E-Commerce Frameworks Integration Guideline

Draft 0.3
September 25, 2001

NOTE: the value of this document is a work-in-progress research, for discussion among project members – no final conclusions should be drawn from it. Please provide your comments and ideas to the project participants, or directly to the project Chair (Andrzej Bialecki, abial@webgiro.com).

Introduction

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 document describes an experimental step-by-step Guideline to solving this issue in case of two incompatible e-commerce frameworks F1 and F2. At some point in time, it will become a part of the General Methodology CWA (ECIMF-GM).

The Guideline has been divided into several steps, to be performed sequentially and iteratively, as needed. The result of their successful completion will be a set of interoperability rules, which allows parties using different frameworks to cooperate towards common business goals.

This 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.

The integration steps according to this guideline can be represented graphically as shown on Figure 1. The layers on the top are addressed first, since they give the broadest context necessary for understanding of the lower-level data transformations.

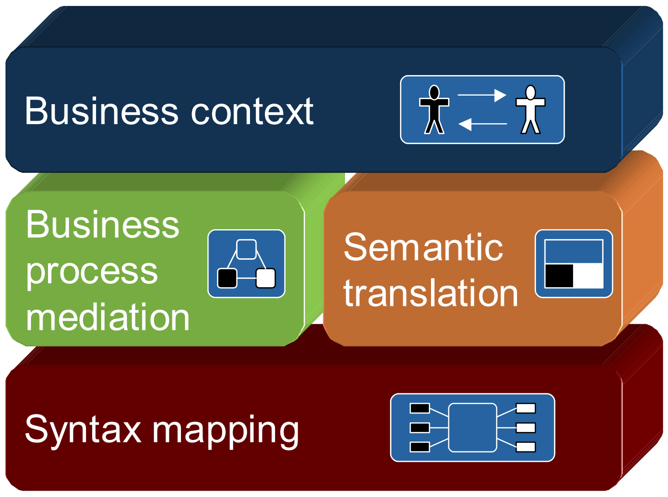


Figure 1 ECIMF layers of integration

These steps are listed below, and explained in detail in further sections:

- **Business Context Modeling**: 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, and its model is prepared.
  This stage corresponds to the Business Requirements View modeling phase in [UMM].
- **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.
  This final step corresponds to the Design phase in [UMM]

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.

1. Business Context Equivalence

1.1. Business Context Meta-model

The **business context** of an integration scenario is a set of *economic resources, events, agents, commitments* and *agreements* related to that scenario. See [REAont] for precise definitions of each of these terms, or the SimpleREA procedure described below.

1.2. Business Context Model

The **business context model** shows a concrete business scenario expressed with the help of these primitive concepts. We suggest using the standard UML diagrams for that purpose, e.g.:

- Use-case diagrams to show a high-level overview, with the detailed scenario descriptions.
- Class diagrams to show the specific types of entities involved.
- Collaboration diagrams to show a specific scenario populated with specific instances of participating entities.
- High-level activity diagrams of business interactions, with clearly marked transaction boundaries and rollback/compensation activities in case of failure (this is optional, because these diagrams will be created in the next step anyway).

Below you can find information on several ways to build such a model:

Business Context Modeling	
Input	Traditional business knowledge, legal agreements between partners, industry specific rules, legal constraints, specific business goals, common business practices and codes of conduct
Output	The Business Context Model for the integration scenario, defined in a set of UML diagrams (use-case, class, collaboration, activity)
Alternative Procedures	
REA	REA ontology [REA], [REAont]
UMM	Business Requirements View in Chapter 9.2 of [UMM] (can be considered a specialized and extended version of basic REA)
EbXML	Business Process Analysis Worksheets and Guidelines [bpWS] (which are also based on REA principles)
SimpleREA	Described below.

