Services Discovery as a Mean to Enhance Software Resources Sharing in Collaborative Networks

Alexandre Perin-Souza, Ricardo J. Rabelo

Services Discovery as a Mean to Enhance Software Resources Sharing in Collaborative Networks

Alexandre Perin-Souza, Ricardo J. Rabelo
Department of Automation and Systems - Federal University of Santa Catarina – Brazil
{perin, rabelo}@das.ufsc.br

Abstract. Collaborative Networks (CN) realization fundamentally relies on the need of collaboration, from diverse perspectives, among partners. From the software perspective, CN members who have SOA-based solutions keep the involved web-services at their local silos. This means that the collaboration potential of CNs might be enlarged and reinforced if such silos could be opened up and shared among their members, hence decreasing development and hosting costs. This is relevant as CN members are mostly composed of small and medium size companies with usual high limitations of ICT resources. This paper presents the result of an exploratory research that proposes a model for service discovery as a mechanism to leverage software services sharing among partners under the Software-as-a-Service (SaaS) access and business models. The developed model allows discovering the most suitable services for the required business processes, considering their contexts (using the UBL standard) and quality of services (QoS). A prototype has been developed to show the concepts. The assessment of the model is given in the end.

Keywords: Collaborative Networks. Web Service Discovery. Business Process. QoS. UBL. SaaS.

1 Introduction

Collaborative Networks (CN) paradigm has been considered one of the most promising strategies. Its ultimate goal is to enable networked organizations to agilely define and set up relations with organizations as well as to be adaptive according to business environment conditions and current organizations' autonomy levels [1]. This requires more effectiveness, flexibility and collaboration in businesses.

From the ICT perspective, new requirements have demanded more advanced infrastructures [2] and a number of ICT approaches have been proposed for that. Two of the most impacting ones are SOA (Service Oriented Architecture) and Utility paradigms [3], which jointly represent a scenario of a large scale of software services distributed over the Internet and accessed on demand, from everywhere, anytime.

The problem tackled in this work refers to the fact that CN members who have SOA-based solutions keep the involved web services stored in their local silos. This means that the SOA potential in terms of reuse could be extended and highly increased if such silos could be shared among CN members. Services could be then accessed by any member so enlarging and reinforcing collaboration while development and hosting costs are decreased. This is relevant as CN members are mostly composed of Micro, Small and Medium Enterprises, without conditions to

maintain IT infrastructures and costly staff. In this sharing scenario, CN members could be both clients and service providers [2].

This is the underlying motivation of this work. CN members - VBE, VE, VO, PVC[1] members and other type of companies (e.g. logistics operators and software providers) have the potential to enlarge their collaboration via an interoperable and transparent collaboration "cloud". Several works have been developed in this direction, such as virtual machining [5], knowledge search and sharing over CN's information repositories [6], virtual shop-floor [7] and CN ICT infrastructures [2].

Supporting this higher collaboration requires, however, coping with many issues, of different levels of complexity. One of the issues refers to the access mode and business models that can be aligned to this collaborative scenario [8]. In the SOA context, SaaS (*Software-as-a-Service*) [9] has arisen as one of the most powerful models. Using SaaS, clients (i.e. CN members) can flexibly build and adapt their services portfolio according to their needs [10] instead of getting held to single vendors of monolithic software packages whose full set of functionalities are often few used/accessed. With SaaS, services are accessed remotely, upon request, paid-per-use, based on contractual rules specified in SLAs (*Service Level Agreement*) [11] for hosting, managing, providing access to them following QoS levels, no matter where the services providers are and how services have been deployed [12]. This seems suitable to CN members due to their intrinsic independence, autonomy, large geographic distribution and heterogeneity. A second issue refers to the discovery of services that are shareable, and this is what this paper is more about.

