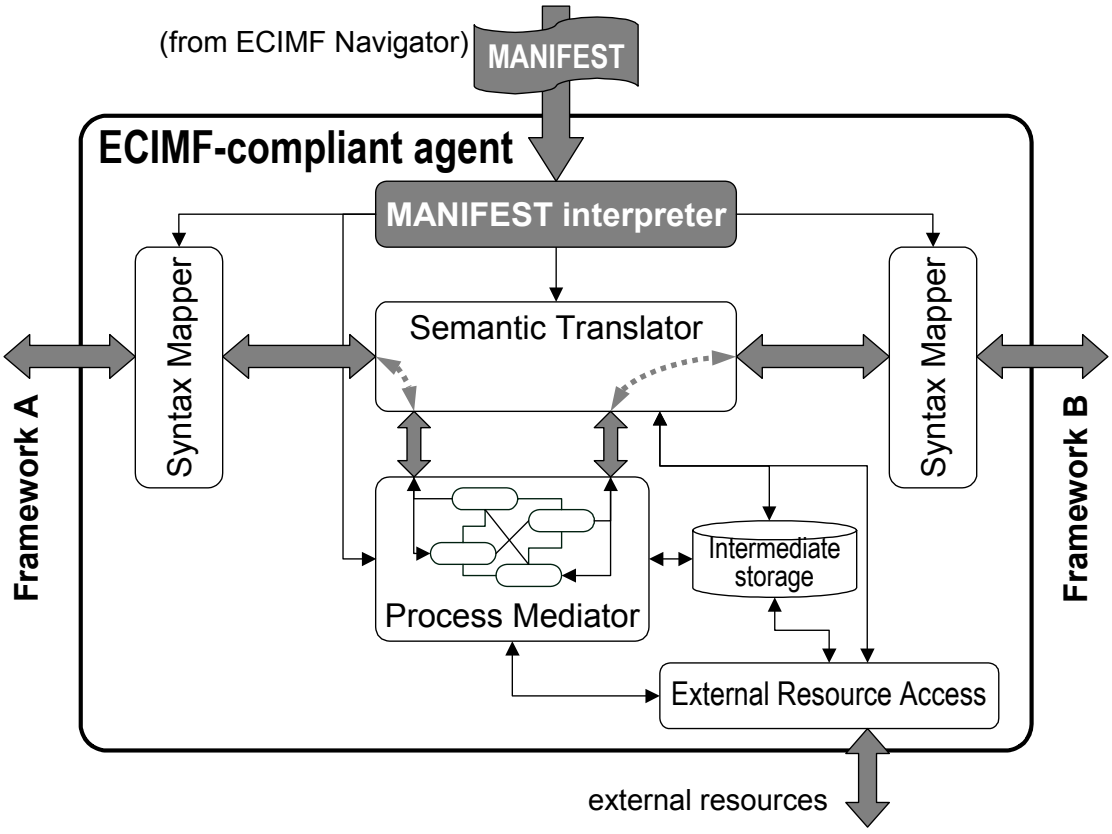


Figure 10 Example of ECIT (ECIMF-compliant agent) facilitating message exchange.

Figure 10 presents a block diagram of an ECIMF-compliant integration agent. The data flow (represented by thick gray arrows) goes first through the low-level data format adapters (named "Syntax Mappers"), then proceeds to the "Semantic Translators" module, and finally is controlled by the "Process Mediator". The "MANIFEST Interpreter", which uses the information provided in the "MANIFEST" specification prepared in the ECIMF Navigator, configures all these building blocks.

It is important to note that in this model, the ECIMF-compliant agent operates not only on the currently arrived data in the current message, but also uses some historical data stored in the intermediate storage, as well as the data available from external resources.

3.1. Syntax Mapper

The syntax Mapper is responsible for translating the message format and transport protocol to/from the internal model representation, which is then used by other modules. This could involve e.g. translating from EDI to XML, and then

building an XML Document Object Model (DOM) tree as a representation of the incoming message. Further processing in the Semantic Translation module processes that internal model representation.