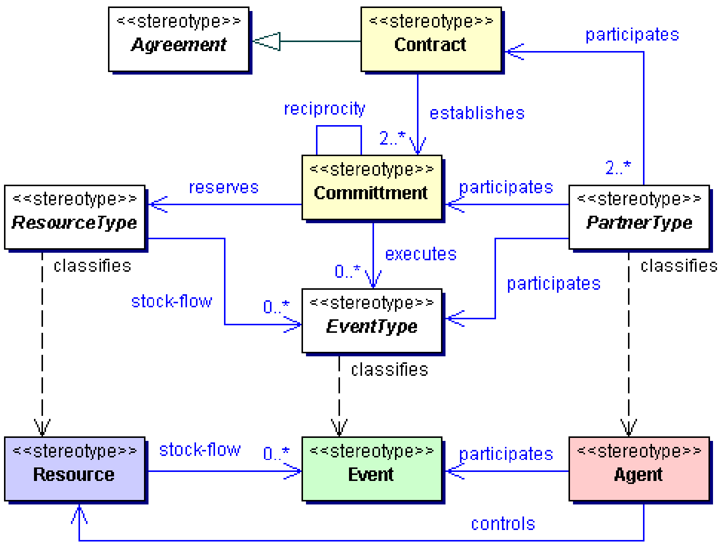
Simple REA
Here we describe a simplified procedure useful for modeling of simple business cases (based on REA, with relationships to UMM BRV and BTV; it should also be compatible with ebXML). As a result of the pragmatic process described below, you will create a business collaboration diagram, which provides a high-level overview of the entities involved in the business activities (this is called the exchanges diagram in REA).

1. **Business Collaboration Diagram (UML collaboration diagram)**
   1.1. Meta-model
   Describe the entities involved in the business case at hand, using the following terms (represented as UML stereotypes):
   - ***PartnerType***: the role that a business partner plays in the scenario (e.g. buyer, seller, payer etc…)
   - ***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.

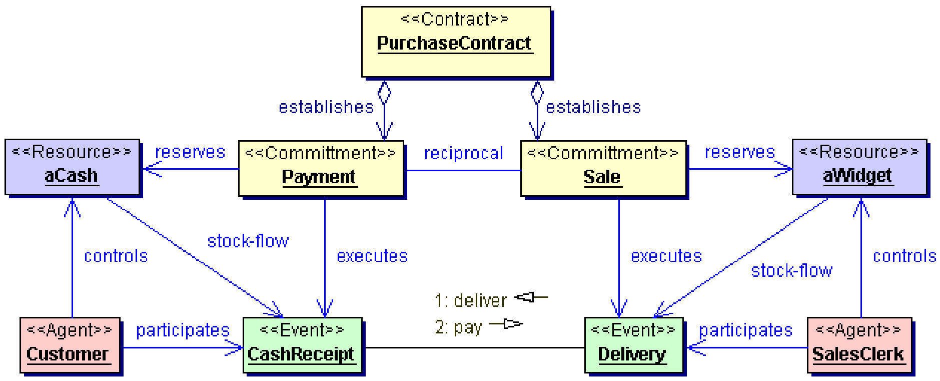
1.2. Meta-model diagram and constraints

The graphical representation of this meta-model is presented on the figure below:



The entities have been color-coded. White color has been reserved for abstract entities.

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).
(NOTE2: consider adding to this a second diagram, presenting the overall context for this particular contract in the resource flow of the whole company – somewhat similar to the top-level business process referred to as "Context overview" [see http://www.eidx.org/publications/business_models/ordmodl1_context.html for such example]).

1.3. Business Context Model Equivalence

As a result of executing the procedures described above, we will 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 are equivalent. There are several requirements that the models have to meet for them to be considered equivalent:

- Parties need to play complementary roles (e.g. buyer/seller)
- The resources expected in the exchanges need to be equivalent to the ones expected by the other partner (e.g. cash for goods)
- The timing constraints on events (commitment specification) need to be mutually satisfiable (e.g. down payment vs. final payment)
- The sequence of expected business transactions needs to be the same (even though the individual business actions may differ)
- *(More?)*

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. So, in other words, after a successful completion of this step we have established a common business context for both parties. We have also identified the events that need to occur, which in turn determine 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. Business Process Mediation

2.1. Business Process Meta-model

This diagram describes 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]) is a sequence of *business tasks* performed by one business partner alone, and *business interface tasks* performed by two or more business partners. In this guideline we will be interested primarily in aligning the **business interface tasks**, although in most cases understanding both types of tasks is needed in order to understand the business process constraints.

