Juan David Méndez Astudillo

UM MODELO DE INTEROPERABILIDADE DE SISTEMAS PARA EMPRESAS VIRTUAIS – UMA ABORDAGEM BASEADA EM ORIENTAÇÃO A SERVIÇOS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas
Orientador: Prof. Dr. Ricardo José Rabelo
Coorientador: Prof. Dr. Fabiano Baldo

Florianópolis
2019

Ficha de identificação da obra elaborada pelo autor
através do Programa de Geração Automática da Biblioteca Universitária
da UFSC.

Méndez Astudillo, Juan David
Um Modelo de Interoperabilidade de Sistemas para Empresas Virtuais : Uma Abordagem Baseada em Orientação a Serviços / Juan David Méndez Astudillo ; orientador, Ricardo José Rabelo, coorientador, Fabiano Baldo, 2019.
176 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2019.

Inclui referências.

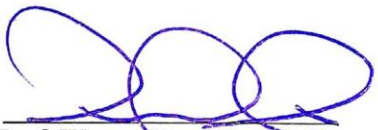
1. Engenharia de Automação e Sistemas. 2. Empresas Virtuais. 3. Integração de Sistemas. 4. Arquitetura Orientada a Serviços. 5. Interoperabilidade. I. Rabelo, Ricardo José. II. Baldo, Fabiano. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Juan David Méndez Astudillo

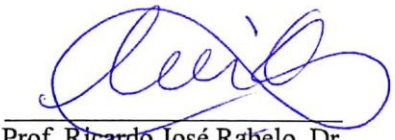
UM MODELO DE INTEROPERABILIDADE DE SISTEMAS PARA EMPRESAS VIRTUAIS – UMA ABORDAGEM BASEADA EM ORIENTAÇÃO A SERVIÇOS

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Engenharia de Automação e Sistemas" e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Florianópolis, 12 de Abril de 2019.



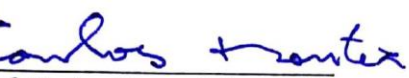
Prof. Werner Kraus Junior, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas



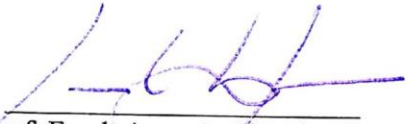
Prof. Ricardo José Rabelo, Dr.
Orientador
DAS / Universidade Federal de Santa Catarina

Prof. Fabiano Baldo, Dr.
Coorientador
Universidade Estadual de Joinville
(Videoconferência)

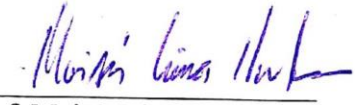
Banca Examinadora:



Prof. Carlos Barros Montez, Dr.
Universidade Federal de Santa Catarina



Prof. Frank Augusto Siqueira, Dr.
Universidade Federal de Santa Catarina



Prof. Moisés Lima Dutra, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado à minha querida família, meus pais Marta, Jesús e a minha irmã Isabela.

AGRADECIMENTOS

À minha família em geral, que sempre me deu o seu apoio e me incentivou para alcançar minhas metas. Especialmente aos meus pais, Marta e Jesús, pelo seu amor, ensinamentos e apoio incondicional ao longo da minha vida, sem eles eu não teria chegado onde estou hoje; à minha querida irmã Isabela, que é um dos meus motivos para me superar e ser uma melhor pessoa a cada dia.

À Andrea, que me acompanhou durante toda essa aventura, pelo seu apoio nos momentos difíceis, pelo seu carinho e compreensão, sem dúvida sua companhia foi uma grande ajuda para atingir o objetivo.

Ao meu orientador o Professor Doutor Ricardo Rabelo, pela confiança depositada em mim, pelas suas orientações e comprometimento durante o desenvolvimento do trabalho.

Ao meu coorientador o Professor Doutor Fabiano Baldo, pela ajuda e contribuições para obter um melhor resultado.

Aos colegas do mestrado e do laboratório (LTIC), que me acolheram e ajudaram a me adaptar em outro país, sempre dispostos para resolver minhas dúvidas, pela sua amizade e pelas experiências compartilhadas ao longo dessa jornada.

Aos professores e ao pessoal administrativo do PPGEAS, especialmente ao senhor Enio Snoeijer pela sua dedicação e disposição para ajudar a todos os alunos.

À Universidade Federal de Santa Catarina (UFSC) e o Governo do Brasil, pela oportunidade de me capacitar em uma instituição educativa de altíssimo nível, da mesma forma, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro proporcionado.