In this services discovery scenario, client and provider perspectives should be taken into account. From the CN clients' side this means how expressing the desired service as well as how finding, selecting and binding services to their composite SOA-based applications. From the CN providers' side, this means how publishing and making their services available. Besides that, CN clients must feel confident to access not only a given functionally-compliant service, but ideally to the most suitable service regarding the computing environment and business process' context [13].

The essential problem addressed in this paper is related to how to discover the most suitable software services dynamically over plenty ones that are made available at CN members' repositories. In this sense, this work presents a comprehensive, integrated, open and standard-based environment for dynamic services discovery and sharing, strongly based on processes and ICT standards, regarding CN particularities.

The paper is organized as follows. Section 1 has framed the problem and the tackled scenario of services discovery and sharing among CN members. Section 2 summarizes the main concepts and literature overview on services discovery. Section 3 presents the proposed model and section 4 presents a software prototype and section 5 discusses the results. Some conclusions are presented in section 6.

[1] VBE (Virtual Organization Breeding Environment), VE (Virtual Enterprises), VO (Virtual Organization), PVC (Professional Virtual Communities).

2 Basic Literature Overview

The problem of web services discovery involves many complex issues, such as: i) technological heterogeneity and low level of interoperability; ii) ambiguity of concepts due to differences of domain applications; and iii) limitations in the technologies used to design services [15]. There are plenty of works and visions on services discovery in the literature. The review here presented encompasses only the considered most relevant works for the purpose of this paper.

Two basic strategies resume the works on service discovery presented in the literature: static and dynamic discovery. There is a number of advantages and disadvantages in both. In the static way services are discovered at design time and they are immediately associated / bind to the given SOA application, but the previously chosen service may be not available when the service is invoked. The main advantage of the dynamic approach is the possibility of replacing services by others on-the-fly. However, this strategy usually presents higher discovery time (as all discovery actions should be done at execution time) and demands more sophisticated discovery algorithms. This sophistication can be in the form of criteria relaxation [16] or other techniques such as classification systems, providers' reputation, costs negotiation and quality of service development process [17].

Existing models and approaches for services discovery can be grouped into four dimensions: *information retrieval, architecture, QoS* and *standards* [15]. In general terms, in the information retrieval dimension the focus has been put on semantics in way to provide greater precision in the services selection [18]. In the architectural dimension, works have focused on aspects such scalability, security, availability, etc. [19]. In the QoS dimension there are three basic fields of research [20]. The first one refers to the definition of attributes and metrics for QoS. The second one involves the establishment of more comprehensive and robust frameworks to represent, select, verify and maintain QoS attributes. The third field involves the development of *ad hoc* strategies to solve specific cases involving QoS. In the standards dimension, the initiatives have focused on interoperability (in heterogeneous environments). ICT standards like UDDI [21], SOAP [22], WSDL [23] have been intensively used.

The problem is that the envisaged collaborative scenario requires a more holistic view about services discovery. Besides requiring a joint view of those four dimensions, other dimensions not tackled in any of the reviewed works are also necessary, namely the consideration of the business level. In spite of the very high complexity all this represents and challenges that have to be faced, this proposed model intends to contribute to this.

3 Proposed Model

Coping with all the requirements involved in the envisaged collaborative scenario of services discovery and sharing is extremely complex and also involves several challenges at business and IT levels [24].

In the developed model for services discovery within a CN scenario, there are five basic issues to be considered: *i) what to express; ii) how to express the desired*

service; iii) who expresses the desired service; iv) who evaluates and how discovery results are evaluated; and v) when expressing and searching.

The first point refers to which information should be expressed to specify the desired service for the given BP. In the proposed model, this involves: i) functional aspects (service's name and operations); ii) inputs and outputs; iii) expected QoS values; and iv) BP's context.

The second point refers to services expression including the components and the relationship between them. Natural language, formal or dedicated languages are examples of how to do this. The proposed model adopts ontology. They create a uniform vocabulary to be used by the discovery mechanisms, besides improving discovery process accuracy and precision.

