Towards Flexible Inter-enterprise Collaboration: A Supply Chain Perspective

Boris Shishkov[1], Marten van Sinderen[2], and Alexander Verbraeck[3]

[1] TU Delft, Department of Systems Engineering/IICREST
Delft, The Netherlands
b.b.shishkov@tudelft.nl, b.b.shishkov@iicrest.eu
[2] University of Twente, Department of Computer Science
Enschede, The Netherlands
m.j.vansinderen@ewi.utwente.nl
[3] TU Delft, Department of Systems Engineering
Delft, The Netherlands
a.verbraeck@tudelft.nl

Abstract. Since neither uniformity nor pluriformity provide the answer to easing inter-enterprise collaborations, we address (inspired by relevant strengths of service-oriented architectures) the problem of supporting such collaborations from an infrastructure perspective. We propose architectural guidelines for interactively establishing a suitable inter-enterprise collaboration scheme, before the exchange of actual content takes place. The proposed guidelines stem from an analysis of some currently popular approaches concerning the achievement of inter-enterprise collaborations with ICT means. Taking into account the strong relevance of these issues to the Supply chain domain, we put our work in the Supply chain perspective. We also illustrate our architectural guidelines with an example from this domain. It is expected that the research contribution, reported in this paper, will be useful as an additional result concerning the (ICT-driven) inter-enterprise collaboration.

Keywords: Inter-enterprise collaboration, Service-oriented architectures, (non-) Standardized collaboration, Broker-mediated collaboration, Supply chain, Knowledge-based traceability.

1 Introduction

An *inter-enterprise collaboration* requires collaboration mechanisms that concern both *organizational aspects*, e.g. to agree on a joint process, and *technological aspects*, e.g. to enable the information exchange [7]. Depending on its role in such collaborations, an enterprise may need to exchange information with up to hundreds of other enterprises, as is often in a *supply chain* [3,28,8], for example. A Supply Chain (*SC*) collaboration can be considered as consisting of a number of bilateral collaborations where the enterprises involved in more than one bilateral collaboration are responsible for the (self-assumed or agreed-upon) coordination of these collaborations. Each bilateral collaboration is usually driven by a *collaboration contract* that describes the

J. Filipe and J. Cordeiro (Eds.): ICEIS 2009, LNBIP 24, pp. 513–527, 2009.

'normal' scenario for service delivery between two enterprises as well as a number of foreseen 'exception' (deviation) scenarios.

An enterprise may encounter difficulties however, if it would need to replace a peer enterprise by a new one in an existing collaboration and/or start up a new collaboration. Such changes are not easy to achieve since new contracts and underlying mechanisms have to be established; moreover, accidentally introduced errors may proliferate via the enterprise to other collaborations. Inter-enterprise collaboration is facing therefore *inflexibility*, unacceptable *management overhead*, *error propagation*, high *latency*, and *inefficient use of resources* [4].

Standardizing the mechanisms for bilateral collaborations is not very helpful since the resulting standards must be all-encompassing (i.e., address all aspects of the collaboration that fulfil the requirements of as many enterprises as possible). Such standards would be therefore complex and voluminous, and also hard to agree upon and/or change [4,5,23]. Introducing broker systems that handle multiple specialized collaborations and provide bridges between them is not very helpful either. The reason for this is that full translation between collaboration mechanisms is expensive or sometimes even impossible, in which case human intervention is needed [21]. Moreover, broker approaches are inflexible since a change in one collaboration mechanism impacts all translations between this mechanism and the other mechanisms that the broker supports. Finally, it is often that enterprises do not trust a broker.

Considering *SOA* - Service Oriented Architecture [22] and adopting a SOA approach, in which each enterprise presents its collaboration capabilities as self-contained and loosely-coupled services, may be considered attractive for the following reasons:

- services are technology-agnostic, i.e. they may be implemented by an enterprise in any way without visibility for other enterprises that act as service users;
- services are self-describing and discoverable, i.e. they have descriptions stored in a registry that can be queried by service users;
- services are composable, i.e. orchestrations of services can be specified and executed resulting in 'higher-level' services for service users.

This certainly requires a distributed computing infrastructure, which is supported nowadays to some extent by Web services standards [17].

SOA has therefore some clear potential benefits, with an open question nevertheless: whether relatively simple services can be defined and described, such that suitable orchestrations of such services can realize long-running inter-enterprise collaborations with many enterprises involved. Another question is who would actually do the orchestration and provide the overall functionality.

For this reason, we expect that a SOA approach can be beneficial, however only if it is the case that the specific problems of inter-enterprise collaboration are adequately addressed, especially with regard to the SC domain.