Tudo aquilo que o homem ignora, não existe para ele. Por isso o universo de cada um, se resume ao tamanho do seu saber.
(Albert Einstein)

RESUMO

A interoperabilidade de sistemas é uma questão importante a ser enfrentada ao suportar a execução de Empresas Virtuais (EVs). Mesmo sendo uma atividade fundamental para a criação e operação de EVs ainda é um tema que tem vários pontos em aberto para se explorar, isto em consequência de que essa integração e composição dinâmica de sistemas é um processo complexo que abrange diversos aspectos. Caraterísticas inerentes aos membros de uma EV como são a autonomia, distribuição geográfica, alta heterogeneidade dos modelos de dados e processos de negócios usados por eles fazem com que seja necessário aplicar alguns procedimentos para efetivamente suportar uma EV. Muitos trabalhos na literatura apontam que empresas que buscam entrar em um ambiente para criação de EVs deveriam ter alguma preparação, inclusive no nível de TI, a fim de criar uma solução viável para conseguir interoperar agilmente em EVs. Este trabalho pretende superar algumas desvantagens das abordagens atuais para a integração dos sistemas das diferentes empresas que participam de uma EV, propondo um modelo onde os parceiros da EV possam entrar mais facilmente quando a EV é criada (*plug*) e para que a EV seja executada satisfatoriamente durante sua operação (*play*) e dissolução (*unplug*), incluindo um nível de suporte à interoperabilidade semântica. Um protótipo foi implementado e os resultados são discutidos.

Palavras-chave: Empresas Virtuais, Integração de Sistemas, Arquitetura Orientada a Serviços, Ontologias, Interoperabilidade Semântica, Serviços Web.

ABSTRACT

Systems interoperability is an important issue to be faced when supporting the execution of Virtual Enterprises (VEs). Although it is a fundamental activity for the creation and operation of VEs, it is still a topic that has several open points to explore, because the complexity of integration and dynamic composition of systems. Characteristics inherent to VE members such as the autonomy, geographic distribution, high heterogeneity of the data models and business processes used by them, makes necessary to apply some procedures to effectively support a VE. Many approaches in the literature point out that companies that seek to enter into a VE environment should have some preparation, including at the IT level, in order to create a viable solution to interoperate smoothly with VEs. This work intends to overcome some of the disadvantages of the current approaches for the integration of the VE participants' systems, proposing a model where the VE partners can enter easily when the VE is created (plug), so that the VE is executed seamlessly during its operation (play) and dissolution (unplug), including some support for semantic interoperability. A prototype was implemented and the results are discussed.

Keywords: Virtual Enterprise, Systems Integration, Service Oriented Architecture, Ontologies, Semantic Interoperability, Web Services.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACV	Ambiente de Criação de organizações Virtuais
Apdex	Application Performance Index
API	Application Programming Interface
BP	Business Process
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Modeling and Notation
B2B	Business-to-Business
CRM	Customer Relationship Management
ebXML	Electronic Business using eXtensible Markup Language
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
EV	Empresa Virtual
FTP	File Transfer Protocol
HTML	HiperText Markup Language
IDL	Interface Description Language
IIOP	Interoperable Inter-ORB Protocol
JMS	Java Message Services
JSON	JavaScript Object Notation
OASIS	Organization for the Advancement of Structured Information Standards
OV	Organização Virtual
PJ	Pessoa Jurídica
PLN	Processamento de Linguagem Natural
PME	Pequenas e Médias Empresas
QoS	Quality of Service
RCO	Redes Colaborativas de Organizações
RDF	Resource Description Framework
REST	REpresentational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call

SA-WSDL	Semantic Annotation WSDL
SCM	Supply Chain Management
SLR	Systematic Literature Review
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOAP	Service Oriented Architecture Protocol
TIC	Tecnologias da Informação e Comunicação
UBL	Universal Business Language
UDDI	Universal Description Discovery & Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WS	Web Services
WSDL	Web Service Description Language
WSMO	Web Service Modeling Ontology
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas a velocidade das mudanças nas empresas tem aumentado de forma significativa, devido ao surgimento de demandas por produtos com maior qualidade e nível de personalização, menores custos, menores tempos de entrega, crescente número de regulamentações regionais, nacionais e internacionais, maior concorrência, maior necessidade de entrada em novos mercados, aumento de uso das tecnologias como e-commerce, entre vários outros aspectos. Como consequência, as empresas vêm trabalhando para achar os meios que as permitam melhor atuarem visando se manterem competitivas no mercado.