The third point refers to who informs the desired service. Traditionally, the SOA application designer is the one who explicitly specifies the details of the desired service. Considering the role of BPM systems in this work, four ways might be considered: Automatic (A), Strongly Based on Designer (SBD), Semi-Automatic (SA) and Assisted (As). In the 'A', the discovery task is left to the BPM environment. It automatically identifies needs, makes changes in the expression and determines which service will be selected and be bind to a given BP. This process occurs without designer intervention. In the 'SBD' way is somehow the opposite case, where the designer is the responsible for indicating the required service and for evaluating and deciding about the most adequate service to be selected. The designer interacts with the BPM environment intensively. 'SA' is a hybrid way. The BPM does the whole process but asks to the designer for some constraint relaxation in the case no expected service is found out. In the 'As' way, there is a closer interaction between BPM environment and designer. The designer specifies service's requirements whereas BPM informs BP's context. The discovery evaluation is jointly performed.

The fourth point involves determining who evaluates the discovery results and how this evaluation occurs (assisted, without designer participation, etc). Four situations are considered: i) more than one 'perfect' service (i.e. there is full compliance between desire and outcome), which means having the need for ranking and selecting the most appropriate service; ii) just one perfect service; iii) none service; iv) one partial service (any 'perfect' service); and v) more than one partial services.

The fifth point refers to at which moment the activities of identifying the BP, its requirements, the results evaluation, etc. will be made by of the discovery actions.

A key aspect in the proposed model is a change in the way the problem of dynamic services discovery have traditionally been seen. In this work, the problem is split into two phases: *design* phase and *running* phase (Fig. 1). The essential rationale of this separation is trying to prevent CN managers from handling IT issues as much as possible. In this sense, at BPM level, it is considered that they are familiarized with business rules and can specify how business processes should be executed. On the other hand, at the running time, users shouldn´t be worried about discovery itself as this should be essentially left to the SOA environment / discovery system to do it.

The design phase refers to the specification of the SOA application to be built, including capturing the given BP's context and indication of the required QoS, which are crucial elements for a more proper discovery (left side of Fig. 1). This is done off-line and CN managers are assisted by the composition environment (see third point above). Initially, they use a BPM environment for composing applications (step 1)

which offers the access to a standard *BPs catalog* (step 2). This catalog, an additional element of the model, comprises all the business processes of UBL standard (the one chosen in this exploratory work), meaning that all composite applications will be compliant with a standard or with a specialization of it. Yet, it is possible to identify and to capture the BP's context. This speeds up the application design and provides greater semantic richness to the discovery since services (from the several CN providers) are published taking a UBL standard ontology into account.

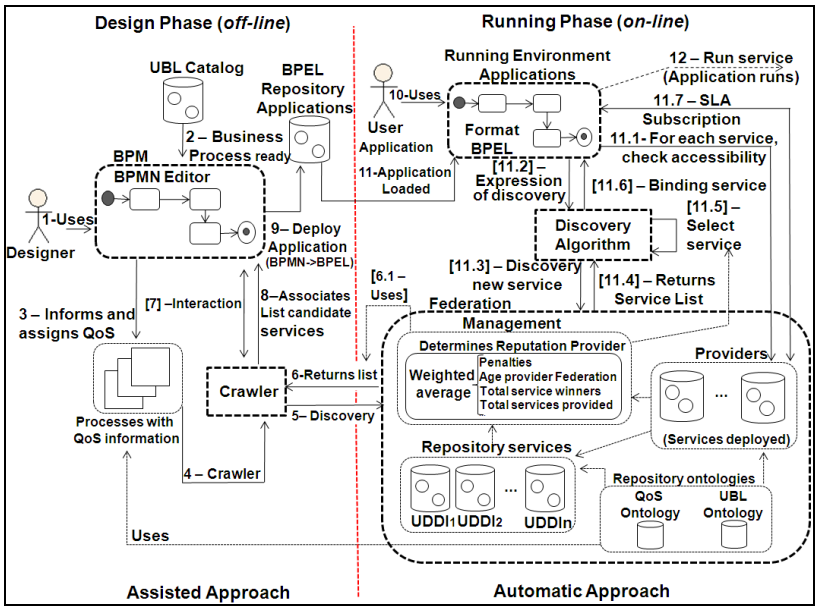

Fig. 1. Overview of the Model Operation.