Inspired by some recent related achievements [27], we propose in this paper *architectural guidelines* that concern inter-enterprise collaboration. In particular, it is envisioned interactively establishing a *suitable inter-enterprise collaboration scheme* before the exchange of actual content takes place. The scheme may specify: (i) collaboration protocols; (ii) content structures; (iii) orchestration processes. The successful negotiation of a collaboration scheme determines which partners can be

selected for specific business collaboration. Our proposed guidelines stem from an analysis of some currently popular approaches concerning the achievement of inter-enterprise collaborations with ICT means. Taking into account the strong relevance of these issues to the SC domain, we put our work in the SC perspective. We also illustrate our architectural guidelines with an example from this domain. It is expected that the research contribution, reported in this paper, will be useful as an additional result concerning the (ICT-driven) inter-enterprise collaboration.

This paper is further organized as follows. Section 2 contains an analysis of some currently popular inter-enterprise collaboration mechanisms. Section 3 considers (on this basis) related implications from the SC perspective. Section 4 presents our proposed architectural guidelines. Section 5 discusses an example of how they could be realized. Section 6 outlines some related work. Finally, Section 7 presents the conclusions.

2 Collaboration Approaches

Inter-enterprise collaborations are essential nowadays for enterprises in their aiming at delivering competitive services [18,29]. We can distinguish at least two collaboration perspectives, namely the 'informa perspective' (concerning information exchange and respectively the ability to formulate and interpret messages) and the 'performa perspective' (concerning the essential human ability of doing business, by engaging into commitments, either as performer or as addressee of a coordination act) [26]. In the following we focus mainly on the *informa perspective*: the exchange of messages with an agreed meaning that concern the achievement of some business effect. In addition, we explicitly consider *long-running collaboration propositions*, since usually inter-enterprise collaborations are evolving over a longer period of time. Such collaborations involve negotiations, commitments, contracts, shipping and logistics, tracking, varied payment instruments, deviation handling and customer satisfaction [21]. Furthermore, we assume that collaborations: (i) represent a function that is critical to the business and therefore should concern a shared business meaning; (ii) are usually evolving on top of a standards-based formal trading partner agreement, such as RosettaNet, Partner Interface Processes (PIPs) or ebXML Collaboration Protocol Agreements (CPAs); (iii) are driven by strict collaboration syntax and rules; (iv) define communications protocol bindings [6].

Inter-enterprise collaborations in general require distributed solutions of heterogeneous ICT systems.

Some of the currently popular inter-enterprise collaboration mechanisms are standardized (for example through the Electronic Data Interchange – EDI [21]) with limited application while others are rigid, hard-to-develop, non-standardized; neither of these mechanisms however fully responds to the increasing demands for flexible and adaptive collaboration, and according to some [19,18], the solution should be in the direction of service-oriented rule-based approaches. Others [7] claim that supporting such collaborations by means of brokers would respond better to these demands. Inspired by these and other studies, we propose the following classification of possible inter-enterprise collaboration mechanisms: (i) non-standardized bilateral collaboration; (ii) standardized bilateral collaboration; (iii) broker-mediated collaboration. Each of these collaboration types is elaborated below:

- *Non-standardized bilateral collaboration* consists of a set of pair-wise 'closed' collaborations, using rules which are private between each pair of enterprises and not approved by some standardization body, as illustrated in Figure 1*a*. Hence, if Enterprise A wants to collaborate with Enterprise B and Enterprise C, then collaboration schemes have to be agreed upon between each two parties separately. Problems thus are that each enterprise has to 'talk' many 'languages' and it is difficult to introduce a new enterprise in the collaboration because all others would have to 'learn' a new language, assuming that the new enterprise can not be forced to 'speak' languages already in use by the other enterprises.

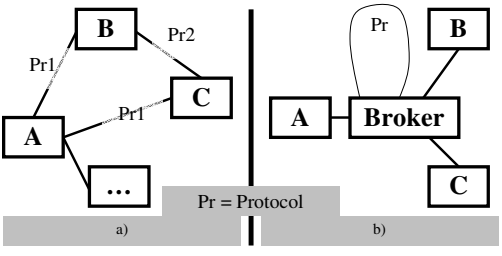

Fig. 1. Non-standardized collaboration (a), Broker-mediated collaboration (b)

- *Standardized bilateral collaboration* consists of a set of pair-wise collaborations driven by standards. Although standardization of inter-enterprise collaborations has often been proposed as a possible solution, standardization does not address all the observed difficulties, because of its decreasing flexibility. Next to that, it is hard for a single standard to deal with all types of deviations from the normal scenario, especially if the standard is to be used by thousands of enterprises with slightly different requirements. The early efforts in EDI have clearly shown this problem. Standards such as UN/EDIFACT [31] and ANSI ASC X12 [2] have become increasingly complex, while still not being able to address all potential problems in the collaboration process [19]. Due to its complexity, it is difficult to reach a global agreement on a standard, and on proposed changes and extensions. Standardization of inter-enterprise collaboration is a slow process [5], and changes are difficult to enforce and unpopular after the initial version, because there is already an installed base of users with dedicated software. In addition, such all-encompassing standards are hard to learn, not easy to adopt, and the software support is expensive.

