A Plug and Play Integration Model for Virtual Enterprises

Juan D. Méndez, Ricardo J. Rabelo, Fabiano Baldo, Maiara H. Cancian

A Plug and Play Integration Model for Virtual Enterprises

Juan D. Méndez[1], Ricardo J. Rabelo[1], Fabiano Baldo[2], Maiara H.Cancian[3]

[1] Department of Automation and Systems Engineering, Federal University of Santa Catarina
Florianopolis (SC) Brazil
juand.mendez28@gmail.com; ricardo.rabelo@ufsc.br

[2] Department of Computer Science, Santa Catarina State University
Joinville (SC) Brazil
fabiano.baldo@udesc.br

[3] Estacio University
SC401 Road km 1 – Florianopolis (SC) Brazil
maiara.cancian@estacio.br

Abstract. Systems integration is a key issue to be faced when supporting Virtual Enterprises (VE) execution. However, it is very complex regarding the dynamic composition, autonomy and large distribution of its members as well as the high heterogeneity of IT and business processes' models used by them. It has been realized by many works in the literature that SMEs should have some preparedness, including at the IT level, in order to create a feasible solution for them to get onboard and agilely interoperate in real VEs. This work intends to overcome some drawbacks of current approaches for that, proposing a model where VE partners can more easily get into a VE when it is created (*plug*) so that the VE can be executed seamlessly during its operation (*play*) and dissolution (*unplug*), including the support of some level of semantic interoperability. A prototype has been implemented, and results are discussed.

Keywords: Virtual Enterprises, Service Oriented Architecture, SOA, Systems Integration, Semantic Interoperation, Ontologies, Enterprise Service Bus, ESB

1 Introduction

Virtual Enterprises (VE) are increasingly becoming a net-working pattern to boost companies' agility. Bearing in mind that one of the fundamental VE's properties is that the transactions between its members should be predominantly done via computer networks [1], systems integration becomes a cornerstone issue.

Broadly, VE integration means allowing companies (and their computing systems) to be seamlessly and agilely tied up and to interoperate so as to more effectively support the execution of the involved VE's business processes [2].

This scenario imposes considering some tough aspects in terms of systems integration, like as [2,3]: the dynamic, temporary and simultaneous belonging of companies to different VEs throughout their life cycles; the massive use of computing networks required to support the communication between VE members regarding the collaborative and sharing natures of the work; companies are geographically distributed and independent, adopting different IT, business process (BP) models, terminologies and working methods; the different governance rules imposed to each VE; and the different security domains to be crossed by the BPs' transactions.

Dealing with all these aspects is very complex. EU Research projects, like PRODNET, DAMASCOS, CrossWork, ATHENA, ECOLEAD and COIN [1], have addressed them in the last two decades providing comprehensive frameworks that embrace the entire VE life cycle at variable levels of depth, although being naturally restrained by the advances of IT and Internet in the time. In order to decrease the problem complexity, some assumptions have been usually made, such as [2,3,4]: companies are 'somehow' already and properly IT-enabled to participate in a VE; VE members adopt the same integration technologies as well as BP models and terminologies; in practice, once the VE is created it does not change its composition; and they basically do not handle the VE dissolution phase. Besides that, they are either theoretical or too general when trying to propose solutions to face those assumptions, or they implement some solutions using approaches that ended up leaving VE partners' integration a not so agile and seamless process [3].

This paper is not another wide framework like those ones. Instead, it focuses on the VE integration part, approaching it in a different way towards decreasing the mentioned assumptions. Its contribution is represented by a so-called *plug & play* model, where VE partners (i.e. their IT systems) can more easily get into a VE when it is created (*plug*) so that the VE can be executed seamlessly during its operation (*play*) and dissolution (*unplug*), including some level of semantic interoperability.

The model conception and its implementation rely on SOA (*Service Oriented Architecture*) [5], open IT and BP standards, integration patterns, and on ESB (*Enterprise Service Bus*). A software prototype has been implemented to evaluate the proposed model in a controlled environment and results are presented and discussed.

This paper is organized as follows. Section 1 has introduced the problem and the objectives of the work. Section 2 summarizes the main outcomes of related works. Section 3 presents the proposed plug & play model. Section 4 describes the implemented prototype and the achieved results. Section 5 presents some preliminary conclusions and the next main steps of this work.