3.2. Semantic Translator

This module is responsible for changing the information model according to the translation rules, so that the information contained in the original message is understandable for the other party according to its (different) data model and meaning. This module operates only on the internal representation of the data.

3.3. Process Mediator

This module aligns the conversational patterns of each of the frameworks. It should be noted that this might require working not only with the currently received data in the message, but also with some historical data in the context of the same conversation. Also, there may be a need for using a given piece of information later during the same conversation, as specified by the differing message formats. For these reasons, the process mediator needs to use an intermediate storage, in which the data related to the context of current conversation may be kept.

4. Frameworks Integration Guideline

The main objective of the ECIMF project is to provide clear guidelines and methodologies for building interoperability bridges between different incompatible e-commerce standards.

This section presents a general guideline to solving this issue in case of two incompatible e-commerce frameworks F1 and F2. Annex 1 gives additional supporting information.

The guideline has been divided into several steps, to be performed sequentially and iteratively, as needed. The steps follow the methodology described in the previous section – the layers on the top are addressed first, since they give the broadest context necessary for understanding of the lower-level data transformations. The successful completion of all steps will result in a set of interoperability rules, enforced by a framework mediating agent, which will allow parties using different frameworks to cooperate towards common business goals.

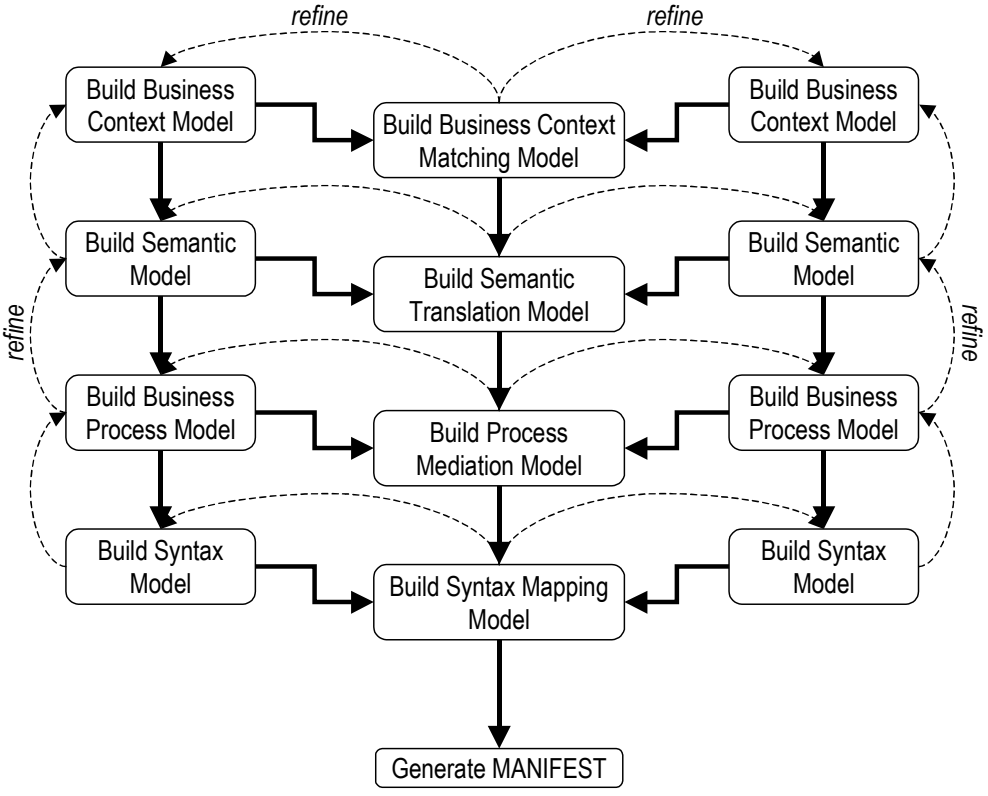


Figure 11 The process of modeling the integration recipes between two e-commerce frameworks.