- *Broker-mediated collaboration* makes use of brokers capable of 'understanding' multiple languages. There could be a central broker responsible for mediating all the 'conversions', as depicted in Figure 1*b*. Our assumption is that such a broker not only translates the syntax and semantic but also takes care of aligning the protocols, so that there are no process mismatches. Although protocol translations and syntax translations are possible, semantic problems often remain, and can lead to misinterpretation, source of failure in the collaboration, and a need for human interventions [21,32]. Being an intermediate party in the collaboration, the broker would often be insufficiently involved in dealing with deviations and errors, unless rigid rules are applied. The latter however would defy the flexibility and efficiency for which the broker was introduced. Another

concern is trust. Enterprises often do not trust a broker with important and commercial business content, thus reducing the applicability of brokers [19].

Hence, we argue that non-standardized bilateral collaborations are most secure, taking into account that it is not trivial for another entity to 'jump in'. The disadvantage here is the inflexibility with regard to new enterprises that might appear to be desired collaborators. This seems to be solvable by the enforcement of standards, which nevertheless points to many disadvantages at the same time, including the usual disagreement on who should introduce the standards and the insufficient security level. Brokers could be the desired mediators among enterprises, with a remaining question however whether everybody would trust a broker.

Applying SOA in each of the mentioned cases would result in collaborations running on top of a service infrastructure that takes care of basic interaction needs. For example: SOAP can be used for information exchange over HTTP and accessing a Web service; UDDI can be used for publishing and finding Web services. Ontologies that formalize relevant concepts, might also be used in each of the mentioned cases [1]. A broker, for instance, can be supported by a central ontology introduced. This would actually result in a single ('neutral') language, as illustrated in Figure 2 (as the figure shows, translations between languages A, B, and C come through the neutral language, x). Then, for each enterprise one would need only a translation between the enterprise language and the neutral language. Since it is impossible defining one ontology for the whole world, it would be necessary defining an 'upper ontology' (that is only reflecting the basic concepts and their relationships) and a specific ('domain') ontology, with a 'middle ontology' in between [16].

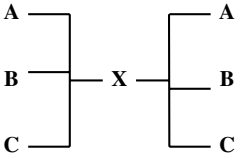

Fig. 2. Introducing a Central Ontology

And in the end, we would summarize the cross-cutting concerns that are to be taken into account when considering new architectural approaches for inter-enterprise collaboration: (i) flexibility in terms of collaborators and time; (ii) deviation handling; (iii) security and trust; (iv) cost; (v) change handling (how an enterprise would update its 'behavior' if there is a change in the environment).

3 Implications from the Supply Chain Perspective

In a SC, each organization typically needs to exchange information with several (possibly even thousands) other organizations (for example, if an organization has both a purchase and a sales function). Although SC standardization (ERP, CRM, workflow) has been proposed as a solution to the many-to-many problem, its effect is partial, just like in the general case and wrapping existing systems for talking to both enterprises seems only appropriate. The many-to-many problem has been hampering the successful

introduction of EDI for e-business, mainly because of increased implementation costs. Clearly, the readiness to be able to exchange information according to a number of protocols and workflow interaction patterns has a high price.

Information brokers have hence been introduced, 'acting between organizations', e.g. supporting the spot buy purchases of certain goods through trade exchanges [5]. Such brokers just combine information on supply and demand of goods without ever owning the goods themselves. The organizations still need to be able to exchange business messages with each other, in addition to 'finding' each other through the broker. Of course, brokers with more extensive functionalities exist as well: they take care of a larger part of the workflow. The network between organizations remains large however and a broker does not replace the majority of the connections. Organizations still want to be able to do business with each other without the broker (and avoid the additional payment for the broker's services).

A solution, widely considered, is replacing legacy applications by services, in the light of some claims that 'heavy-weight' monolithic systems could be usefully replaced by loosely-coupled (orchestrated) services providing adequate inter-organizational interactions [22,27]. These claims have not yet been properly proven in practice however. An important practical concern is the scalability issue: would it really be easy to scale up to realistic SCs [6]? It would only be easy, we argue, for organizations to exchange SC business messages if the syntactic, semantic, and pragmatic definition [16,26] of the required and offered services match, as it is depicted in Figure 3*a*.

With respect to this, we have several observations: (i) It is often hard for industry to agree on one representation standard and that's why a number of standards have been enforced over the last years by different software vendors trying to sell out their own solutions that are usually incompatible with other vendors' solutions [23]. (ii) Assuming that two organizations have different business logic, it would be challenging to channel all information through interfaces with the same definitions. Imagine, for example, that one of those organizations is introducing an additional service, enforcing thus the interface standards to accommodate the existence and functionality of that service for all organizations; (iii) With respect to service composition, a relevant question is who would actually 'orchestrate' the constituent services and provide the overall functionality.

