Establishing Interoperability of Coordination Protocols in ad-hoc Inter-Organizational Collaborations

Bettina Bazijanec, Antonia Albani, Klaus Turowski

Business Informatics and Systems Engineering
University of Augsburg
Universitätsstraße 16, 86135 Augsburg, Germany
{bettina.bazijanec, antonia.albani, klaus.turowksi}@wiwi.uni-augsburg.de

Abstract: In the area of inter-organizational collaboration many standards and technologies have been developed in order to support interoperability between enterprises, but limited success has been achieved so far. Especially for *ad-hoc collaboration* there is no technical support available allowing the dynamic composition and connection of sub-systems for collaborating partners. Since companies are increasingly collaborating in dynamic changing value networks, the necessity for ad-hoc collaboration support becomes more and more important. In this paper we therefore propose a process for establishing interoperability of coordination protocols in ad-hoc inter-organizational collaboration. This process describes the necessary steps in automatically constructing an agreed overall coordination protocol addressing business-related as well as communication-related tasks.

1. Introduction

In inter-organizational collaboration business processes that span multiple partner organizations have to be efficiently supported while preserving autonomy of each partner. Due to the fact that no internal process details should be disclosed only messages containing relevant information can be exchanged between business partners for coordination purposes. It is crucial that messages reach business partners on time and that they contain exactly the information necessary to complete the next internal process step. Therefore, collaboration partners agree on coordination protocols that define valid messages and a chronological order for the information exchange. Every partner is then able to implement a binding of these protocols to internal business processes and applications. In ad-hoc collaborations the agreement on coordination protocols is much harder to achieve, because in such a scenario it is not guaranteed that every partner knows each other's capabilities. Hence, at the beginning of every ad-hoc collaboration there has to be a setup phase where partners agree upon basic facts regarding their joint business activities. This may also include providing trust between partners as this is a crucial factor in inter-organizational collaboration. The focus of this paper is set on establishing technical interoperability assuming that a necessary level of trust between business partners already exists.

In the aforementioned setup phase two main activities have to be performed: agreeing on *collaboration content* as well as on the *chronological order* of collaboration tasks. Agreement of collaboration content includes the definition of desired collaboration goals and necessary tasks that will lead to these goals e.g. tasks like *order* or *quote* are relevant to the goal *supply of material*. The assignment of roles (e.g. buyer or seller) based on identified tasks and the partition of tasks into sub-tasks are also included in this phase (see figure 1).

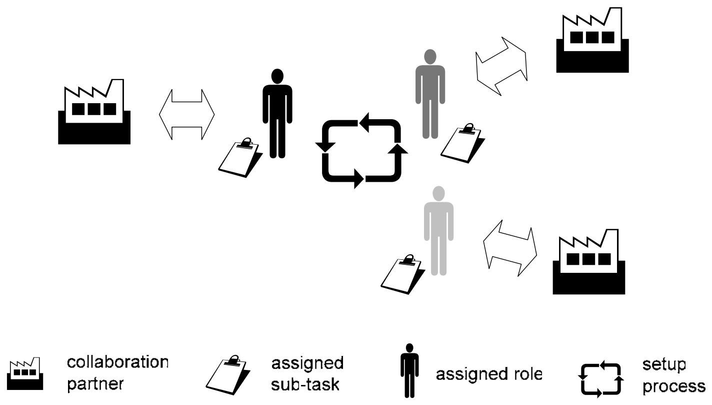


Fig. 1. Agreement on collaboration content

As sub-tasks have to be performed in a coordinated way to achieve the given goal their chronological order based on causal connections and technical capabilities of the participating partners has to be defined by the means of coordination protocols.

In this paper a setup process for ad-hoc collaborations will be introduced that describes these main setup activities in detail and shows how to obtain an interoperable, overall coordination protocol. At first, section 2 introduces tasks that are relevant to inter-organizational collaboration and how transition systems can be used to formally define coordination protocols. These concepts are then used in section 3 to describe and explain the process steps. In section 4 related work is discussed and in section 5 conclusions are drawn and an outlook on future work is given.