The guideline has a modular structure, reflected in the fact that in each step several so-called *alternative procedures* have been defined. Each *alternative procedure* refers to a well-defined unit of work that needs to be done (a part of integration step), and allows you to replace or extend the approach suggested for that step with other methods of your choice, as long as they provide you with similar results as the input to the next step. The boundaries of each alternative procedure are clearly marked, and the input/output deliverables are specified.

You can also find a common meta-model defined in each of the steps, which serves as a common vocabulary (shared ontology) for understanding the incompatible frameworks.

One important thing to note here is that the integration modeling between two frameworks is asymmetric, i.e. the integration model will usually contain two elements that refer to the same individual model elements, but defined differently depending on the direction in which the data is traveling.

The subsections below present the details of the guideline.

4.1. Analysis of the Business Context Matching

4.1.1. Creating Business Context Models

A **business context model** shows a concrete business scenario expressed with the use of economic modeling elements, e.g. those found in the REA meta-model. We suggest using the following standard UML diagrams for that purpose:

- Class diagrams to show the specific types of entities involved.
- Collaboration diagrams to show a specific scenario populated with specific instances of participating entities.
- Value-chain diagrams (REA process diagrams), to clearly define the flows of resources, and how they depend on the collaboration between partners.

For examples of such business context models, please see the ECIMF-POC document.

4.1.2. Checking the Business Context Matching Rules

Each of the context matching rules needs to be checked, and any additional requirements or assumptions made need to be recorded, so that they can be used to understand the interactions in the lower layers of the ECIMF model.

Below is a table that summarizes this step of the guideline:

Business Context Matching	
Input	Traditional business knowledge, legal agreements between partners, industry specific rules, legal constraints, specific business goals, common business practices and codes of conduct
Output	Two Business Context Models for the integration scenario, defined in a set of UML diagrams (class, collaboration, activity), and an analysis of their matching (and any additional requirements on which the matching depends).

4.2. Creating the Business Process Mediation Model

4.2.1. Creating the Business Process models

A **business process model** shows concrete business collaboration, expressed as series of business activities and transactions between the partners. We suggest using the standard UML activity diagrams for that purpose, one diagram for each collaboration.

4.2.1.1. Identify the Business Collaboration Tasks

For each collaboration link in the Business Context diagram, a Business Collaboration Task is created.

4.2.1.2. Identify the Business Transactions

For each collaboration, and for each Agent, the business transactions are discovered and described. Since the Agents possibly use different frameworks, there might be different transactions expected even for the same collaborations.

For examples of such business process models, please see the ECIMF-POC document.

4.2.2. Creating the Mediation model

(NOTE: describe how the process mediation model can be created, using concepts from the Mediation meta-model.)
(NOTE2: the relationship to eBTWG BOT's [Business Object Types] need to be analyzed. BOT's define not only the class (+properties), but also the behavior, state and methods. As such, they are the best candidates to provide the intermediate internal model, and the problem of process mediation could be reduced to the problem of reconciling the state diagrams of the key BOT's).

Below is a table that summarizes this step of the guideline:

Business Process Mediation	
Input	Business Context models, other information on business processes supporting the business context, semantics of the business processes (obtained in the next step), etc.
Output	Business Process Models, Business Process Mediation Model for the integration scenario, defined in a set of diagrams (activity/business process, ECIMF process mediation diagram)

4.3. Creating the Semantic Translation Model

4.3.1. Acquiring the source ontologies

(NOTE: describe the process of discovering the ontologies from e-commerce standards, best practices, business rules etc…)

4.3.2. Selection of the key concepts

(NOTE: describe how the business context and business process models help to determine the key concepts …)

4.3.3. Creating the mapping rules

(NOTE: describe how the mapping rules can be created, based on one of the alternative procedures …)

Below is a table that summarizes this step of the guideline:

Semantic Translation	
Input	Two source ontologies, obtained from formal specifications, UML models, textual descriptions, knowledge of domain experts etc.
Output	Semantic Translation Model, containing rules for equivalence of the key concepts.