2 Literature Review

There are different definitions for systems integration and interoperation. In this work the following ones were adopted: *systems integration refers to the ending goal of making a global system architecture to work united and completed, involving coordination, coherence and uniformization. Systems interoperability refers to a means to achieve integration, endowing systems with the ability to exchange/use parts (data, and functionalities or services) of another system without loss or distortion, aiming at mutual understanding within a loosely-coupled federated environment* [3].

As mentioned before, a number of works have been addressing systems integration and interoperation in networked organizations (as VEs). They basically set up a (sometimes complex) wrapper, using different IT, and a communication layer over SMEs' systems to make them interacting via a common, predefined and mostly proprietary BP model, with almost none semantic interoperability support, and with quite low means for VEs to indeed work as a dynamic network in a more agile process of get-in and get-off from the VE throughout its life cycle [2,3,4,8,9].

[1] https://www.cordis.europa.eu/project/rcn/35881_en.html, 53702_en.html, 71382_en.html, 72762_en.html, 74487_en.html, 85550_en.html.

In terms of approaches to tackle semantic interoperability in SOA and/or ESB-based environments, the following ones can be highlighted.

Schratzenstaller *et al.* [6] proposed the creation of a single ontology to support semantic interoperability in VEs without the need to annotate the partners services' interfaces. An ESB is used and enriched with the ontology, but the entire process should be done manually, non-automatically, including partners' identification and the configuration of their services' references and the equivalent payload terms in the ESB. They do not support at all the VE dynamics and partners unplugging.

Zhu [7] developed a prototype to enhance the ESB with semantic mediation. A central ontology is created representing the proprietary enterprise's vocabulary, whose concepts are considered in the services' payloads. Services' descriptions are annotated based on these concepts using SAWDSL for SOAP web services, and ontologies for REST services are created using WSMO Lite (a W3C submission for semantic descriptions). All the mappings between the services payload's terms and the enterprise vocabulary are predefined and manually set up. Once this is done, the prototype can do the mediation between different payloads in run time. This work has considered just one company, and not a VE and the related issues.

Shi *et al.* [8] proposed an ontology-driven integration framework for heterogeneous systems based on SOA. It is divided into three main modules: integration control, service bus and system integration. A hybrid ontology architecture is used to reduce the high coupling and the number of mappings between ontologies. All mappings are carried out manually. This work has considered a hypothetic BP model as well as a generic company scenario, and not a VE and the related issues.

Khalfalla *et al.* [9] proposed a two-phase model to ensure semantic interoperability in a cloud based platform, enabling collaboration within networked organizations in the aerospace industry. The model uses a central ontology to support semantic interoperation, which serves as a major reference in the collaboration. In the first phase (off-line) a sort of actions are performed: creation of the reference and mapping ontologies, the publishing of enterprises systems' APIs as web services considering their native data models, and the creation of rules to make the conversion and mapping between the these models and the reference ontology. The second phase (on-line) allows companies collaboration using those rules and a transformation service that automatically executes the transformations between the different companies' data models. As this scenario focuses on data exchange, without considering the coordinated execution of BPs, ESB is not used. Although this scenario has comprised networked organizations, they did not support a truly VE.

3 The Plug & Play Integration Model

A fundamental assumption in the proposed model is that all VE members belong to a long-term alliance of type VBE (*Virtual organization Breeding Environment*) [1], which is intrinsically grounded on trust, resources sharing, members' autonomy, common working principles and whose members have the willingness to collaborate.

The model's design principles have tried to follow some elements of advanced visions on integration of VBE- and VE-like networks involving SMEs, which include [4,10,11]: open, non-intrusive, scalable, loosely-coupling and service-oriented architectures, plug & play collaboration infrastructures, and integration at the design

time and at the run-time as well. Considering these principles, the model was designed to support the integration of companies' systems in VEs so that they can be called to execute the required VE's BP activities (considering semantic interoperability) and hence the VE can seamlessly operate and further be dissolved. For that, the model splits the VE integration process into three major phases:

I. *Plug-in* phase: preparation phase executed when a member is recruited by a VBE and its systems should be prepared to work with when VEs were created.
II. *Play* phase: the phase where VBE members are selected to participate in VEs and their systems are integrated and interoperate in order to execute the required VE's BPs. Considering the VE reference framework proposed in [2], this phase assumes that activities as VE planning, partners' search and selection, negotiation and contracting have already and somehow been done.
III. *Unplug* phase: the phase where VE members' systems are logically disconnected from the VE when a given member leaves it.