2. Collaboration tasks and coordination protocols

In the following sections two important concepts are introduced that will later allow to specify necessary steps in the setup phase of an inter-organizational collaboration: *tasks* and *coordination protocols*.

2.1. Types of collaboration tasks

In a single collaboration scenario several tasks that address different aspects of collaboration have to be coordinated at the same time. Hence, different types of collaboration tasks can be identified [3] that fall into one of the following categories:

- *Setup related tasks*: As already outlined above, these tasks have to be performed before an ad-hoc collaboration can start and include search for potential collaboration partners that are capable to perform certain tasks (e.g. in a central registry), negotiation of collaboration content and adjustment of remaining incompatibilities.
- *Business related tasks*: These tasks define the collaboration subject i.e. those steps that are necessary to achieve the desired goal. Since every new collaboration scenario may have other goals than previous ones partners have to agree on these tasks each time again.
- *Context related tasks*: Communication in inter-organizational collaboration is performed between two or more partners over open networks. Therefore, additional tasks have to be carried out in order to assure correct, secure, and consistent delivery and storage of business-related information. Hence, they constitute the execution context of business-related tasks.

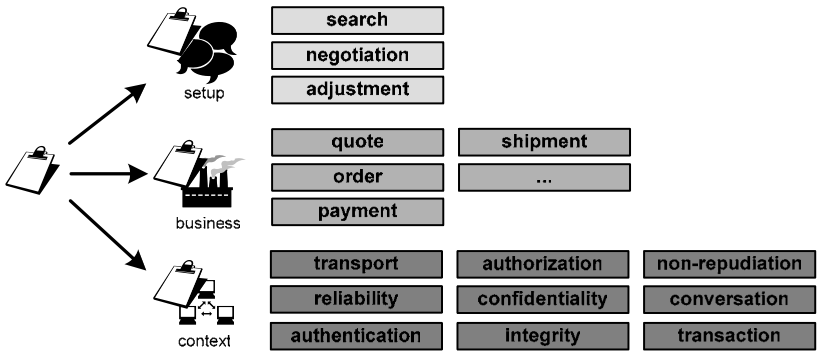


Fig. 2. Setup-, business- and context-related coordination tasks

Figure 2 shows the identified task categories and corresponding task types. Whereas the number of setup and context related tasks is more or less limited there are a lot of business related tasks that partners can choose from. However, during the setup phase a set of business-related tasks will be fixed for the scope of the collaboration scenario (see section 3). Every collaboration task generates a coordination effort between the partners that has to be mastered by the means of message exchange, e.g. the task order implies the exchange of a purchase order by the buyer role such that the seller role can process it and then acknowledge or deny. Hence, for each task a protocol can be defined. As setup-related tasks have to be performed in order to agree on business-related and context-related tasks it is assumed that the setup process as it will be introduced in this paper can be understood and performed by each potential partner.

2.2. Definition of coordination protocols

As already mentioned, coordination protocols define message exchanges that are necessary when performing inter-organizational collaboration tasks. A *protocol* can be formally defined following the definition of transition systems [10] as a 5-tuple p=(Q, Σ, δ, s0, F) where Q is a finite set of states and Σ is a finite set of valid labels that represent protocol messages which are exchanged during an interaction. Messages are marked *out* if they represent outgoing messages and *in* if they represent incoming messages. $\delta \subseteq Q \times \Sigma \times Q$ is transition relation, s0∈Q is the initial state, and F is a set of final states. The resulting transition system is specific for a particular role in an interaction. Each outgoing message in the protocol description corresponds to an incoming message of another role's protocol description. As an example, a simple purchase protocol from a seller's viewpoint can be defined as follows:

Q = {q0, q1, q2, q3, q4}, Σ = {RFQ, QUOTE, PO, POA}, F = {q4}, s0 = q0
δ = {(q0, in:RFQ, q1), (q1, out:QUOTE, q2), (q2, in:PO, q3), (q3, out:POA, q4)}

First, the seller receives a request for quote (RFQ) which leads to a protocol state change from q0 to q1. In this state he is able to send out a QUOTE-message that contains current conditions. After sending out the message he again changes his state where he is then able to receive a purchase order (PO) and finally to send out a purchase order acknowledgement (POA).