4.4. Creating Syntax Mapping Model

4.4.1. Data element mapping

(NOTE: describe how the external formats can be mapped to internal representation …)

4.4.2. Message format mapping

(NOTE: describe how the message well-formedness rules can be satisfied. This may involve proactive "asking" for more information in order to satisfy the demands of a given message format…)

4.4.3. Message packaging mapping

(NOTE: describe how the message packaging [encoding, charset, MIME, etc] can be aligned)

4.4.4. Transport protocol mapping

(NOTE: describe how the transport protocol parameters need to be defined.)

Below is a table that summarizes this step of the guideline:

Syntax Mapping	
Input	Semantic Translation Model, simple mapping of primitive data types, external resources to be used.
Output	Syntax Mapping Model, containing the exact mapping of data elements, message formats, packaging and transport protocols.

For additional details, and more information on alternative procedures available for each of these steps, please refer to the Annex.

5. References

[UMM]: *Unified Modeling Methodology*; UN/CEFACT TMWG N090R9.1; available from: UN/CEFACT TMWG. A copy of the draft can be also found at: http://www.ecimf.org/doc/other/TMWG_N090R9.1.zip

[ebCDDA]: *Core Components Discovery and Analysis*; ebXML, May 2001; available from: http://www.ebxml.org/specs/ebCDDA.PDF

[ccDRIV]: *Catalog of Context Drivers*; ebXML, May 2001; available from: http://www.ebxml.org/specs/ccDRIV.PDF

[CID52]: *Conceptual Navigation and Multiple Scale Narration in a Knowledge Manifold*; Ambjörn Naeve; KTH, 1999; available from: http://cid.nada.kth.se/sv/pdf/cid_52.pdf

[OB00]: *Ontology-Based Integration of Information — A Survey of Existing Approaches*; H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner; University of Bremen, 2000; available from: http://www.tzi.de/buster/papers/SURVEY.pdf

[SAGV00]: *Semantic Translation Based on Approximate Re-Classification*, Heiner Stuckenschmidt, Ubbo Visser; University of Bremen, 2000; available from: http:// www.tzi.de/buster/papers/sagv-00.pdf

[SW00]: *A Layered Approach to Information Modeling and Interoperability on the Web,* Sergey Melnik, Stefan Decker; Stanford University, 2000; available from: http://www-db.stanford.edu/~melnik/pub/sw00/sw00.pdf

Annex 1 – Additional supporting materials for the Frameworks Integration Guideline

(non-normative?)

(NOTE: the parts in Times New Roman require still significant amount of work – both editing and conceptual. The parts in Arial seem to be mostly OK… The notes in italics mark the areas requiring additions and discussions.)

1. Business Context Matching

Business Context Matching	
Input	Traditional business knowledge, legal agreements between partners, industry specific rules, legal constraints, specific business goals, common business practices and codes of conduct
Output	Two Business Context Models for the integration scenario, defined in a set of UML diagrams (class, collaboration, activity), and an analysis of their matching (and any additional requirements on which the matching depends).
Alternative Procedures	
REA	REA ontology [REA], [REAont]
UMM	Business Requirements View in Chapter 9.2 of [UMM] (can be considered a specialized subset of REA)
EbXML	Business Process Analysis Worksheets and Guidelines [bpWS] (which are also based on REA principles)
SimpleREA	Described below.

1.1. Creating Business Context Models

Simple REA

Here we describe a simplified procedure useful for modeling of simple business cases (based on subset of REA, with relationships to UMM BRV and BTV; it should also be compatible with ebXML). As a result of the pragmatic process described below, you will create an economic exchange diagram, which provides a high-level overview of the parties involved in the business activities; and a value-chain diagram which puts this exchange in a context of the whole enterprise.