In this model, each 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**.

Here are more detailed descriptions of each:

- **BusinessProcess**: a sequence of BusinessActivities needed to perform in order to achieve a business goal.
- **BusinessTask**: a logically related group of BusinessActivities.
- **BusinessActivity**: a business communication (initiated by a requesting or responding business partner). BusinessActivities may lead to changes in state of one or both partners.
- **BusinessTransaction**: a set of business information and business signal exchanges between two business partners 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. A BusinessTransaction is realized by a series of BusinessDocument exchanges.
- **BusinessDocument**: a message sent between partners as a part of information exchange.

2.2. Business Process Model

Business processes are most often modeled using UML activity diagrams (or similar notation), where each diagram represents a BusinessTask. This view relates to the business context view in the following way:

- The Events correspond to BusinessActivities
- The execution of all Events according to the Commitments corresponds to the BusinessTasks

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 though they are not subject to the same Business Agreement (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 BusinessTask between two specific partners).

(NOTE: using this meta-model, the BusinessProcess view will be equivalent to the Process Context diagram mentioned above – NOTE2, line 134.Obviously, this needs more discussion…)

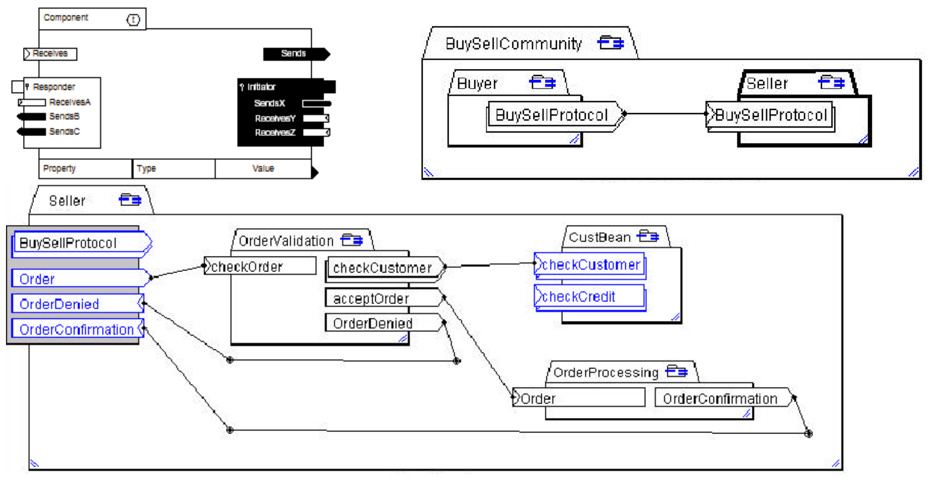
- **Identify the business processes** that support the Committments identified in the previous step.
- **Find corresponding BusinessTasks** that support the execution of the BusinessEvents identified in the previous step.
- **For each** business task in each framework:
    - Identify **request and response messages**. We suggest also building a more complete diagram containing two activity diagrams: one for requesting party, other for responding party. The diagram should also contain the messages passed between the parties. *(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 **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:
    - **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 occur 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 safe to change the order within the transaction boundaries, 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???)

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.
After successful completion of these steps, you can prepare the process mediation model.
(NOTE: the table below was prepared with the assumption that the UML-EDOC profile will be used for the notation elements).

Business Process Mediation Modeling	
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



(Some notation elements from EDOC profile – for consideration to use in the process models)

3. Semantic translation

- **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 Modeling	
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".

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: how to describe the exceptions? Well, for that matter, how to describe the rules? ☺)
  (NOTE 2: 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)

- **Message format** translation
  - **For each** data element E_i in M_i define the translation rules, based on the context of:
    - **Semantic differences**: identified in the Semantic Translation step
    - **Dynamic differences**: identified in the Process Mediation step
- **Message transport** translation
  - **Align packaging and transport** protocols, based on the specifications in each framework.
- *(to be continued…)*

5. References

[ECIMF-GM]: *E-Commerce Integration Meta-Framework, General Methodology*, CEN/ISSS/WS-EC project, 2001; available from: http://www.ecimf.org/doc/CWA/GM/ECIMF-GM.pdf

[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