This kind of protocol definition only uses a set of message names that denote the types of corresponding messages. In order to also define these types a data structure is used describing the respective message, i.e. if a purchase order message and its corresponding purchase order type are defined, then every message with the same data structure is considered to be a purchase order [4]. Data structures have a specific information content and can be described on three semiotic layers (see figure 3): syntax, semantics, and pragmatics [11].

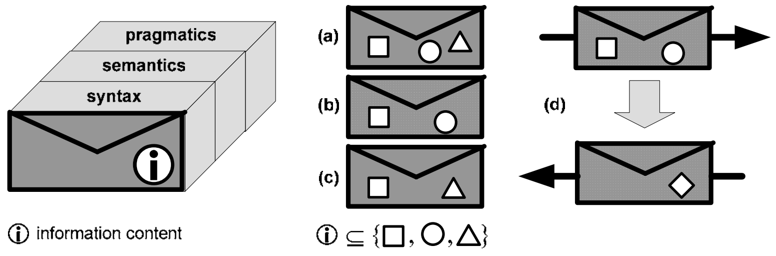


Fig. 3. Messages: syntax, semantics and pragmatics

The *syntax* describes which expressions are valid and which rules are used to build up the message's data structure from these expressions. A schema definition can provide this kind of description. Since the mentioned expressions and also data structures that are build up from expressions, refer to real world objects they have a specific meaning which is called *semantics*. Although two expressions are different they may have the same meaning e.g. `addr.` and `mailaddress` can both refer to a postal address. Another aspect of semantics is the information content that is captured in a data structure [14], because different partners may have defined separately a type with the same name that should denote the same real world object but nevertheless they are different because one type includes more information than the other, e.g. an address may be defined with or without providing country information. If, for example, an organization is only acting within the US then this information is not relevant. Therefore is has to be clear which information content is minimal to define the shared semantics of a message. Figure 3 gives an example: if square and circle define one real world object then (a) and (b) are representations of this object although (a) provides more information. (c) does not refer to the same object (even if it has the same name as described above). If only the square defines an object then all three messages would represent the same object. Ideally, two separately defined message types with the same name should have the same information content. Finally, the *pragmatics* of a message provides causal relations between a received message and the following action. With regard to protocol definitions a received message can enforce the sending of a message with a specific type, e.g. a request for quote requires to be followed by a quote (see also figure 3 on the right). It is also possible that a message requires some particular action to be taken by the receiving actor that are not obviously linked to the next following message, e.g. a message requires a specific processing or it has to be decoded by a specific algorithm so that the content can be read out.

3. Establishing coordination protocol interoperability

A collaboration scenario is a combination of various coordination steps supporting several tasks. These tasks have to be identified and agreed upon by the participating parties. Necessary coordination steps for each task are defined by the means of so called protocol definitions as described in the previous section. As every protocol introduces its own messages and sequences an overall collaboration protocol has to be composed from single coordination protocols in a way that it leads to the desired collaboration outcome. This process of setting up an ad-hoc relation between collaboration partners is shown in figure 4 and will be described in the following sections:

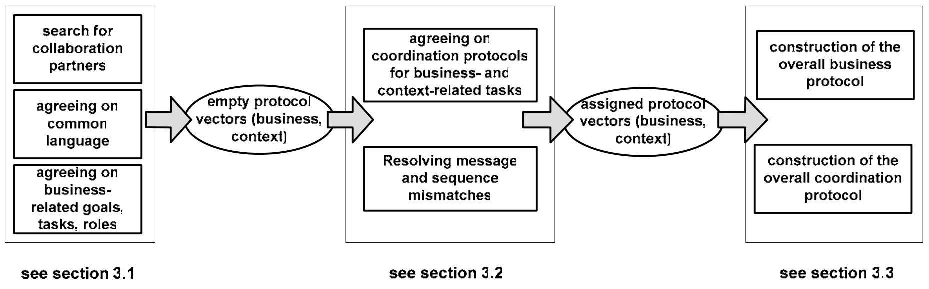

search for collaboration partners
agreeing on common language
agreeing on business-related goals, tasks, roles
empty protocol vectors (business, context)
agreeing on coordination protocols for business- and context-related tasks
Resolving message and sequence mismatches
assigned protocol vectors (business, context)
construction of the overall business protocol
construction of the overall coordination protocol
see section 3.1
see section 3.2
see section 3.3