1. Economic Exchange Diagram

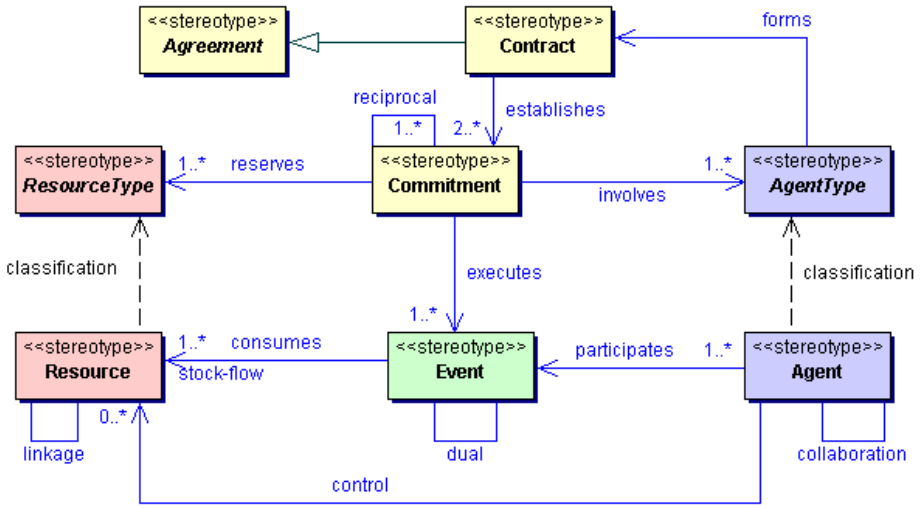
1.1. Meta-model

Describe the entities involved in the business case at hand, using the following terms (represented as UML stereotypes):

- ***AgentType***: the role that a business partner plays in the scenario (e.g. buyer, seller, payer etc…). This is an abstract classification of the concrete Agents involved.
- ***Agent***: if needed, specifies a concrete representative of a business party, which fulfills a given partner type (e.g. a sales clerk [= seller], a customer [= buyer]).
- ***Agreement***: an agreement is an arrangement between two partner types that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration scenarios, etc.) A special kind of agreement (contract) commits partners to execute specific events, in which economic resources are exchanged.
- ***Commitment***: an obligation to perform an economic event (i.e. transfer ownership of a specified quantity of a specified economic resource type) at some future point in time.
- ***EventType***: an abstract classification or definition of an economic event. E.g. rental, service order, direct sales, production (of goods from raw materials), etc …
- ***Event***: an economic event is the transfer of control of an economic resource from one partner type to another partner type. Examples would include the concrete sales, cash-payments, shipments, leases, deliveries etc. Economic Events usually cause changes in the state of each partner type (so called business events). Therefore they are directly related to (and determine) the transaction boundaries.
- ***ResourceType***: an economic resource type is the abstract classification or definition of an economic resource. For example, in an ERP system, ItemMaster or ProductMaster would represent the Economic Resource Type that abstractly defines an Inventory item or product. Forms of payment are also defined by economic resource types, e.g. currency.
- ***Resource***: if needed, specifies a quantity of something of value that is under the control of an enterprise, which is transferred from one partner type to another in economic events. Examples are cash, inventory, labor service and machine service. Contracts deal with resource types (abstract definitions), whereas events deal with resources (real entities). You may use this distinction if needed.