Pequenas e Médias Empresas (PMEs), que representam a grande maioria das empresas no Brasil e no Mundo, são as que mais têm dificuldades para operacionalizar essas mudanças dadas as suas usuais limitações de recursos financeiros, humanos, conhecimento e tecnologia (CAMARINHA-MATOS; AFSARMANESH, 2005). Para fazer frente a tais limitações, uma das estratégias emergentes mais proeminentes atualmente são as Redes Colaborativas de Organizações - RCO do inglês (*Collaborative Networked Organizations*).

RCOs se baseiam na ideia do compartilhamento de recursos e conhecimento entre grupos de empresas (essencialmente PMEs). Isto visa reduzir custos e riscos gerais de operação, aumentar a capacidade e escala de produção assim como o nível geral das suas competências uma vez que, individualmente, não conseguiriam realizar ou, se sim, com enormes dificuldades. Dada a essas características, as RCOs fornecem maiores meios das PMEs conseguirem competir com grandes empresas e evoluir com menores custos e riscos através do aprendizado coletivo, contínuo e compartilhamento de melhores práticas (CAMARINHA-MATOS; AFSARMANESH, 2005; RABELO et al., 2016; VERNADAT, 2010).

Por definição, os membros de uma RCO são empresas juridicamente independentes, autônomas, geograficamente distribuídas e heterogêneas em termos organizacionais, cultura, capital social e estratégia de negócios, mas que se unem visando atingirem objetivos comuns. Em termos gerais, a diferença das RCOs para outros tipos de estratégias empresariais é que, nestas, a colaboração é sistemática, atrelada ao processo normal de operação das empresas-membro, e não algo feito apenas de forma pontual, excepcional. Ainda, as interações entre seus membros em um dado negócio ou oportunidade de colaboração se dá preponderantemente via rede de computadores (como a Internet), integrada aos sistemas locais das empresas, e não por meios não digitais, como telefone (CAMARINHA-MATOS; AFSARMANESH, 2005).

RCO é um conceito amplo, tendo vários tipos de manifestações. Exemplos incluem ambientes de criação de organizações virtuais - ACV (*Virtual Organization Breeding Environment*), organizações virtuais (*Virtual Organizations*), empresas virtuais - EV (*Virtual Enterprises*), cadeias de suprimentos (*Supply Chain*), laboratórios colaborativos virtuais (*Collaborative Virtual Laboratories*), aglomerados industriais (*industrial clusters*), consórcio, cooperativas, entre vários outros (CAMARINHA-MATOS; AFSARMANESH, 2005).

Dois tipos de RCO são considerados nesta dissertação: ACV e, mais profundamente, EV. Um ACV é uma associação de longo prazo de PMEs que oferece de forma suficiente os meios necessários para que seus membros se preparem para trabalhar de forma colaborativa uns com os outros (CAMARINHA-MATOS; AFSARMANESH, 2005).

Há diversos tipos e níveis de colaboração entre membros dentro de um ACV; uma delas é na forma de EV. Em termos gerais, uma EV é tida como sendo uma aliança estratégica temporária e dinamicamente estabelecida de um grupo de empresas independentes e geograficamente distribuídas formada para atender “sob medida” uma oportunidade de negócio ou objetivos comuns de colaboração. A efetivação de uma EV é sempre regida por um compartilhamento coordenado de habilidades, conhecimentos, informação e recursos por meio de redes de computadores (CAMARINHA-MATOS; AFSARMANESH, 2005).

Uma EV pode ser constituída por empresas de variados perfis; por exemplo, empresas de manufatura, de serviços, operadores logísticos, entre outros. Cada EV define um modelo de governança que rege todas as atividades que dentro dela ocorrem, estabelecendo os direitos e deveres que cada membro tem. Normalmente uma RCO orientada a oportunidades, como é o caso da EV, tem um tempo de vida curto, se caracterizando por encerrar suas atividades assim que o objetivo para o qual foi criada é atingido, ou seja, assim que o produto/serviço final é feito e entregue ao cliente considerando-se todas as obrigações legais associadas ao negócio (CAMARINHA-MATOS; AFSARMANESH, 2005).

O modelo de RCOs e suas manifestações (como EVs) vem cada vez mais sendo reconhecida como a abordagem mais moderna em negócios colaborativos dadas sua maior flexibilidade e agilidade de integração em ecossistemas empresariais, e principalmente como um novo modelo de sustentabilidade para PMEs (ROMERO; VERNADAT, 2016).