Fig. 4. Collaboration setup process (overview)

3.1. Agreement on the collaboration content

In order to identify relevant tasks for an inter-organizational collaboration goals have to be defined and partners have to be found. In an ad-hoc scenario there is typically one organization that has some kind of demand and tries to satisfy this demand by requesting some value-adding activities from one or more other organizations. This can be the shipment of physical goods as well as the computation of driving directions on the Internet. Since potential partners are not known in advance some kind of directory is needed to search for them. This is a common approach also used in the field of Web Service discovery [1]. The directory provides contact information and a classification of enterprises based on industry sector or even based on a service classification. After having identified a set of potential partners (assuming the general willingness to collaborate) negotiation in form of direct interaction can begin in order to gain knowledge about different task types which need to be performed. In this negotiation, first a common language has to be determined. There could be a different understanding of task identifiers e.g. a task *purchase* could only include the exchange of order information or it could also include prior exchange of price information. A possible way to overcome such problems is to agree on a taxonomy that contains a hierarchy of business related tasks. Today, taxonomies are well-known in the field of material classification (e.g. eCl@ss [6]). After having established a common understanding of task types the agreement on tasks themselves can start. Figure 5 shows a possible outcome of this task agreement step.

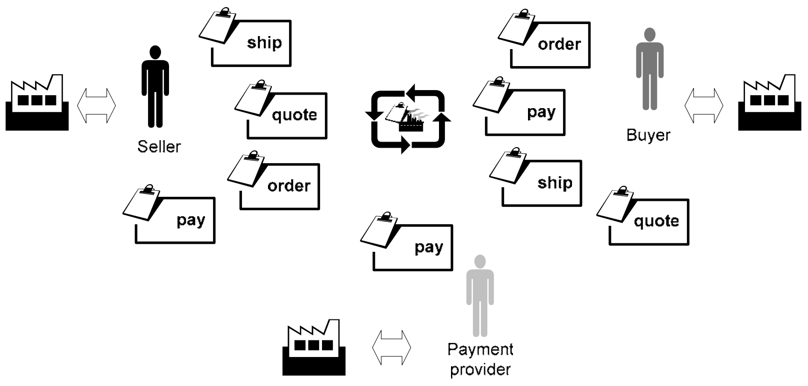


Fig. 5. Agreed business tasks and roles

Three roles have been identified and for each role tasks have been defined. At this time no sequencing of tasks is expected, only the capability to perform these tasks in the given role is relevant. Since each task execution relies on the usage of coordination protocols, as introduced in section 2.2, two so-called *protocol vectors* can be defined, namely one protocol vector V_B for business-related and one protocol vector V_C for context-related tasks. A protocol vector represents one possible protocol combination to coordinate given tasks and is defined as a tuple $V \in P_1 \times P_2 \times \cdots \times P_n$ where P_i is the set of protocols $p_i^1, \ldots, p_i^n$ that coordinate task *i*. In order to be able to agree on a common protocol vector it is first necessary to determine the set of possible coordination protocols for each of the positions in both of the vectors i.e. all partners have to provide information about their supported protocols.