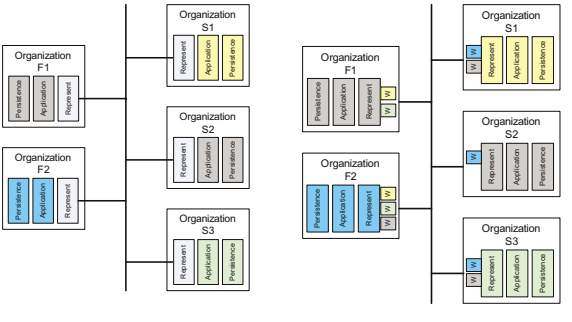

Fig. 3. Service-Oriented solutions - a) Idealistic services; b) Wrapped services

Indicative concerning the challenges mentioned above, is the incompatibility between industry solutions, such as Rosettanet [24], ebXML [9], and OpenXchange [20], not only at the representation level but also with respect to the business logic that constitutes the workflow. Hence, we would consider the popular claim that a possible solution to the pluriformity in the services domain is to introduce wrappers, keeping however the 'SOA context'. Figure 3*b* is presenting this solution in which wrappers would usefully enable services to 'talk' to many other implementations of the services in a service composition. This would mean, nevertheless, that a number of organizations should implement the same type of wrappers, which is costly and can easily lead to errors. The difficulty in making wrappers lies mainly in semantic and definitional differences, not so much in pure syntactic differences – these seem easily solvable, taking into account that current Web-service technology is standardized around XML and SOAP messages, with clear definition of the message formats in XML-schema or DTD files [34].

And in the end of this section, we will partially re-visit the cross-cutting concerns outlined at the end of the previous section; we would pay attention only to the SC-relevant ones: (i) Flexibility. In a spot-buy market, it is not known on beforehand with whom an enterprise will collaborate next. Therefore, the collaboration process has to be flexible, and should be able to work in many different configurations and under many different circumstances. (ii) Resilience to change. External requirements for the collaboration process are subject to continuous change. Policies and legal requirements change, and lead to needed adaptations of the process and of the information exchanged. These changes should not lead to drafting of a new version of the standard. (iii) Superfluous parts. A one-size-fits-all (all-encompassing) approach confronts the processes and users with a majority of fields and process steps that are not needed for the particular process, but rather for exotic versions of the process. This leads to overhead, errors, and implementation difficulties. (iv) Self-explanatory. When buying an IKEA cupboard [15], we don't expect everyone to know exactly how to put it together. Instructions are included and can be read on beforehand. In our information systems, even service-oriented ones, the instructions are not included and each system in the chain should know exactly how to handle each eventuality that can happen. Why not use the IKEA analogy for supporting inter-enterprise collaborations? This is a basis for the derivation of the main requirements (presented in the paragraph below) concerning the solutions to be proposed in Section 4.

REQUIREMENTS: (i) service orientation (as we have already concluded, a service-oriented solution would provide flexibility, re-use, and openness); (ii) protocol alignment (it is crucial that not only the syntax and semantics of the exchanged messages match, but also the message exchange protocols - this is pointing to needs for change impact analysis and related reasoning); (iii) no need for definitions of new standards (enforcement of yet another standard will not be easy); (iv) no broker-driven solutions (brokers are often not trusted as third parties).

4 Solution Directions

Concluding that neither uniformity nor pluriformity provide the answer to easing inter-enterprise collaborations, we have to look for another solution. When humans

exchange information in a complex setting, they discuss firstly the process, terms and conditions, after which the process is executed according to what has been agreed upon. This mechanism has been 'reflected' in workflow and orchestration languages describing a sequence of process steps that have to be executed between parties in order to implement an inter-enterprise collaboration; such languages have been analyzed by Honig [14]. Often, however, these workflows are defined once, and are not adapted to the properties of the particular process. There are also languages that describe a contract between parties, which specifies the conditions under which the exchange takes place, but they are usually quite static in nature [13]. A flexible variant that combines the dynamics of workflow and orchestration with the rigidness of the contracts (pre-conditions, post-conditions, time-outs, exception handling, and so on) would be attractive. First attempts to define languages describing the content of contracts that contain both content and workflow have been proposed already, e.g. the LinC language [14]. Still, these languages are not yet self-explanatory and do not contain their own 'manual' how to implement them.

In a sense, the exchange is a matter of matching the processes and information sent and received seen from one party with the processes and information received and sent by the other party.