Considering the VBE life cycle stages [14], the *plug-in* phase is executed in the *VBE creation* stage, (*ICT Setup* and *Member population* steps). The *play* phase is executed when the VE is created (VE *creation* stage) and operated (VE *operation* and *evolution* stages). The *unplug* phase is executed in the *evolution* or *dissolution* stages.

The model relies on two basic approaches regarding its design principles and goals: the use of reference models for BP modeling and of service orientation (SOA) as the underlying basis for systems implementation.

BP reference models, like *EDIFACT, Rosettanet* and *ebXML*, have been used by companies since many years as a "*lingua franca*" to reduce semantic interoperability problems in the transactions between business partners. In this work the UBL (*Universal Business Language*) open BP reference model supported by OASIS [12] has been adopted as its BPs are devised to support supply chains, which is a type of network quite close to VEs. UBL 2.2 is composed of 68 processes, each of them having activities and data structures to be exchanged when they are executed.

This does not mean that all VBE members must have to adopt it. UBL is internally used in the model's architecture as a meta-model which all transactions between VE members are converted and based on. Actually, the model's architecture is open to work with any BP reference model, be it standard or proprietary. It is up to the VBE organization, or even to the VE, to decide about which BP model is to be adopted.

Service Oriented Architecture (SOA) has been increasingly used to deliver interoperable, flexible and loose-coupled solutions [4]. Although SOA is technology independent, some open standards have been developed to implement its concepts, as *Web Service, SOAP, WSDL* and *BPEL*, from W3C[2], which are used in the model internal implementation. In the same way, VBE members' systems do not have to adopt them at all as the model's architecture is open to support multiple technologies. This flexibility is granted by another technology, the ESB. The ESB (*Enterprise Service Bus*) acts as a computing infrastructure that provides secure and reliable communication involving heterogeneous and loose-coupling environments.

However, ESB commercial software do not handle semantics. In order to overcome this, ontologies have been considered as the most suitable approach [4,6]. The proposed model supports the OWL and OWL-S[3] semantic web technologies,

[2] https://www.w3.org
[3] https://www.w3.org/Submission/OWL-S and www.w3.org/TR/owl2-overview

delivering the UBL ontology to represent the information that is exchanged within the VE and the ones representing the native semantics of each VE member in its systems.

3.1 The *Plug-in* Phase

This is an off-line and very preliminary phase, fundamentally related to companies' IT preparedness, which is one of the pre-conditions for their participation in VEs [1,10]. IT preparedness involves many aspects, as processes and IT governance, legacy systems integration, software maturity, security and supporting hardware [2,10].

This work assumes that all this is somehow and previously dealt with by the own companies when they are accepted to be members of a given VBE [1, 14]. This also means that companies should handle hiding their organizational and cultural heterogeneity when they decide to work in a VBE/VE and share common working principles [1]. Besides that, and regarding the adopted SOA approach, it is also assumed that their systems are duly wrapped and exposed as software services, implemented in different technologies (as *web services* and *REST*), in order to be further accessed by VBEs' and VEs' related client applications [10].

Therefore, the core goal of the plug-in phase is to make VBE members 'ready to participate' in VEs once they are created.

Still in this phase, as a step related to the semantic interoperability, each company's service is semantically annotated and mapped against the respective UBL BP's activities ontology. This also considers the different services granularities (to be considered when the systems are wrapped) and implementation technologies. Companies' systems can keep working using their proprietary BPs, terminologies and data models. However, if a legacy system uses a proprietary protocol or it is not exposed as a service, a wrapper has to be built up to enable it getting into the VBE. For example, if a given VE member deals with logistics (to send some product´s parts to another member) its logistic legacy system has to be wrapped as a service (with its interface) respecting the activities of the *shipping* UBL BP.

Wrapping legacy systems as services (or more modern ITs) and adopting given BP models to leverage higher interoperability is actually a common practice in companies when doing business. What the model proposes is a configurable, open and standard-based integration connector, a "plug", bridging the companies' internal models and the VBE e VE computing infrastructure, so preserving companies' heterogeneity. This also means that companies can use different "plugs" enabling them to make business with other networks.