3.2. Adjustment of coordination protocols

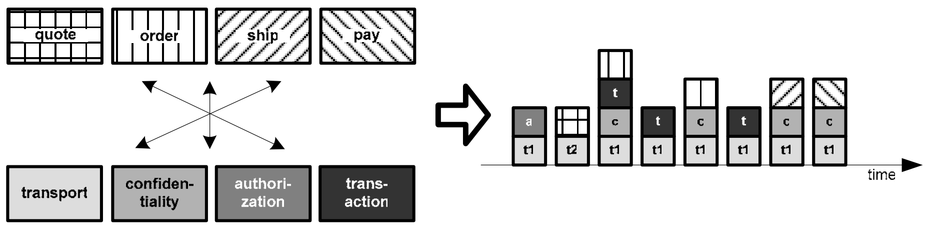
This process step begins with the publication of supported coordination protocol standards (for V_B as well as for V_C) by each business partner. These can be standardized protocols like for example described in a RosettaNet PIP or in an OpenTrans document exchange. If no standard can be referenced a protocol description has to be provided. These descriptions can be specified with the help of standards like BPEL or ebXML/BPSS which allow to define view-specific coordination protocols [1]. Since both references to standardized protocols and self defined protocols can be found in this negotiation step they have to be made comparable. As formal protocol descriptions like transitions systems (see section 2.2) or Petri Nets [15] allow reasoning about correctness or compatibility of definitions [17] a mapping from protocol standards to transition systems has to be provided. For example, compatibility of protocols can be tested by computing the intersection of two transition systems as described in [17]. If a protocol is defined according to a reference standard then additionally a formal description has to be provided for this protocol. If a description standard is used then a mapping of description language constructs to the formal notation is necessary. If protocols are defined in an aggregated way (e.g. a complete purchasing protocol including task like quote, order, and payment) these have to be split so that a sub-protocol for each task can be provided. The goal here is to be able to compare protocols and reason about their compatibility. This is necessary because there can be protocol mismatches that hinder partners to interoperate through given protocols [18], [5]. Those problems are:

- *Message mismatch*: Data structures representing message types may differ in their syntax or semantics. Ideally, two separately defined types with the same name should have the same schema and the same information content, but that is dependent on the actors that defined these types. In an inter-organizational scenario with many participants it is desirable to agree upon a data schema in order to avoid problems with differently defined types. Especially in ad-hoc collaborations this can not be achieved so these problems may occur.
- *Sequence mismatch*: Mismatches can also occur regarding message exchange orders. If patterns of interaction do not fit together [8] problems like deadlock or unspecified reception may arise [18]. Although partners support coordination of the same task they are not able to do so. Sequence problems may be caused by different message pragmatics (i.e. different reactions of partners are expected after receiving a message e.g. because of internal processing restrictions), or different semantics of same message types (i.e. types of the same name do not describe the same information content).

Some message and sequence mismatches can be resolved. This is called *mediation* and is based on the construction of a mediator [16], also named proxy or adapter, that interposes message exchange. This mediator is able to transform messages i.e. to map a source to a target schema which includes for example rearrangement and aggregation of expressions [13]. Mediators may therefore filter out unnecessary information or add information from external sources. Mediators may also resolve sequence mismatches by collecting or splitting information and generating new messages [18]. For those coordination protocol mismatches that remain unsolved *human intervention* is necessary to resolve these problems if possible. At best, all partners use the same reference standard for a certain coordination task e.g. RosettaNet PIP 3A4 for the order task. In this case a simple *adaption* of technology can achieve interoperability as long as all parties can agree on essential protocol parameters. If not, then for each task their protocol descriptions have to be analyzed with regard to protocol compatibility i.e. if potential message or sequence mismatches can be resolved.