In Figure 4, we see that Enterprise E1 expects to engage in a workflow of messages with another enterprise of the following sequence: (send A, receive B, send C, receive D, send E). Therefore, it looks for a partner that can interact in the following way: (receive A, send B, receive C, send D, receive E). When the two enterprises can agree on this workflow before starting to send and receive actual messages with content, i.e. agreeing on the protocol of interaction, they can be sure that the workflow can be continued from start to finish, and that the two enterprises will be able to make the deal they intended. To make this work, it is necessary that all issues mentioned above are adequately negotiated.

One of our proposals therefore would be considering an inter-enterprise modeling language that contains: (i) description of the normal workflow between parties; (ii) description of exceptional workflows between parties where needed; (iii) conditions on the workflow (e.g. timing, consistency, stop criteria); (iv) description of the content for each interaction step (using an existing ontology); (v) conditions on the content before and after each interaction step; (vi) instructions on how to handle the content in each instruction step (either computer readable or computer executable in case of a normal process, or human readable in case of error handling or escalation). Of course, such an inter-enterprise collaboration modeling language (to be addressed in more detail in further research) should again be described and formalized using another language. Furthermore, because we reason about content, an ontology is needed that describes the (part of the) world we are interested in. As we do not need one ontology, but we can use multiple ontologies and choose the one that is most appropriate for the problem at hand, it is not necessary to standardize this, which again helps with the flexibility demand. In realizing this we propose using the steps outlined in the following paragraphs.

First Step. Parties need to find each other. This is not much different from the discovery function in service-oriented architectures [17]. In many cases, parties will already know each other, and discovery is not a necessary function.

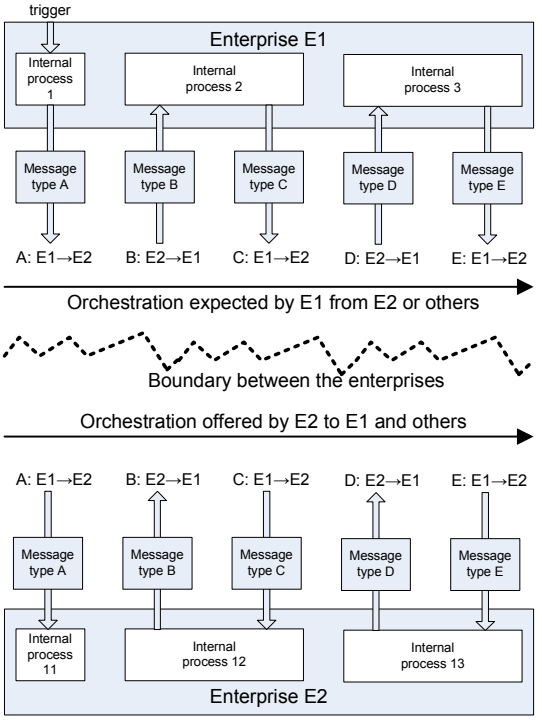

Fig. 4. A simplified example for an exchange between enterprise A and enterprise B that matches

Second Step. Matching the workflow (including deviations) and content (including ontology) between the enterprises. When enterprise A has defined a number of potential partners to work with, B_1, B_2, ..., B_n (in a simple bilateral agreement), a negotiation process is initiated that should identify whether each pair of enterprises can agree on joint content based on a joint ontology, and on a joint process. In a simple implementation, this matching process could be a simple one: enterprise A has a number of potential processes it could use (maybe a subset of an openly available set of potential inter-enterprise workflows), and each enterprise B_i has a set of processes as well. A match between the processes can establish whether they share a common workflow. If not, the negotiation with potential partner B_i can stop, and A can continue to negotiate with potential partner B_{i+1}. In a more complex solution, an algorithm can judge whether the differences are part of the critical part of the process or not, and can decide to use a partly overlapping process, or leave the decision to a human. In the end, this leads to the identification of zero or more partners to work with. If zero, the discovery phase can be redone or widened, or the failure can be escalated to a human who can decide on what to do. If more than one partner is found,

the best match can be chosen, or, in case of a spot buy market, the process can be continued with multiple partners.

Third Step. The process can be executed, following the agreed workflow; in some cases, executable components might be available to help with the execution of the process itself. These could be small services that are part of the workflow, and that can handle certain tasks. The fact that the executable components are part of the package, also could give the other partner(s) in the inter-enterprise collaboration some insight into how certain steps are executed. This could increase trust, and lead to a better acceptance of the inter-enterprise collaboration process. Furthermore, these components would reduce costs for the enterprises that implement the platform, especially if they are based on accepted standards, e.g. XML, SOAP, etc. [22].

In many cases, more than two enterprises would be involved in the information exchange. The orchestration can be done in exactly the same way, with increased complexity however, and more points of potential failure. Thus, we suggest considering the workflow as a transaction that can be rolled back completely, in case it does not succeed.