Three steps have been devised to support this process, illustrated in Figure 1.

i. The construction of the UBL-based OWL ontology. This step is supported by the *Automatic Ontology Creation module*, which is used by the VBE IT administrator to feed it with the vocabulary's XML/XSD schemas and returning the equivalent OWL ontology. This ontology will be used in further steps to establish mappings with the messages exchanged by the VE members' systems and to resolve the mediation between different data models at runtime.

ii. Once the global vocabulary is defined the next step is taking every company's single service (previously wrapped), describing it semantically using OWL-S [15] and registering it in the VBE services repository. This action is performed by the IT manager of each company.

iii. The last step refers to use the created semantic descriptions of each service to extract information about the services' *payloads* and to further map it against the UBL Ontology. This is supported by the *semi-automatic mapping module*.

This topology, which combines one central ontology (the UBL one) and the many other ones (representing the members' services), considers the performance benefits in the mediation process as showed in [16].

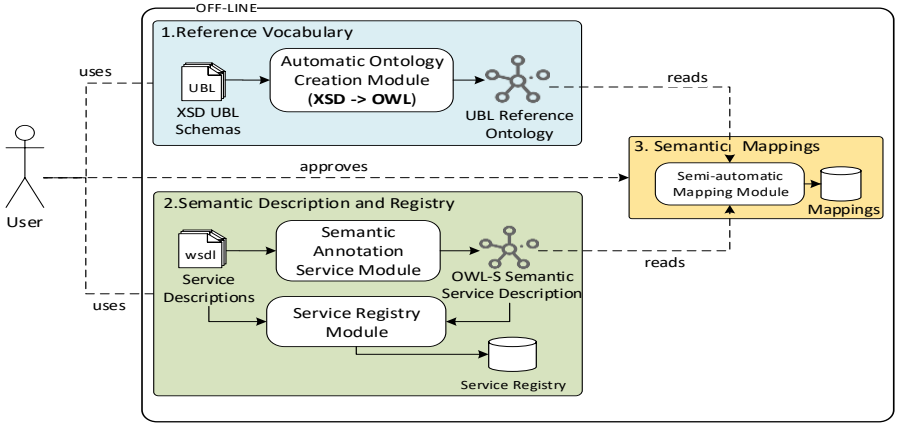

Fig. 1 The Plug-in phase steps

3.2 The *Play* Phase

It is an on-line phase. It is executed to support each company (its systems) to get dynamically, logically and temporarily integrated and interoperate within the VE computing environment once it is selected to be a VE member.

It is important to highlight that plenty of VEs can be created simultaneously and that a given company (i.e. its systems) can get into several VEs simultaneously too.

This phase is executed in three moments: 1) when the company is selected for a VE (*creation stage*); 2) when other companies are selected to get onboard to the VE due to extra activities not initially planned and then the VE should be recomposed (*evolution stage*); and 3) when a given VE member does not accomplish its duties and one or more new members should get onboard to replace it (*evolution stage*).

Three steps are also involved to support this process, as illustrated in Figure 2.

i. The information flow starts by running the given VE's BPs in the *BPEL Execution engine*, represented in the BPEL open standard format. This also contains the VE members (*Enterprises 1, 2 ...*) involved in each BP.
The model follows the classical BPM-SOA approach, being each BP modeled in BPMN (based on e.g. UBL) and further transformed into a BPEL file [5]. However, VE members' services are not invoked directly. Instead, BPEL invokes the ESB (see below), which makes the services invocation themselves.

ii. The model exposes a proxy (*integration endpoint*) in the ESB infrastructure, which receives all the invocations made by the *BPEL Execution engine.* Those invocations have information containing: the *payload* (information sent in proprietary format and data model of the sender member); the address of the source and the receiver services' interfaces; and the operation to be invoked in

the receiver party. There is only one ESB for the entire VBE. The ESB allows the dynamic, logical and "temporary" integration as well as the binding, communication and execution of companies' services associated to the several VEs' BPs while preserves the different implementing IT of companies' systems.

iii. Once the integration endpoint receives an incoming message from the *BPEL Execution engine,* it passes it to the *semantic mediator service*. It uses the sent interfaces' addresses to query the *mappings database* looking for the necessary mappings to transform the source payload into the receiver data model. Once the mappings are received the necessary protocol/format transformations of the message are managed by the *ESB*. The ESB uses the received interface address and operation name to invoke it, following the router integration pattern provided by the *ESB* so completing the communication between VE members while supporting syntactic and semantic interoperability.