3.3. Construction of an overall collaboration protocol

If an agreement on one coordination protocol is possible for every position in vector V_B then the overall sequencing of business related tasks has to be defined i.e. an overall business protocol on a higher aggregation level has to be constructed. Here again, potential problems due to sequencing mismatches may be encountered. For example, the buyer wants to receive the goods before payment but the seller only ships goods after payment. The techniques used to check compatibility of single coordination protocol can again be used in this process step. After having finished this first construction step an overall collaboration protocol has to be constructed where protocol information stored in vector V_C is added to the overall business protocol. As context-related tasks may be performed at each single step of a business protocol one context protocol may appear more than once in the overall coordination protocol (e.g. if multiple transactions have to be implemented). Also, context protocols may affect the construction of a business messages (e.g. if encryption is necessary). Figure 6 shows an example of a simple purchasing protocol generated from four business tasks (*quote*, *order*, *ship* and *pay*). This protocol has to be extended with context related tasks, namely *transport*, *confidentiality*, *authorization*, and *transaction*. Collaboration partners have to specify where each context related task has to be performed, so that an overall protocol can be constructed. In this example two transport protocols are used for different business protocol steps e.g. t1 represents HTTP and t2 represents SMTP. So, all message exchanges except the exchange of quote information, which is done via SMTP, will be performed using HTTP. For the other three context-related tasks there exist also agreed protocols. The transition definition part of each protocol is given in figure 6 (gray background color):



(q0, in:RFQ, q1)
(q1, out:QUOTE, q2)
(q2, in:PO, q3)
(q3, out:POA, q4)
(q4, out:SHIP, q5)
(q5, in:PAY, q6)

overall business protocol

(s0, in:T[<req>], s1)
(s1, out:ENROLL, s2)
(s2, in:ENROLLED, s3)
(s3, out:T[<resp>], s4)
(s4, in:PREPARE, s5)
(s5, out:PREPARED, s6)
(s6, in:CONFIRM, s7)
(s7, out:CONFIRMED, s8)
(s4, in:CANCEL, s9)
(s9, out:CANCELLED, s8)

transaction protocol

(r0, in:USR_PW, r1)
(r1, out:AUTH_OK, r2)
(r1, out:AUTH_DENY, r3)

authorization protocol

(t0, in:C(<req>), t1)
(t1, out:C(<resp>), t2)

confidentiality protocol

(u0, in: USR_PW, u1)
(u1, out:AUTH_OK, u2)
(u1, out:AUTH_DENY, u2)
(u2, in:RFQ, u3)
(u3, out:QUOTE, u4)
(u4, in:C(T[PO]), u5)
(u5, out:ENROLL, u6)
(u6, in:ENROLLED, u7)
(u7, out:C(T[POA]), u8)
(u8, in:PREPARE, u9)
(u9, out:PREPARED, u10)
(u10, in:CONFIRM, u11)
(u11, out:CONFIRMED, u12)
(u8, in:CANCEL, u13)
(u13, out:CANCELLED, u14)
(u12, out:C(SHIP), u15)
(u15, in:C(PAY), u16)

overall collaboration protocol

Fig. 6. Construction of overall collaboration protocols (example)

It is also shown how an overall protocol definition can be generated using single protocols (see right column in figure 6). Sometimes it is only necessary to concatenate a context related protocol like the authorization protocol in this example. Other protocols are inserted between business protocol steps like the part of the transaction protocol that coordinates the two-phase commit (PREPARE/CONFIRM-CANCEL). Then there are protocols that change business protocol messages. This can simply be additional processing information in a header (brief X[<…>]) or a message encoding like ciphering (brief Y(<...>)). In this case placeholders are used in the protocol descriptions that are specified by the partners.

4. Related Work

This paper introduces a process model for establishing interoperability in ad-hoc collaborations. The approach to publish capabilities and select intersecting capabilities for configuration of the partners' integration systems can also be found in ebXML. There, a so-called collaboration protocol agreement (CPA) is built up from the partners' collaboration protocol profiles (CPP) but the specification of single supported business tasks is very limited because it is only possible to reference whole business processes specified as ebXML artifacts [9] what simplifies the automatic negotiation of CPAs. However, a definite negotiation process is still not specified in the ebXML standard although the OASIS/ebXML CPPA Technical Committee is currently working on it. [2] describe requirements for the process of e-contracting including consistency rules. The proposed framework can be used to check correctness and consistency when constructing a negotiation process itself but gives no guidance how to specify or negotiate business protocols [2]. The problem of business protocol interoperability is addressed by several authors (cf. [17], [12]). The goal is to automatically decide if protocols are compatible which allows building architectures for ad-hoc collaboration. Approaches in the field of component interoperability, e.g. [18], [7], refer to similar problems so that their results, e.g. automatic generation of mediators form protocol descriptions, can be used with business protocols as well.