In step 3, CN managers, together with an IT technician, inform and assign QoS attributes for each BP's activity. A QoS ontology (see section 4) is used for this (Fig. 1 right / bottom), and it is also used by CN providers when publishing their services. The search expression is then assembled. The expression to find the web service W is structured as a 2-*tuple* $W = (F,Q)$, where F is the set of functional requirements identified via the BP's ontology, and Q the non-functional requirements (i.e. QoS).

An internal crawler service is activated (step 4) having the search expression as input. The crawler begins searching (step 5) for services in the CN members' services repositories with the aim of bringing a list of possible services that matches the subprocess requirements. It is also important to mention that the crawler acts in "background" (i.e. candidate services are retuned at design time and not at the running time). The underlying strategy is to decrease the whole discovery time, avoiding an exhaustive search when the composite SOA application would be in execution.

A list of services is returned (step 6) and further ranked according to CN providers' reputation (step 6.1 - historical, penalties, etc). As mentioned in the previous section, a 'perfect' matching may be not occur, and the crawler should handle this (step 7). If the list contains just one service, it is already settled. If the list is empty, the crawler asks for a requirements relaxation for a new search, or leaves the option (or risk) of

keeping the same requirements and doing the search at running time. If the list has more than one service, they are stored as XML files for further treatment at run time. In step 8, still at the design phase, this list of candidate services is passed to the BPM environment. The SOA application is then composed, converted into BPEL [25] format and gets available to run (step 9).

It is important to clarify one aspect related to the used ontologies and the matching. The use of ontologies is a strategy to face semantics interoperability problems when consumers go for a discovery and providers publish their services. UBL and QoS ontologies act as a common and agreed set of terminologies and concepts used to express BPs and QoS. This allows a syntactic matching in the discovery, via a comparison of the same (semantic) terms used in the adopted ontologies.

The second phase of the model – the on-line phase – starts when the CN manager or collaborator wants to run a given application (that one previously composed). At this time, it is relatively transparent to him that it is a SOA-based application. He navigates through the process / BPEL repository and chooses which application he wants to run (steps 10 and 11, Fig. 1 right side). A runtime task checks if the pre-selected web services are actually available (step 12), invoking the selected ones in the case they are ok. Otherwise the second one from the list will be picked up, and so forth. In the case none of them are available then the whole discovery activity is triggered again, with a new possible list, etc. Once everything is settled for a given service, the respective SLA (Service Level Agreement) is dynamically generated (step 11.7) and integrated into the BPM environment, as proposed in [11]. Thus, and helped by a BPEL engine, the desired composite SOA application is ready to run, with the most appropriate services (step 12).

This proposed model has a set of assumptions. Two of them are important to be highlighted. The first assumption is what we call 1:1 relation. This means that the composition model is prepared to find and to compose *1* service for each *1* BP's activity. For instance, regarding UBL ontology, *Ordering* process is composed of a set of activities e.g. *AcceptOrder*. Therefore, the model will understand that a discovery action (considering functional, QoS and context requirements) will be carried out aiming at finding a sort of web services for *AcceptOrder* but only one web service will run it. This assumes that services providers would have implemented one (1) service per BP's activity (eventually they can have different versions, associated or not to different business models). The second assumption is that the so-called CN manager and IT technician are expected to be experienced on the company's processes and rules (and UBL ontology) as well as have the 'feeling' about suitable QoS metrics (and ontology) for the SOA application that is going to be built up.

4 Software Prototype

The architecture of the prototype is organized in five modules and one central internal repository of web services.

The *editor* module allows the definition of SOA-based applications (based on UBL standard). It uses the *IBM WebSphere Business Modeler* tool and supports BP management and BPMN modeling.