The ESB setting up and configuration as well as the services instantiation, semantics treatment and services invocation for each VE are performed automatically, dynamically and transparently to the users and companies.

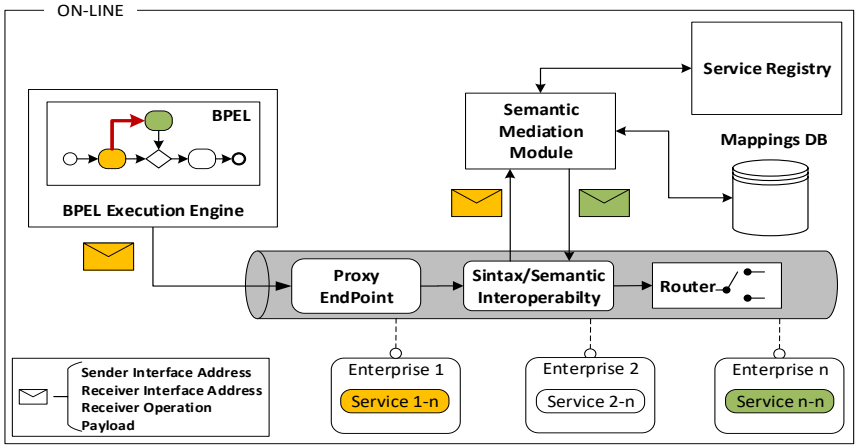

Fig. 2 The Play phase steps

3.3 The *Unplug* Phase

It is also an on-line phase. It happens in two situations: 1) when the VE member accomplishes its duties (during the VE operation or evolution stages) and naturally leaves the VE (in the VE dissolution stage); and 2) when a given VE member has some problems to accomplish its duties and should leave the VE after a sort of general analyses (during the VE operation stage).

The VE Coordinator should update the VE plan in the situation 2 according to the results of the new partners' selection process (step *i* of the *Play* phase). The VE's BPEL file is automatically reconfigured.

VE dissolution is a topic not much covered neither in practice nor in the literature. Several complex issues, at different types, are actually involved when a VE member leaves a VE, especially in the situation 2. Besides legal issues (also considering that a

VE should be kept operational until all the legal obligations have been fulfilled), it is necessary to handle logistics and other aspects related to the physical parts that were being produced. There are also some tough IT problems. For example, when a given member is producing its parts, its software services are also in execution and have some associated state memory. From the services technology point of view, replacing a VE member by one company (or even by more than one) would also mean to pass the same state memories to the new members' services, which is very complex.

None of these issues are currently supported by this presented model. However, besides the provided support for that BPEL reconfiguration and for that logical disconnection of the VE member's services, this model simplifies the problem, supporting a replacement of one member by only one, assuming that this replacement will imply that the involved services are no longer held to any process.

4 Implementation & Preliminary Assessment

This section presents the VE scenario and computing prototype that was implemented to qualitatively assess the proposed integration model.

All the model's artifacts were implemented using SOAP, web services and Java language, and deployed in a local network and controlled environment.

Related to VE members' services, plenty of services were implemented in a simple way also using those ITs to simulate the VBE members' services in this current version of the prototype. The use of web services is not a limitation at all as the model is able to handle multiples technologies thanks to the ESB capabilities.

In the *Plug-in* phase, a key issue that had to be faced was the one related to ontologies, the conversion of XSD/XML to RDF/OWL. There are many approaches for that [17] and the one developed in [18] has been chosen and implemented as it has fit best the requirements of the intended model.

A second implemented artifact was the *semantic annotation service*, which generates compliant OWL-S ontologies from a WSDL description. The work developed in [19] was modified and implemented as a service. This artifact generates the process model, the grounding and profile ontologies that store all the necessary information to get the service description, data model and implementation details.

The third artifact was the *Ontology Matcher Service.* It is responsible for automatically finding the semantic equivalence between services' terms (from the companies' data models) and the respective terms in the UBL ontology. There are also some approaches to tackle this, as in [20,21,22]. The *Agreement Maker Light* matcher [23] was chosen regarding the desired level of precision and execution time for the model. It was also modified and wrapped as a service.

The result of all this process is a set of mappings. Considering the complexity involved in matching problems [23] ontology mappings can have some imprecision. Given that the quality of this mapping is crucial for the proper functioning of the whole model, the mappings are not immediately set up for being used. Instead, each company's IT manager is assisted with an interactive GUI to make the final checking (and corrections) of the suggested mapping of each service against the UBL ontology.