5. Summary and Outlook

This paper presents the definition of a setup process that has to be performed before an ad-hoc inter-organizational collaboration can start. Within this process agreement on collaboration content can be achieved as well as an overall coordination protocol can be constructed which defines a chronological order of tasks out of single protocols that describe the information exchanges that are necessary to coordinate given tasks. Our goal is to define a component-based architecture that implements this process and uses the outcomes to dynamically configure itself. Using this architecture it will be possible to technically support a wide range of ad-hoc collaboration scenarios as long as mismatches can be resolved. Due to the extensibility of component-based architectures it will be possible to easily deploy additional components implementing for example a new coordination standard for transactions which then can be used as one possible assignment to a protocol vector position. Future work therefore is twofold: the presented setup process has to be refined and implemented, and the component model as well as the composition model for the architecture has to be defined. For negotiation, compatibility testing, and mediator generation previous work in the fields of agent communication and component interoperability will be further examined and integrated.

References

1. Alonso, G., Casati, F., Harumi, K., Machiraju, V., *Web Services - Concepts, Architectures and Applications*, Springer-Verlag, Berlin, 2004.
2. Angelov, S., Grefen, P., *An Approach to the Construction of Flexible B2B E-Contracting Processes*, University of Twente, 2002.
3. Bazijanec, B., Winnewisser, C., Albani, A., Turowski, K., *A Component-based Architecture for Protocol Vector Conversion in Inter-organizational Systems*, International Workshop on Modeling Inter-Organizational Systems, Larnaca, Cyprus, Springer, LNCS 3292, 2004, pp. 556-567.
4. Bussler, C., *B2B Integration*, Springer, Berlin, 2003.
5. Bussler, C., *Public Process Inheritance for Business-to-Business Integration*, Technologies for E-Services, Third International Workshop, TES 2002, pp. 19-28.
6. *eCl@ss - Standardized Material and Service Classification*, http://www.eclass-online.com.
7. Farias, A., Südholt, M., *On components with explicit protocols satisfying a notion of correctness by construction*, DOA/CoopIS/ODBASE 2002 Confederated International Conferences 2002, pp. 995-1012.
8. Garlan, D., Allen, R., Ockerbloom, J., *Architecture Mismatch: Why Reuse is so Hard*, IEEE Software, 12 (1995), pp. 17-26.
9. Hofreiter, B., Huemer, C., Klas, W., *ebXML: Status, Research Issues, and Obstacles*, 12th Int.l Wrkshp on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE.02) 2002, pp. 7-16.
10. Hopcroft, J. E., Motwani, R., Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, 2001.
11. Liu, K., *Semiotics in Information Systems Engineering*, Cambridge University Press, Cambridge, UK, 2000.
12. Mallya, A. U., Singh, M. P., *A Semantic Approach for Designing Commitment Protocols*, Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04) 2004.
13. Rahm, E., Bernstein, P. A., *On Matching Schemas Automatically*, VLDB Journal, 10 (2001), pp. 334-350.
14. Sheth, A., *Data Semantics: What, Where, and How?",*, in R. Meersman and L. Mark: *Data Semantics (IFIP Transactions)*, Chapman and Hall, London, 1996, pp. 601-610.
15. van der Aalst, W. M. P., Weske, M., *The P2P Approach to Interorganizational Workflows*, Proceedings of the 13th International Conference on Advanced Information Systems Engineering 2001, pp. 140-156.
16. Wiederhold, G., *Mediation in information systems*, ACM Computing Surveys, 27 (1995), pp. 265–267.
17. Wombacher, A., Fankhauser, P., Mahleko, B., Neuhold, E., *Matchmaking for Business Processes Based on Choreographies*, IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04) 2004, pp. 359-368.
18. Yellin, D. M., Strom, R. E., *Protocol Specifications and Component Adaptors*, ACM Transactions on Programming Languages and Systems, 19 (1997), pp. 292-333.