O tema de EVs tem sido pesquisado em dezenas de projetos de pesquisa há pelo menos duas décadas, principalmente em função do seu

impacto junto a PMEs (RABELO et al., 2016; ROMERO; RABELO; MOLINA, 2013). Porém, ainda que as vantagens de concretizar colaborações entre empresas sejam grandes, o processo de materialização de EVs não é trivial, tendo-se em conta os inúmeros aspectos organizacionais, legais, técnicos, computacionais e culturais que devem ser tratados para uma colaboração fluída (AFSARMANESH; CAMARINHA-MATOS; ERMILOVA, 2008; ROMERO; GALEANO; MOLINA, 2008).

Como mencionado anteriormente, uma das principais características das EVs é o uso intensivo de redes de computadores ao realizar suas transações. Isto implica que, para efetivamente uma EV funcionar, todos os sistemas computacionais dos membros devem estar preparados para interoperar (CAMARINHA-MATOS; AFSARMANESH, 2005; RABELO et al., 2004). Esses sistemas interatuam segundo os fluxos de informação e controle associados aos processos de negócio das EVs nas quais participam. Isto se estende a situações em que a EV é modificada; por exemplo, pela saída de um membro da colaboração, (e.g., devido ao descumprimento das atividades assignadas e sua posterior substituição por um ou mais membros). Ainda, como uma empresa pode participar de várias EVs (i.e. de diferentes negócios com diferentes parceiros) simultaneamente, os seus sistemas computacionais deverão ser prontamente integrados a inúmeros diferentes sistemas, nem sempre previamente conhecidos.

Este problema é agravado na medida em que os sistemas computacionais e arquiteturas de suporte usadas pelas empresas-membro são em grande parte distribuídas e heterogêneas (em termos de TIs usadas, terminologias, sintaxes, assim como, tipo de processos, incluindo sistemas de chão de fábrica). Isto traz grandes desafios ao tentar suportar um ambiente flexível e escalável (ULLAH; LAI, 2013; ZHONG et al., 2017).

Em resumo, esse cenário significa em se suportar uma configuração, integração e interoperação (incluindo a semântica) fluída e dinâmica de inúmeros sistemas computacionais diferentes e largamente distribuídos em cada EV e suas empresas-membro bem como de toda infraestrutura geral de suporte à comunicação entre elas, considerando ainda que a composição de uma EV pode mudar ao longo da sua operação e, assim, a composição dos sistemas envolvidos (DEN HAMER; SKRAMSTAD, 2011; PANETTO et al., 2016). Sendo este o problema-foco desta dissertação.

Em termos gerais, integração de sistemas tem a ver com fazer com que os sistemas de uma empresa consigam se comunicar de forma correta

e coordenada de acordo com os fluxos de controle e dados requeridos. Já interoperação se refere a garantir que na integração dos sistemas estes possam trocar e usar dados, funcionalidades ou serviços uns dos outros sem perda ou distorção (ROMERO; VERNADAT, 2016).

Existem inúmeras estratégias e arquiteturas de integração (MYERSON, 2001). Para lidar com arquiteturas distribuídas, heterogêneas e escaláveis, SOA (*Service Oriented Architecture*) tem sido um estilo arquitetural bastante apropriado dado seu caráter desacoplado de integração e de independência tecnológica. Isto é em parte garantido pelo fato de que, em contrapartida ao modelo tradicional de sistemas monolíticos, uma arquitetura baseada em SOA é composta por um conjunto de módulos independentes, reusáveis e flexíveis, configurados para interagirem entre si dentro de uma lógica de regras de negócio. (JOSUTTIS, 2007; PAPAZOGLOU, 2012). Desta forma, ao se projetar uma arquitetura de integração baseada em SOA, benefícios como flexibilidade e escalabilidade são mais fáceis de serem suportados e de garantir um melhor alinhamento entre os objetivos de negócio e a arquitetura tecnológica disponível (BERNSTEIN; HAAS, 2008).

Nessa direção, esta dissertação visa explorar o processo de integração e interoperação dos sistemas de um conjunto de empresas pertencentes a um ACV (e assim das EVs que dele surgirem), considerando um ambiente colaborativo e distribuído. O pressuposto de base deste trabalho é que tais empresas tomaram como uma das decisões estratégicas em termos de "preparação de TI" (CAMARINHA-MATOS; AFSARMANESH, 2005) que todos seus sistemas computacionais são passíveis de serem modelados ou encapsulados como serviços sob uma visão SOA.