At a later stage, brokers could be added to this picture again, taking care of tasks that parties want to outsource (this is another role than that of the brokers discussed earlier) as part of their workflow process (payment through credit cards, certification of creditworthiness of partners, dealing with customs, and so on).

In responding to the trust concern however, a possible solution, especially relevant to the brokerage approach, would be in the direction of brokerage software that might be downloaded by each enterprise and translate within their boundaries the messages making them easily exchangeable with other enterprises using the same software.

Knowledge-Base Support. Matching the protocols, content structures, and orchestration of multiple partners, to find matches or near-matches, is considered to be a challenge. If we know for example that (i) Enterprise E_1 has dealt with Enterprise E_2 on orchestration O_3 for content types C_1, C_2 and C_3; (ii) Enterprise E_2 has dealt (at some point in time) with Enterprise E_3 using the same orchestration and content types; (iii) E_3 does not object to share this information with the partners of its partners (analogue to LinkedIn, FOAF, Amazon), then, if E_1 asks E_2 about suitable partners, E_2 would inform E1 about the existence of E_3. Such matching nevertheless requires complex *reasoning* related to tracing all relevant information.

We thus need to perform *knowledge-based traceability*, tracing back the previous collaborations of each of the enterprises; inspired from previous experience [27], we propose a traceability framework, whose technology-independent view is depicted in Figure 5.

As the figure suggests, the infrastructural support to inter-enterprise collaborations should include keeping track of previous collaborations. The relevant information should be stored accordingly in a knowledge-base that could be queried whenever an enterprise is about to launch a new collaboration, in order to have a basis for reasoning in support of the discovery of suitable partners. The reasoning results should not only be presented to the enterprise but they should also deliver adaptation instructions that concern the list of most appropriate enterprises that is to be made as a final result for the support of the templates choice.

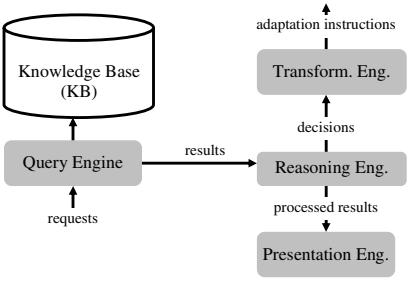

Fig. 5. Knowledge-based traceability – a technology-independent view

We will only partially illustrate some of these proposed solution directions in the following section, not going for a more thorough illustration for the sake of brevity.

5 Example

When two enterprises engage in an inter-enterprise collaboration, the overall pattern of collaboration involves a number of exchanges, e.g. when one enterprise buys a product at another enterprise, the minimal exchange taking place is depicted in Fig. 6:

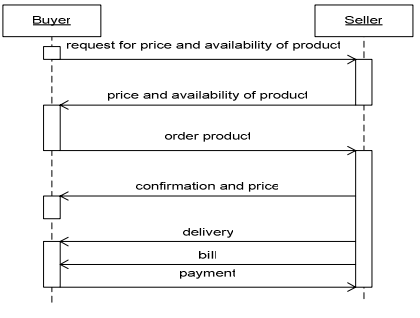

Fig. 6. Simple sequence of activities for buying a product

Of course, this process gets more complex when we have more than two parties involved and when we have a longer sequence of processes, or sub-processes. Furthermore, there are many points where this process can stop or time-out. The process might stop if the product is not available, or too expensive. A time-out occurs for instance when the intended seller does not answer the buyer in time. In that case, the buyer might look at another seller to see if this enterprise is more prompt, and whether the item is available there. If after that point in time the first seller would react, the process should prescribe what to do; discard the offer or take it into account anyhow. The seller also has a time-out on the offer; after some time the availability and price information does not hold anymore.

Let's call the buyer E_B and the seller E_S. The messages they try to exchange are the request M_R, availability and price M_A, order M_O, confirmation M_C, delivery M_D, bill

M_B, and payment M_P. There are also a number of protocols available to exchange messages, $P = \{p_i\}$, which is defined centrally and can be extended at any point in time. E_B can work with a subset of these protocols P_B, and E_S with a subset P_S. Furthermore, each of the message types $M = \{m_i\}$ has a definition, e.g. in an XML schema or DTD. In a simplified form, we could say that each message type consists of a sequence of name-class tupels $t_j = (n_j, c_j)$ where the name and class (type) are defined in an ontology. Thus, $m_i = \{t_{i,1}, t_{i,2}, ..., t_{i,n}\}$. An orchestration O is defined by a sequence $\{(M_1, d_1), (M_2, d_2) ..., (M_n, d_n)\}$, where $d_i \in \{in, out\}$ indicates the direction.