The *catalog* module implements a set of services and is itself a SOA application [26]. Fig. 2 shows the interface where the designer uses *WebSphere* tool and loads some process stored in the catalog. The designer knows the business rules, the context's semantics, the services' provider, the service's name and that the required SLA has been previously agreed. The SOA designer can either compose the application as presented by UBL (as a kind of default) or can compose it regarding companies' specificities. During this composition, the designer specifies QoS values and required composition's orchestration.

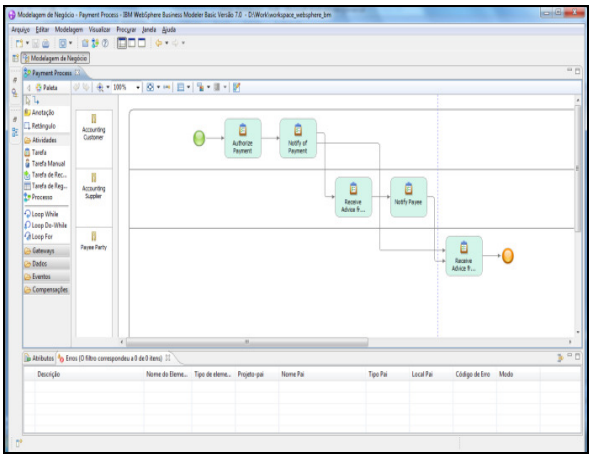

Fig 2. Interface of an editing process accessing the catalog.

The *discovery* module has a crawling. It takes as input the required service (using UBL ontology), the QoS constraints (using QoS ontology), a variable related to the provider reputation and the list of available UDDIs in the Federation. The algorithm creates and triggers threads for doing the search and the services selection over each UDDI. A looping for a systematic search for services that satisfy the discovery expression is started. The algorithm tests each service comparing functional aspects (name and adherence with the UBL ontology) and checks QoS specifications against what has been required. If they satisfy the discovery expression then the service is added to the service candidates list. Once the search ends, the variable which considers providers' reputations is checked. If it is true, the list of candidate services is classified (using providers' reputation) and it is returned by crawler (Fig. 3).

Discover Services for Tasks

Task Name	Task Ontology	QoS Constraints	Matching Services
Authorize Payment	ubl/payment/paymentprocess/accounting...	1	9
Notify of Payment	ubl/payment/paymentprocess/accounting...	1	9
Notify Payee	ubl/payment/paymentprocess/accounting...	1	9
Receive Advice from Accounting Customer	ubl/payment/paymentprocess/accounting...	1	15
Receive Advice from Accounting Customer1	ubl/payment/paymentprocess/payeeParty...	1	12

Fig 3. Sample of returned services after the crawling algortim.

The first column reports the list of names of services discovered (*AuthorizePayment, Notify of Payment, etc.),* the UBL's activity classification UBL is

presented in the second column (according to the developed ontology), the number of QoS characteristics desired for a given activity is shown the third column (1, in this case). The total number of discovered services that satisfy the expression of discovery is showed in the fourth column (*Matching Services*).

The forth module is used as an environment for running the composed SOA application. It was implemented using *Apache ODE* [27], integrated with *Intalio* BPMS tool. Intalio's web interface was used to visualize the status of running processes. This module uses the dynamic discovery algorithm (DDS) to test each candidate service in the list before invoking them; the DDS algorithm ends when it finds the first available service that meets the discovery expression. Similarly as in the crawling phase, this module also involves the generation and management of the SLA between the involved parts.

The fifth and last module refers to the implementation of the logical and virtual aggregation of providers (i.e. the CN members' services), which is called *federation*. It includes four elements: (1) Ontology repository (composed of UBL and QoS ontologies), both implemented using *Protégé* tool. QoS ontology was conceived based on a high-level ontology [28] and QoS definitions were based on [29]; (2) The set of repositories managed by the federation. The access to the repositories is made from a list of a XML file stored in a UDDI master (ubl-uddi). This was implemented using Apache jUDDI [30], which implements *UDDI 3.0*; (3) A general Management submodule, which determines the reputation of providers and; (4) UBL Providers, representing the UBL-compliant services providers that composes the federation.