In the *Play* phase, the given VE's BPs are modeled in BPMN and further converted to BPEL 2.0 following the WS-BPEL standard. The BPEL process is

deployed using the execution engine supported by *Apache ODE*[4] orchestrator. It invokes the integration endpoint, which is exposed as a SOAP service in the chosen message infrastructure (supported by *MULE ESB*[5]) and that is responsible for managing the messages exchange between VE members. The members' services are registered using *jUDDI*[6] service registry (that supports UDDI 3.0 standard), and the ontology mappings are stored in a MySQL database.

In order to evaluate the proposed model and the semantic mediation approach, a scenario has been devised and modeled in BPMN using UBL, as showed in Figure 3.

The scenario refers to a hypothetic customer who asks for a given product close to a given VBE's company. This product is basically composed of three parts, being one part produced by this company. This company then triggers the process of VE creation, ending up by forming the following VE: this company ('Partner 1') as the *VE Coordinator* and which interacts with the customer; and 'Partner 2' and 'Partner 3', which manufacture the other two product's parts. These two partners should send their parts directly to Partner 1 for the final assembly once they are finished.

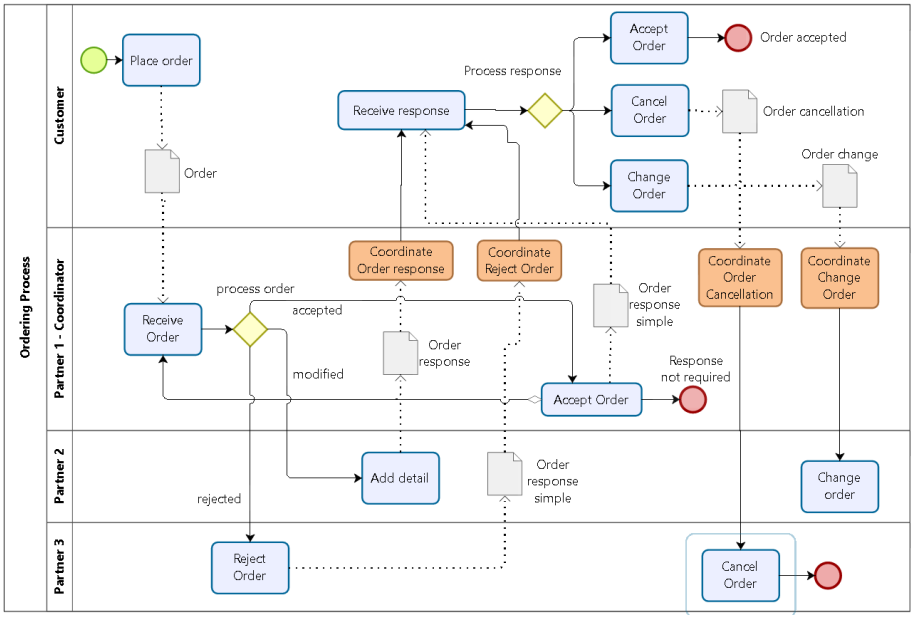

Fig. 3 Ordering process expressed in BPMN

This scenario is associated to the UBL process *'Ordering Process'* (one of the 68 BPs of the standard), which has a number of sub-processes and internal activities. A number of documents are also needed to be exchanged within this process and, in the case of a VE, different partners are responsible for executing some of the sub-processes. Due to space restrictions, Figure 3 only shows one part of the *Ordering*

[4] http://ode.apache.org/
[5] https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb
[6] https://juddi.apache.org/

Process, covering the acceptance and rejection of the business request. Many BPs will be normally involved in a 'real' VE regarding its full life cycle and the BP's needs of the customer order in place.

From the execution point of view, the way the model treats one VE and one UBL BP is exactly the same than as for many VEs and multiple BPs.

This scenario assumes that services are made available by the involved companies (the *plug-in* phase). Several different services were implemented using different data models so as to better test the mediation process in each BP transaction.

Figure 4 presents an excerpt of the results of this process. The example shows two mappings established between the ontology (representing the ordering SOAP service of *Partner 2* ('E2')) and the UBL reference ontology. It can be observed that the concept *orderRspns* present in the VE member's service (its internal local representation) was mapped against the *OrderResponse* concept present in the UBL reference ontology and *purchaseOrder* was mapped to *Order*.