To find out whether we have a match, there are a number of steps. First, E_B sends P_B to E_S. E_S calculates $P_B \cap P_S$ and sends back the result $P_{B \cap S}$. If $P_{B \cap S} = \varnothing$, a match cannot be made and E_B will have to look for another supplier. If a protocol can be chosen, the next step is to see if E_B and E_S can work with the same message types. E_B sends each of the message types that it has in its sequence $m_{B,i}$ to E_S. E_S will match this with its internal list of available message types, and makes comparisons for each name-class tupel $t_j = (n_j, c_j)$ to see if it matches. A match in this case could also be formed by a subset of the range for the values. For each $m_{B,i}$ for which a matching $m_{S,k}$ has been found, E_S will send back the $m_{S,k}$ for inspection by E_B. If E_B agrees on the match as well, the orchestration negotiation can start.

For the orchestration, E_B will send its request $O_B = \{(M_A, out), (M_O, in), (M_C, out), (M_D, in), (M_B, in), (M_P, out)\}$ to E_S. E_S will look in its repository of available orchestrations around these messages to see if it has a matching protocol, replacing d_i in each (M_i, d_i) by $\neg d_i$. When it finds one, it sends back the confirmation to E_B, which can choose to work with E_S now to work on an actual collaboration where the messages are exchanged in the indicated order governed by O_B, with message content defined by $M_{B \cap S}$, and using a network protocol $p_i \in P_{B \cap S}$.

6 Related Work

The key issue of inter-enterprise collaboration is *interoperability*. The trend to globalized markets has painfully made clear that many enterprise systems are not designed to interoperate with other systems of other enterprises [33]. Most of the problems emerge from proprietary development or extensions, unavailability or oversupply of standards, and heterogeneous hardware and software platforms. A major challenge is to achieve and sustain interoperability in the face of planned and spontaneous changes, with proper alignment between and integrity of the business and technology levels [29]. Facing this challenge and developing solutions that not only solve current enterprise interoperability problems but also create new business opportunities, is one of the focal points of the European Commission [11,12].

Our contribution in this paper focuses on the exploration of new architectural patterns for collaboration (or interoperability). We explained the principle steps in the approach that embodies these patterns, but we are well aware that many issues need to be addressed and technological support need to be developed in order to make this work in practice.

To take some inspiration in this, we turn to some recent results which directly or indirectly relate to the problems we are addressing in this research. The mentioned results are in general in three directions: (i) Cross-organizational collaboration; (ii)

Service design and execution; (iii) Interoperability architectures. We will only mention several relevant recent examples of related work, one in each of these three directions, for brevity.

With regard to cross-organizational collaboration, Schroth has proposed a service-oriented reference architecture for business media that overcomes the typical B2B software drawbacks, by considering four main views, namely community (structural organization), process (process-oriented organization), services and infrastructure [25]. Concerning services, a service bus actually enables and facilitates interactions on the basis of operational as well as coordination services.

Concerning service design and execution, some new service models have been introduced, such as the one proposed by Esper [10], which is essentially driven by business interoperability constraints.

As it concerns interoperability architectures, a meta-model has been proposed by Ullberg [30], which meta-model could support the creation of enterprise architecture models amenable to service interoperability analysis. This can be represented using influence diagram with attributes affecting service interoperability.

All this experience has inspired us mostly indirectly for our architectural contribution, and the analysis presented in this section further justifies in our opinion the claim that current approaches (including standard and broker-based) are insufficiently capable of facilitating inter-enterprise collaborations.

7 Conclusions

In this paper, we have presented solution directions that concern inter-enterprise collaboration and in particular, the desired capability of enterprises changing flexibly their (supply-customer) networks. We propose architectural guidelines for establishing the inter-enterprise collaboration protocols, content structure, and orchestration (the inter-enterprise process) *before* the exchange of actual content takes place.

Distinctive features of the proposed guidelines are the close-to-real-life service-oriented collaboration and related to this: no need for support through all-encompassing standards and/or undesired third parties (brokers). Moreover, with regard to multiple partner's needs to find matches or near matches, we have addressed the challenge of matching the protocols, content structures, and orchestration of multiple partners.

To further this research, we plan to: (i) elaborate more on the solution we have proposed, by considering functional and informational architectural issues; (ii) propose an implementation related to the suggested knowledge-based traceability, possibly using Prolog.