5 Experiments and Evaluation

Four main experiments were performed to evaluate the developed model. The involved cases were deployed in a PC with 1.66GHz, 2GB RAM and Microsoft Windows XP. The first experiment aimed at checking the behavior of the prototype against the metrics of *precision* and *recall* (the two traditional metrics in the area of information retrieval [31]). The scenario was composed of 3 service repositories (uddi-ubl1, uddi-ubl2 and ubl-uddi3) and 10 providers registered in the federation (provider-ubl1 to provider-ubl10). A provider was chosen randomly and 10 services related to the UBL process *Ordering* were published. This process was repeated 20 times for each repository, applying the adopted UBL and QoS ontologies, giving a total of 600 services. These services are actually instances of the same process' activity, related to the *Ordering* process. Services' code was the same but with different QoS values assigned to each registered instance. These values were generated in a stochastic way, in the interval [0,100]).

The experiments 2 and 3 were performed using the same discovery expression but increasing the number of UDDIs to 5 and 7, all of them deployed in a local network. In the fourth experiment, 1 UDDI was locally deployed and 2 others were deployed remotely, simulating a more pervasive scenario. Table 1 shows the discovery time behavior and the total of services in the federation in the four experiments.

As imagined, the use of QoS has minimized the number of returned services while its usability (adequacy) for the current business and computing conditions is maximized. The use of UBL ontology mitigates interoperability / interpretation

problems. This is also valid to QoS, where both clients and providers can share a common conceptual vocabulary.

Table 1. Results of experiments.

Tests	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
	Federation with 3 local UDDIs		Federation with 5 Local UDDIs		Federação com 7 local UDDIs		Federation with 3 UDDIs (1 local e 2 distributed)	
	Service (T/R)	**Time (s)**	**Service (T/R)**	**Time (s)**	**Service (T/R)**	**Time (s)**	**Service (T/R)**	**Time (s)**
1º	600/17	9	1000/27	6	1400/36	8	600/11	9
2º	750/19	6	1250/31	6	1750/45	7	750/13	10
3º	930/25	6	1550/36	6	2170/55	8	930/18	12
4º	1170/29	7	1950/41	7	2730/71	9	1170/23	14
5º	1470/34	7	2450/49	7	3430/90	9	1470/30	16
6º	1860/41	8	3100/65	11	4340/115	18	1860/41	22
7º	2340/52	11	3900/72	15	5460/152	28	2340/55	26
8º	2940/70	15	4900/112	23	6860/188	26	2940/72	32
9º	3690/92	17	6150/147	25	8610/226	51	3690/85	40
10º	4620/109	24	7700/191	46	10780/284	67	4620/105	51

Tests column identifies the set of discoveries carried out in the four experiments. For example, in the first test, 17 services were found out (i.e. attended the discovery expression) over 600 available in the federation, taking 9 seconds to perform this operation (in a local network). In the other extreme, in the tenth test, 105 services were discovered over 4620 available services, and this took almost 1 minute to be executed in a deployment scenario composed of 1 local and 2 distributed / remote UDDIs. Although part of this important increasing of time is caused by the intrinsic latency in the network, a good part of this is caused by the discovery algorithm itself, which grows almost exponentially as long as the number of nodes / providers increases.

Despite the complexity involved in a complete solution for services discovery, the proposed model presents some contributions. The experiments showed that the most suitable services for a given process can be found out. The crawler made possible filtering non adequate services, hence allowing a potential quicker decision about the final service. The quality of the selection is higher as the list brought up by the crawler only contains services that effectively cope with the indicated requirements, meaning a lower potential to have problems during the execution of the (SOA-based) application. The discovery can be considered more trustful as there is some computing assistance in the process. Once the company detects a need for a change in a given process, the designer can immediately redesign the business processes and a new SOA application can be "immediately" composed with suitable services.