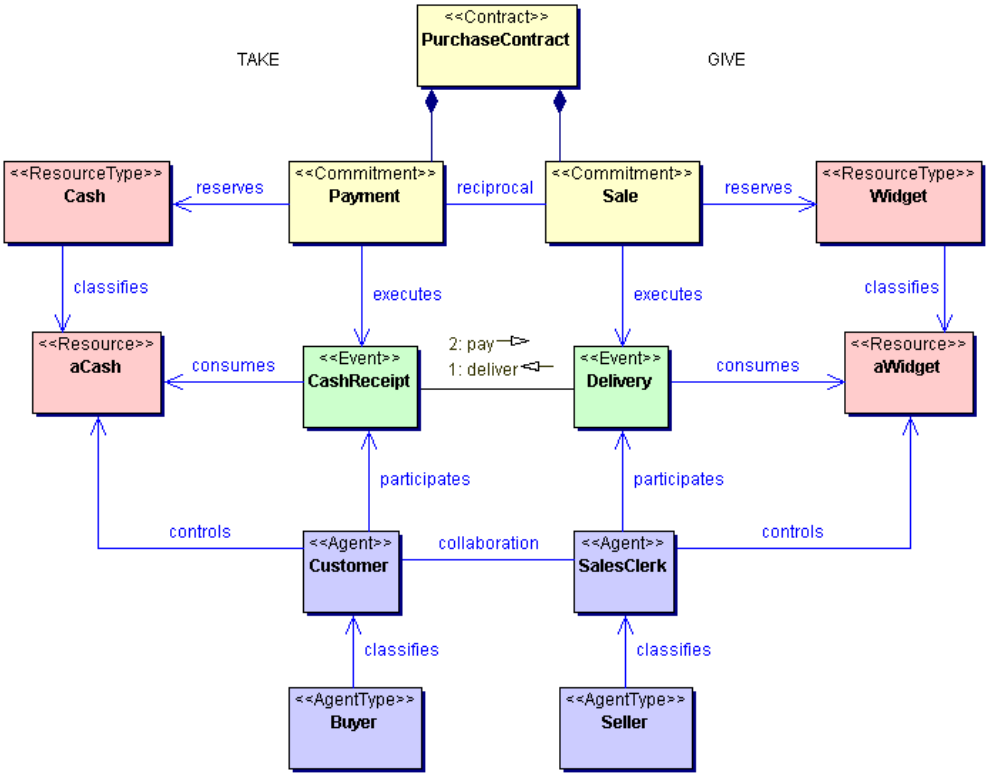
1.2. Meta-model and constraints

The meta-model for building the economic exchange diagrams is presented on the figure below:



The entities have been color-coded. The collaboration between Agents is realized with the BusinessTasks (collaboration protocol), which may be represented as UML activity diagrams.

1.3. Model example



The coloring schema on this diagram corresponds to that on the meta-model diagram.

Note: this diagram shows instances (concrete entities) of types specified above in the meta-model diagram. This is indicated by the UML stereotypes (labels in guillemots). Notice the two messages exchanged in this model – the first is to deliver, the second to pay (but it may be the other way around – an advance payment). This diagram helps us to identify the business transactions (in this case: {deliver, pay}), and also shows us the timing constraints (in this case: first deliver, then pay).

(NOTE: any useful real-life scenario would be more complicated. It could e.g. contain a catalog lookup, negotiation, shipment, blanket agreement, etc… This diagram serves therefore only as an illustration of the approach).

1.2. Checking the Business Context Rules

#1 Complementary roles

Parties need to play complementary roles (e.g. buyer/seller)

#2 Expected resources

The resources expected in the exchanges need to be equivalent to the ones expected by the other partner (e.g. cash for goods)

#3 Timing constraints

The timing constraints on events (commitment specification) need to be mutually satisfiable (e.g. down payment vs. final payment)

#4 Transaction boundaries

The sequence of expected business transactions needs to be the same (even though the individual business actions may differ)

2. Business Process Mediation

Business Process Mediation	
Input	Business Context models, other information on business processes supporting the business context.
Output	Business Process Models, Business Process Mediator Model for the integration scenario, defined in a set of diagrams (activity/business process, ECIMF process mediation diagram)
Alternative Procedures	
UMM + ECIMF-PM	UMM-BOV, and the ECIMF Process Mediation Model
UML-EDOC + ECIMF-PM	UML-EDOC, and the ECIMF Process Mediation Model
EbXML + ECIMF-PM	Business Process Specification Schema, and the ECIMF Process Mediation Model

2.1. Creating the Business Process Models

(to be completed...)

2.2. Creating the Business Process Mediation model

2.2.1. Check the Business Tasks alignment

- Identify **request and response messages**.
  (NOTE: this step will benefit from information collected in BOV and FSV models, if available (cf. [UMM]))
- Determine **legal obligations** boundaries: which interactions and messages bring what legal and economical consequences. This can be established based on the relationship to the business context diagram.
  (NOTE: needs more substance…)
- Determine the **business transaction boundaries**, rollback (compensation) activities and messages for failed transactions. The transaction boundaries can be better identified with the help of the business context diagram.
  (NOTE: needs more substance…)
- **Identify the differences** in message flow, by comparing message flows between requesting/responding parties for each business task.