Acknowledgements. This work has been supported by the Systems Engineering Dept. at TU Delft and by the Freeband A-MUSE project (http://a-muse.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services, Concepts, Architectures and Applications. Springer, Heidelberg (2004)
2. ASC X12 2008, `http://www.x12.org`

3. Bowersox, D.J., Closs, D., Cooper, M.B.: Supply Chain Logistics Management. McGraw-Hill, USA (2002)
4. Boyson, S., Corsi, T.M., Dresner, M.E., Harrington, L.H.: Logistics and the Extended Enterprise. John Wiley, USA (1999)
5. Boyson, S., Corsi, T.M., Verbraeck, A.: The E-supply Chain Portal: A Core Business Model. Logistics and Transportation Review Part E 39, 175–192 (2003)
6. Boyson, S., Harrington, L.H., Corsi, T.M.: In Realtime: Managing the New Supply Chain. Greenwood Publishers / Praeger (2004)
7. Camarinha-Matos, L.M. (ed.): Collaborative Business Ecosystems and Virtual Enterprises. Kluwer Academic Publishers, Dordrecht (2002)
8. Corsi, T.M., Boyson, S., Verbraeck, A., Van Houten, S.P.A., Han, C., MacDonald, J.: The real-time global supply chain game: New educational tool for developing supply chain management professionals. Transportation Journal 45(3), 61–73 (2006)
9. ebXML 2008, http://www.x12.org
10. Esper, A., Sliman, L., Badr, Y., Biennier, F.: Towards Secured and Interoperable Business Services. In: Mertins, K., Ruggaber, R., Popplewell, K., Xu, X. (eds.) Enterprise Interoperability III. Springer, Heidelberg (2008)
11. European Commission: Enterprise interoperability research roadmap. Version 4.0 (July 2006), ftp://ftp.cordis.europa.eu
12. European Commission: Unleashing the potential of the European knowledge economy. Value proposition for enterprise interoperability. Version 4.0 (January 2008), ftp://ftp.cordis.europa.eu/pub/ist/docs/ict-ent-net/isg-report-4-0-erratum_en.pdf
13. Gelernter, D., Carriero, N.: Coordination Languages and Their Significance. Communications of the ACM 35(2), 97–107 (1992)
14. Honig, J.: Towards On-line Logistics: the LinC Interaction Modeling Language, PhD Thesis. TU Delft Press (2004)
15. IKEA 2008, http://www.ikea.com
16. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)
17. Newcomer, E.: Understanding Web Services, XML, WSDL, SOAP and UDDI. Addison-Wesley, Boston (2002)
18. Orriens, B.: Modeling The Business Collaboration Context, Technical Report, Tilburg University, The Netherlands (2006)
19. Orriens, B., Yang, J.: Establishing and Maintaining Compatibility in Service-Oriented Business Collaboration. In: 7th International Conference on Electronic Commerce (2005)
20. OXpedia 2008, http://www.open-xchange.com/en/oxpedia
21. Papazoglou, M.: The World of e-Business: Web-Services, Workflows, and Business Transactions. In: Bussler, C.J., McIlraith, S.A., Orlowska, M.E., Pernici, B., Yang, J. (eds.) CAiSE 2002 and WES 2002. LNCS, vol. 2512, pp. 153–173. Springer, Heidelberg (2002)
22. Papazoglou, M.P.: Web Services: Principles and Technology. Addison-Wesley, Reading (2007)
23. Poirier, C.C.: Advanced supply chain management. Berrett-Koehler Publishers, San Francisco (1999)
24. Rosettanet 2008, http://www.rosettanet.org/cms/sites/RosettaNet
25. Schroth, C.: A Service-oriented Reference Architecture for Organizing Cross-Company Collaboration. In: Mertins, K., Ruggaber, R., Popplewell, K., Xu, X. (eds.) Enterprise Interoperability III. Springer, Heidelberg (2008)

26. Shishkov, B., Dietz, J.L.G., Liu, K.: Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In: ICEIS 2006 8th Int. Conf. on Enterprise Inf. Systems (2006)
27. Shishkov, B., Van Sinderen, M.J., Quartel, D.: SOA-Driven Business-Software Alignment. In: ICEBE 2006, IEEE International Conference on e-Business Engineering (2006)
28. Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E.: Designing & Managing the Supply Chain, 2nd edn. McGraw-Hill, USA (2003)
29. van Sinderen, M.J., Johnson, P.C., Kutvonen, L.: Report on the IFIP WG5.8 Int. Workshop on Enterprise Interoperability (IWEI 2008). In: ACM SIGMODD Record (2008) (to appear)
30. Ullberg, J., Lagerstrom, R., Johnson, P.C.: Enterprise Architecture: A Service Interoperability Analysis Framework. In: Mertins, K., Ruggaber, R., Popplewell, K., Xu, X. (eds.) Enterprise Interoperability III. Springer, Heidelberg (2008)
31. United Nations: Electronic Data Interchange for Administration, Commerce and Transport (2003), http://www.un.org/Pubs/whatsnew/e99v06.htm
32. Van de Kar, E.A.M., Verbraeck, A.: Designing Mobile Service Systems. Research in Design Series, vol. 2. IOS Press, Amsterdam (2007)
33. Wang, H., Zhang, H.: Enabling Enterprise Resources Reusability and Interoperability Through Web Services. In: ICEBE 2006, IEEE Int. Conf. on e-Business Engineering (2006)
34. XML 2008, http://www.w3.org/XML