Based on the literature overview and achieved results, it is believed that the model introduces some aspects of innovation. One of them concerns the integration and use of a catalog of business processes in the composition of applications, combing it with a flexible and ubiquitous CN scenario based on SaaS access mode. Furthermore, the model uses the semantics obtained from the ontology of UBL-based business processes to publish and discover services, allowing getting the process' context to better filter services. This somehow ensures that the results from the discovery action will only bring services semantically aligned with business processes expressed in

UBL, i.e. with the BPM layer. The model uses of a logical entity (Federation), which highlights the importance of UDDI registries (or equivalent structure) in the discovery process as a way to provide some standardization in the publication process.

The presented results are very dependent of the adoption of BP standards and of the so-called 1:1 relation, two assumptions used in the model.

6 Conclusions

This paper has presented results of an exploratory work on dynamic services discovery as a mechanism to enhance collaboration among CN members at IT level with a sustainable business model (SaaS).

The proposed model creates a kind of cloud over CN members' services repositories and then a mean to access/share them. It is strongly based on standards, which mitigates interoperation problems and hence increases the potential of a larger acceptance of advanced concepts – BPM and SOA in the case – by CN members.

From the exploratory perspective of this research, it can be said that the proposed model is able to support the theoretical part of the problem (based on the chosen approach), which refers to how CN members can share their web services and enlarge their collaboration and trust building. This can give suitable conditions for a more agile adaptation in the members' business processes, in particular when they need to cope with specific business opportunities to create a virtual organization.

However, from the applied perspective of this research, a model like that cannot be seen as the solution for all issues involved in a full-fledged services discovery and services sharing model for CN. For example, SaaS in an emergent area and there is not solid theoretical foundations about it yet. Trust, adequate ICT infrastructures, security, cultural changes, BP (re)organization, integration with legacy systems & services, partners preparation to indeed offer their services under the SaaS base, the adoption of common ontologies (even though relied on standards), among many other aspects, are examples of additional and complex issues that should be dealt with.

From the SOA and composition points of view, it not believed that all enterprise applications should be transformed into services and accessed freely from any CN member. There are some applications that are naturally locally deployed and of private use besides the fact that some of them require very hard QoS metrics to be indeed useful, which happens in many cases. From the discovery algorithm itself, there is a need of a deeper analysis of its complexity to deal with potentially thousands of providers. This corresponds to one the main next steps of this work.

In spite of all this, it is believed that this model has a structure that can be used as a starting model and useful instrument to allow a wider collaboration among CN members, preserving their independence, autonomy and heterogeneity.

References

1. Camarinha-Matos, L. M., Afsarmanesh, H. Collaborative Networked Organizations: a Research Agenda for Emerging Business Models. Kluwer Academic Publishers (2004)