The precision of the mappings has a measure from 0 to 1. The closer it is to 1 the higher is its precision, i.e. the closer the equivalence between the involved terms is. The implemented matcher has a configurable property for that, called *similarity threshold*. A threshold of 0.95 has been set up in this prototype, and this is also used to help the IT manager in that mentioned final checking of the suggested mappings.

A set of unit and integration tests were made to evaluate the functioning of every implemented module and of the model as a whole, i.e. if it supported the right VE operation, allowing VE members to seamlessly and automatically communicate with each other about BP-related transactions regarding syntactic and semantic differences.

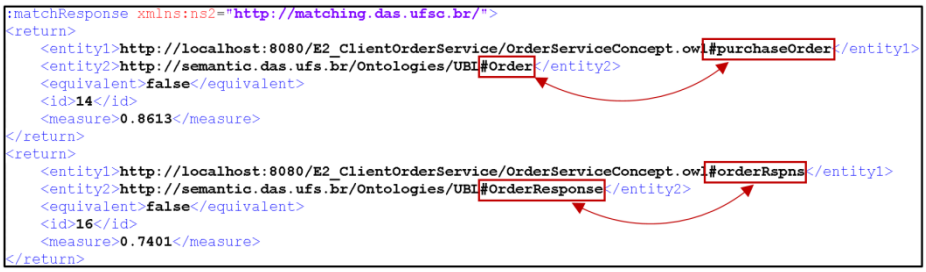

Fig. 4 Semantic alignments with the matching process

5 Conclusions

This paper has presented a contribution towards a more agile integration of companies to VEs and their systems' interoperation.

It is represented by a three-phase model that, respecting the VBE and VE life cycles, helps companies in their IT preparedness, their integration and operation in VEs, and their unplugging from them.

The model was devised based on state-of-the-art design principles, like being open, low intrusive and service-oriented. To be highlighted the strong use of IT and BP open standards as well as of integration patterns in all the involved actions.

Important to mention that, once companies' systems are made available (in the *plug-in* phase), all the VE operation is executed automatically (unless given BPs'

activities are designed to request human intervention) and transparently, dynamically (allowing companies to enter and to leave a VE and automatically respecting the VE's plan), using open and standard-based IT and BP model but preserving companies' autonomy and their IT heterogeneity. VBE members can also come and go but this is transparent for the plug & play model's execution.

The developed prototype could show the potential of the proposed approach and of the semantic mediation techniques used.

Although the whole model is open to handle multiple technologies thanks the use of an ESB, only W3C IT standards were used in this prototype. Yet, although UBL has been adopted as the internal BP reference for the basics of systems interoperability, the model is open to support any other BP model.

The model tries to contribute to mitigate interoperability problems at three levels [13]. At the *technical* (or syntactical) level, it handles the existence of different systems and services, using different data formats and data integration middleware (e.g. ESB), and their access from disparate systems. At the *semantic* level, the meaning of the exchanged data and messages are automatically recognized and further processed via reference data and ontologies. At the *organizational* level, there is a BP alignment and the automated processing of its workflows through the use of common services-based architecture (e.g. SOA). No concrete contributions are provided to the fourth and upper level, the *legal* interoperability.

Next main short-steps of this research are: i) supporting semantic description of REST services; ii) stressing the prototype in scenarios composed of plenty of simultaneous EVs, BPs and companies (i.e. services) in a distributed and less controlled computing environment; iii) the analysis and implementation of how security needs can be expressed as non-functional requirements when modeling BPs at the BPM/BPMN level and be automatically and properly interpreted and dealt with by the ESB; and iv) the integration of the implemented model with an almost finished module to support resilience at runtime of SOA-based VE systems.