2.2.2. Create the Mediation Elements between Business Tasks

- **Missing messages/elements**: are those that are present in e.g. Framework 1 business task B_x (we use the notation $F_1(B_x)$ for that), but don't oc-

cur in the corresponding $F_2(B_y, B_z, \ldots)$. This is also true about the individual data elements, which may become available only after certain steps in the conversations, different for each framework. These messages and data elements will have to be created by the mediator, based on already available data from various sources, such as:

- previous messages
- configuration parameters
- external resources

and sent according to the expected conversation pattern.

- **Superfluous or misplaced messages/elements**: are those that don't correspond directly to any of the required/expected messages as specified in the other framework. Also, they may be required to arrive in different order. The mediator should collect them (for possible use of information elements they contain at some later stage) without sending them to the other party, or change the order in which they are sent. The business context diagram will help determine what kind of re-ordering is possible without breaking the transaction boundaries (it should be possible to change the order within the transaction boundaries without breaking their semantics, but not across them).
- **Different constraints** (time, transactional, legal…): this issue is similar in complexity to resolving the semantic conflicts (see below), and a similar approach could be taken.
  (NOTE: namely???)

3. Semantic translation *(to be completed)*

(NOTE: needs to be harmonized with the methodology section!!!)

- **Identify the key concepts** in use for message exchanges conducted according to each framework, within the context of the selected corresponding business tasks:
  - **For each message** in B_i identify the key indispensable information elements that decide about the success of the information exchange from the business point of view in each of the frameworks:
    $M_i(E_1, E_2, \ldots, E_n)$
  - **For each message** M_i in B_i, based on the framework model, identify the key concepts that these information elements represent. In terms of OO and UML modeling, use the information collected in the previous step to build an object diagram, where instances of classes represent the key concepts (perhaps already identified in the formal framework description) and properties take the values from the message elements:
    $M_i(C_1(E_1, E_2, \ldots), C_2(E_m, E_n, \ldots), \ldots, C_n(E_x, E_y, \ldots))$
    This notation means that each message M_i contains a set of key concepts (classes) – information elements, which decide the meaning of the message.
  - **Collect the key concepts** in a unique set:
    $F_1(C_1, C_2, \ldots, C_n, \ldots, C_x, \ldots, C_z)$

  (NOTE: this is a bottom-up approach. Needs to be re-worked to better reflect the overall top-down approach).
  (NOTE 2: this step corresponds to the process of building conceptual topology of frameworks F1 and F2, which are sets of conceptual neighborhoods [CID52]).
- **Collect more semantic data** about each concept, as expressed by each framework's specifications, in a form of properties and constraints:
  $C_i(p_1, p_2, \ldots, p_m, c_1, c_2, \ldots, c_x)$
  We introduce the notation P_i to denote a property with its accompanying constraints. Therefore we may express the above as follows:
  $C_i(P_1, P_2, \ldots, P_m, c_n, \ldots, c_x)$
  These additional semantic data will probably point to some obvious generalizations, which in turn may lead to reduction of the set of unique concepts.

(NOTE 1: The steps detailed above lead to creation of framework ontologies – or, in the language of [UMM], Lexicons with core components. Similarly, the process described below corresponds to finding a translation between ontologies [OB00] – although, since the ontologies are built from scratch here, the approach to use shared vocabulary may provide useful reduction in complexity (cf. [OB00]). The latter approach is similar to the process described in [ebCDDA] for discovery of domain components and context drivers).
(NOTE 2: the Business Operational View [UMM] model of the frameworks, if available, is a very appropriate source for this kind of information)
(NOTE 3: two concepts $F_1(C_x)$ and $F_2(C_y)$ may in fact represent one real entity – however, due to the different contexts in which they are described they may appear to be non-equal. Such cases will be resolved in the following steps)

- **Generate hypotheses about corresponding concepts** in the other framework:
  - Concepts are likely to correspond if they:
    - have similar properties
    - are similarly classified
    - play similar roles (similar relationships with other concepts, occur in similar contexts)
- **Test each hypothesis:**