2. Rabelo, R. J., Advanced Collaborative Business ICT Infrastructures. In: Methods and Tools for Collaborative Networked Organizations, Springer, pp. 337-370 (2008)
3. NESSI Strategic Research Agenda - Framing the future of the Service Oriented Economy. V. 2006-2-13, http://www.nessi-europe.com/documents/ NESSI_SRA_VOL_1_20060213.pdf , ICT for Enterprise Networking, http:// cordis.europa.eu/ist/directorate_d/en_intro.htm
4. Singh, M., Huhns, M. Service Oriented Computing, Wiley (2005)
5. de_Souza_Jr., J.L.N.J., *et al.*, An Internet-oriented Management and Control System in a Distributed Manufacturing Environment. International Journal of Manufacturing Research, vol. 5, pp. 5-25 (2009)
6. Tramontin_Junior, R.J., Rabelo, R.J., Chirab, H. Customizing Knowledge Search in CNOs through Context-based Query Expansion. Production Planning & Control, vol. 21(2), pp. 229-246 (2010)
7. Ribeiro, L., Barata, J., Colombo, A. Supporting agile supply chains using a service-oriented shop floor. Engineering Application Art. Intelligence (2009)
8. Borst, I., *et al.*, Technical Report (Deliverable) D62.2 ICT-I Business Models, http://www.ve-forum.org/default.asp?P=284
9. Cancian, M. H., Rabelo, R. J., Wangenheim, C. G. V. An approach for the generation of SLAs for Software-as-a-Service [in Portuguese]. In: 8th I2TS Int. Conference. Florianopolis. Brasil (2009)
10. Tsai, W.T. Service-oriented system engineering: a new paradigm. In: Service-Oriented System Engineering, SOSE. IEEE Inter. Workshop, pp. 3-6 (2005)
11. Cancian, M. H., Rabelo, R. J., Wangenheim, C. G. V. An approach for the generation of SLAs for Software-as-a-Service [in Portuguese]. In: 8th I2TS Int. Conference. Florianopolis. Brasil (2009)
12. Ma, D. The Business Model of Software-As-A-Service. In Services Computing. IEEE International Conference, pp. 701-706 (2007)
13. Perin de Souza, A., Rabelo, R. J. An Approach for a more Agile BPM-SOA Integration supported by Dynamic Services Discovery. In Proc. of the EDOC 14th IEEE, Brasil (2010)
14. Perin de Souza, A., Rabelo, R. J. Supporting Software Services Discovery and Sharing in Collaborative Networks. In 11th IFIP Working Conference on Virtual Enterprises, Saint Etienne (2010)
15. Garofalakis, J. D., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.K. Contemporary Web Service Discovery Mechanisms. Journal of Web Eng., pp.265-290 (2006)
16. W3C. Web Services Architecture (WSA), http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
17. Mello, E. R., Fraga, J. S., Wangham, M. S. Using a trust model for the composition of Web Services [in Portuguese]. In: Brazilian Symposium on Computer Networks (2009)
18. Kourtesis, D., Paraskakis, I., Friesen, A., Gouvas, P., Bouras, A. Web Service Discovery in a Semantically Extended UDDI Registry: The Case of Fusion. Book Series. IFIP International Federation for Information Processing. Book Establishing The Foundation of Collaborative Networks. Publisher Springer Boston, vol. 243, pp.547-554 (2007)
19. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S. Preference-based Selection of Highly Configurable Web Services. WWW '07 Proceedings of the 16th international conference on World Wide Web. Alberta, Canada (2007)
20. Wang, X., Vitvar, T., Kerrigan, M., Toma, I. A. QoS-aware Selection Model for Semantic Web Services. Service-Oriented Computing – ICSOC (2006)
21. OASIS. UDDI Specification V 3.0.2, http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html
22. W3C. SOAP Specification V. 1.2, http://www.w3.org/TR/soap12-part1/.

23. W3C. Web Services Description Language Specification V 2.0, http://www.w3.org/TR/wsdl20/
24. Dorn, J.; Grün, C.; Werthner, H. & Zapletal, M. From business to software: a B2B survey. Information Systems and E-Business Management, Springer, vol.7, pp.123-142 (2009)
25. OASIS. Web Service Business Process Execution Language V 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
26. Bezerra, R. O. Proposal for Electronic Catalog of Business Processes Based on UBL for Composition of SOA Applications [in Portuguese]. M.Sc. Thesis, Federal University of Santa Catarina (2011)
27. Apache Fundation. ODE, http://ode.apache.org
28. Tondello, G. F. Semantic specification of QoS: The QoS-MO ontology [in Portuguese]. M.Sc. Thesis, Federal University of Santa Catarina (2008)
29. W3C. QoS for Web Services: Requirements and Possible Approaches, http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/
30. Apache Fundation. jUDDI, http://juddi.apache.org/.
31. Rijsbergen, C., J. van. Information Retrieval, http://www.dcs.gla.ac.uk/Keith/Preface.html