References

1. Camarinha-Matos, L.M., Afsarmanesh, H.: Collaborative networks: a new scientific discipline. *Journal of Intelligent Manufacturing*, N 16 (4-5), pp. 439-452 (2005).
2. Rabelo, R. J., Gusmeroli, S.: The ECOLEAD collaborative business infrastructure for networked organizations. In *Methods and Tools for Collaborative Networked Organizations*, pp. 451-462. Springer (2008).
3. Romero, D., Vernadat, F.: Enterprise information systems state of the art: past, present and future trends. *Computer in Industry*. N 79, pp. 3-13 (2016).
4. Panetto, H., Jardim-Goncalves, R., Romero, D.: New perspectives for the future interoperable enterprise systems. *Computer in Industry*, N 79, 47-63 (2016).
5. Draheim, D.: Service-Oriented Architecture. In *A Unified View on Business Processes, Workflows and Enterprise Applications*, pp. 221–241. Springer (2010).
6. Schratzenstaller, W., Baldo, F., Rabelo, R.J.: Semantic Integration via Enterprise Service Bus in Virtual Organization Breeding Environments. In *Intelligent Information and Database Systems*, pp. 544–553. Springer (2016).
7. Zhu, W.: Semantic Mediation Bus: An Ontology-based Runtime Infrastructure for Service Interoperability. In *IEEE 16th International Enterprise Distributed Object Computing Conference Workshops*, pp. 140–145 (2012).

8. Shi, K., Gao, F., Xu, Q., Xu, G.: Integration framework with semantic aspect of heterogeneous system based on ontology and ESB. In *Proceedings 26th Chinese Control and Decision Conference*, pp. 4143–4148 (2014).
9. Khalfallah, M., Figay, N.: A cloud-based platform to ensure interoperability in aerospace industry, *Journal of Intelligent Manufacturing*, N 27, pp. 119–129 (2016).
10. Picard, W., Paszkiewicz, Z., Gabryszak, P., Krysztofiak, K., Cellary, W.: Breeding virtual organizations in a service-oriented architecture environment. In *SOA Infrastructure Tools - Concepts and Methods*, 375–396 (2010).
11. Dehbokry, S. G., Chew, E.: Toward a Multi-disciplinary Business Architecture Reference Model for SMEs. In *Proceedings 23th European Conference on Information Systems* (2015).
12. OASIS: Universal Business Language Version 2.2, http://docs.oasis-open.org/ubl/UBL-2.2.html, (2018).
13. EIF, European Interoperability Framework: Towards Interoperability for European Public Services, European Commission (2011).
14. Romero, D., Galeano, N., Molina, A.: A Virtual Breeding Environment Reference Model and Its Instantiation Methodology. In *Proceedings IFIP 9th Working Conference on Virtual Enterprises (PRO-VE'08)*, pp. 15–24. Springer (2008).
15. France Telecom, University of Maryland, NIST, Nokia, Stanford Univ., Toshiba: OWL-S: Semantic Markup for Web Services, in www.w3.org/Submission/OWL-S/.
16. Abadi, A., Ben-Azza, H., Sekkat, S.: An ontology-based framework for virtual enterprise integration and interoperability. In *International Conference on Electrical and Information Technologies (ICEIT'16)*, pp. 36–41 (2016).
17. Hacherouf, M., Bahloul, S.N., Cruz, C.: Transforming XML documents to OWL ontologies: A survey. *Journal of Information Science*, N 41, pp. 242–259 (2015).
18. Yüksel, M.: A Semantic Interoperability Framework for Reinforcing Post Market Safety Studies, Ph.D. Thesis, Middle East Technical University (2013).
19. Paolucci, M., Srinivasan, N., Sycara, K.: Towards a Semantic Choreography of Web Services: from WSDL to DAML-S. In *Proceedings International Conference on Web Services (ICWS' 2003)*, pp. 22-26 (2003).
20. Banouar, O., Raghay, S.: Comparative study of the systems of semantic integration of information: A survey. In *Proceedings 12th International Conference of Computer Systems and Applications*, pp. 1–8 (2015).
21. Liu, H., Cutting-Decelle, A.-F., Bourey, J.-P.: Use of Ontology for Solving Interoperability Problems between Enterprises. In *Proceedings 11th IFIP Working Conference on Virtual Enterprises (PRO-VE'10)*, pp. 730-737. Springer (2010).
22. Achichi, M., Cheatham, M., Dragisic, Z., Euzenat, J., Faria, D., Ferrara, A., Flouris, et al.: Results of the Ontology Alignment Evaluation Initiative 2017. In *Proceedings 16th International Semantic Web Conference*, pp. 61–113 (2017).
23. Faria, D., Pesquita, C., Santos, E., Palmonari, M.: The Agreement Maker Light Ontology Matching System. In *Proceedings OTM'13 Conference - On the Move to Meaningful Internet Systems*, Springer, pp. 527–541 (2013).