Semantic Translation	
Input	Ontologies for each framework, containing the key concepts
Output	Semantic Translation rules, defining the correspondence between the key concepts
Alternative Procedures	
BUSTER	Approximate re-classification (described below)
Subsumption	Check the constraints on the properties, describe the differences in property specifications (such as scale, allowed values, code lists, classification) and check the correctness of classification based on the following criteria: • The **necessary conditions** for concept $F_i(C_x)$ is set of values/ranges of some of its properties that are true for all instances of that concept. Therefore, if a concept C_y doesn't display them, it cannot be classified as C_x. Necessary conditions help to rule out false correspondence hypotheses. • The **sufficient conditions** for concept $F_i(C_x)$ is a set of properties and constraints, when met automatically determine the concept classification. Sufficient conditions help us to identify the concepts that surely correspond because they show all sufficient conditions. Example: "TV-set" meets sufficient conditions for being a "house appliance". However, it fails to meet the necessary conditions for a "cleaning house appliance".
Anchor-PROMPT	
Cupid	
MOMIS	
Ontomorph	
Upper-level ontology labeling	*(using terms from upper-level ontology to label the concepts, and then prepare translation formulas based on the formal subsumption algorithms)*

Approximate re-classification
If the above steps result in well-defined rules of correspondence for most cases of the observed concept occurrence, the hypothesis can be considered basically true. It is probably not feasible to strive for exact solution in 100% cases – we may allow certain exceptions. There are several ways to determine the level of proximity:

- **Rough classification:** the concept definition can be treated as having its upper and lower bounds. The upper bound (the most precise) is necessary conditions, and the lower bound (the most general) is the sufficient conditions. We may declare that $F_1(C_x) \rightarrow F_2(C_y)$ even when necessary conditions are not met, but sufficient ones are.
- **Probabilistic classification:** we can determine (based on e.g. available pre-classified data sets) the significance of each property on the result of classification, and so calculate the probability of entity belonging to a specific class.
- **Fuzzy classification:** for each property we define a fuzzy rule, which describes the level of similarity of the tested property. Then, the best match is defined when maximum number of rules gives positive results.

- **Other hypotheses**: if the hypothesis cannot be proven with a sufficient degree of certainty, other hypotheses need to be formulated and tested.
- **Possible difficulties** that may arise:
  - There is **no corresponding** concept: may be there are too many unknown properties to determine the corresponding concept in F_2, because in the context of F_1 they were irrelevant. In this case, the information required to find $F_2(M_x(C_y))$ needs to be supplied from elsewhere, based on properties of the

real entities that $F_1(M_i(C_j))$ and $F_2(M_x(C_y))$ refer to - we need to provide more semantics about the concepts than what is found in the framework specifications (usually from a human expert).

- There are **many corresponding** concepts, depending on which property we choose: we could arbitrarily choose the one that plays the most vital role from the business point of view – and choose which properties decide that an instance of a concept in F1 could be classified as an instance of corresponding concept in F2:

  $$F_1(C_x(P_i)) \rightarrow F_2(C_y(P_j))$$

  See also the section above on probabilistic classification.
- The **conflicts in property** constraints cannot be easily resolved. This case calls for help from the domain expert.

- **Describe the rules and exceptions** (if any), and in what contexts they occur.

(NOTE: there are three ways to address this problem, according to [OB00]:

- *Create a single global ontology, which will include concepts from both frameworks. Not feasible for even moderately complex cases.*
- *Create mappings between concepts in ontologies (this is the approach suggested above, although [OB00] warns again that it leads to very complex mappings)*
- *Using shared vocabulary, re-build the ontologies from scratch – the result will be somewhat automatically aligned. Then, prepare the translation rules, which should be now much simpler.)*

4. Syntax translation (to be completed)

- **Data element mapping**
  (NOTE: describe how the external formats can be mapped to internal representation …)

- **Message format mapping**
  (NOTE: describe how the message well-formedness rules can be satisfied. This may involve proactive "asking" for more information in order to satisfy the demands of a given message format…)

- **Message packaging mapping**
  (NOTE: describe how the message packaging [encoding, charset, MIME, etc] can be aligned)

- **Transport protocol mapping**
  - **Align packaging and transport** protocols, based on the specifications in each framework.
- *(to be continued…)*