1.1 JUSTIFICATIVA

O cenário exposto impõe alguns aspectos difíceis de resolver em termos de integração de sistemas, como: o pertencimento dinâmico, temporário e simultâneo de empresas a diferentes EVs ao longo de seus ciclos de vida; o uso intensivo de redes de computação necessárias para apoiar a comunicação entre os membros da EV considerando a natureza colaborativa e compartilhada do trabalho; a distribuição geográfica e independência das empresas que adotam diferentes modelos de TI, processos de negócio (BP – *Business Process*), terminologias e métodos de trabalho; as diferentes regras de governança impostas a cada EV; os diferentes domínios de segurança a serem atravessados pelas transações

dos BPs; entre outros. Lidar com todos esses aspectos é de grande complexidade.

Variados trabalhos na literatura têm proposto *frameworks* de integração de sistemas em cenários de EV (ou de outros tipos de redes de empresas) em diferentes níveis de profundidade. Contudo, para diminuir a complexidade do problema, alguns pressupostos são normalmente assumidos, como (RABELO; GUSMEROLI, 2008; ROMERO; VERNADAT, 2016): as empresas que participam da EV estão "de alguma forma" já prontas para entrar na colaboração; os membros da EV adotam as mesmas tecnologias de integração, assim como os mesmos modelos e terminologias de processos de negócios; uma vez que a EV tenha sido criada, ela não altera sua composição de membros; não lidam com a fase de dissolução da EV, ou seja, com a saída de empresas (e a consequente "*desplugagem*" dos seus sistemas da arquitetura global da EV). Além disso, usam abordagens que acabam tornando a integração dos parceiros da EV um processo bastante invasivo em termos de mudanças nos sistemas das empresas e não tão ágil e fluído (ROMERO; VERNADAT, 2016). Esses são os principais aspectos onde este trabalho pretende contribuir com o modelo a ser proposto.

1.2 PERGUNTA DE PESQUISA

Considerando-se o problema geral de pesquisa explanado e o foco pretendido de contribuições, esta dissertação busca responder a seguinte pergunta de pesquisa:

> *Como melhorar a agilidade da integração e o processo de interoperação entre as empresas-membro de EVs, considerando o ciclo de vida das EVs?*

1.3 OBJETIVO GERAL

Para responder a pergunta de pesquisa, este trabalho propõe um modelo aberto de integração e interoperabilidade que permita que as empresas-membro de uma EV (ou seja, os seus sistemas de TI) possam entrar, participar na execução de processos de negócio e sair de EVs, sendo o menos invasivo possível com os sistemas das empresas participantes.

1.4 OBJETIVOS ESPECÍFICOS

Derivado do objetivo geral, os objetivos específicos são representados como partes a serem desenvolvidas ao longo da pesquisa visando-se construir e avaliar o modelo desejado como um todo.

Por outro lado, abstraindo-os e de certa forma os agrupando, pode-se dizer que quatro objetivos específicos refletem os principais resultados a serem gerados:

- Projetar o modelo de integração para Empresas Virtuais que suporte interoperabilidade sintática e semântica.
- Criar uma ontologia que represente um modelo de referência de processos de negócio como base para interoperabilidade semântica;
- Implementar um protótipo computacional como prova de conceito, que represente o modelo de integração desejado, sob uma ótica SOA e com uso de padrões abertos de TIs nos diversos níveis envolvidos.
- Avaliar o protótipo experimentalmente com base em indicadores de desempenho.

1.5 ADEQUAÇÃO ÀS LINHAS DE PESQUISA DO PROGRAMA

O trabalho aqui descrito se enquadra no contexto da área de Sistemas Computacionais, que é uma das linhas de concentração onde o Programa de Pós-Graduação em Engenharia de Automação e Sistemas realiza atividades de formação e pesquisa. Considerando a problemática exposta e a linha de concentração, este trabalho se adere ao tópico relacionado às Empresas Virtuais.

1.6 ORGANIZAÇÃO DO DOCUMENTO

Além da introdução e contextualização apresentada neste primeiro capítulo, o restante do trabalho está dividido em outros seis capítulos, descritos brevemente a seguir:

O capítulo 2 descreve o procedimento metodológico usado para abordar o tema de pesquisa.

O capítulo 3 sumariza a revisão dos conceitos fundamentais para o entendimento deste trabalho.

O capítulo 4 apresenta o estado da arte, condensando os principais trabalhos relacionados ao tema de pesquisa e é apresentada uma comparação dos trabalhos com a proposta desta dissertação.

O capítulo 5 introduz o projeto do modelo de interoperabilidade com todos seus módulos e interações entre eles.

O capítulo 6 mostra a implementação do protótipo computacional, bem como os resultados dos testes feitos nas diferentes etapas de execução do modelo.

Por fim, o capítulo 7 condensa o trabalho por meio das conclusões, contribuições, limitações e trabalhos futuros.

2 METODOLOGIA DE PESQUISA

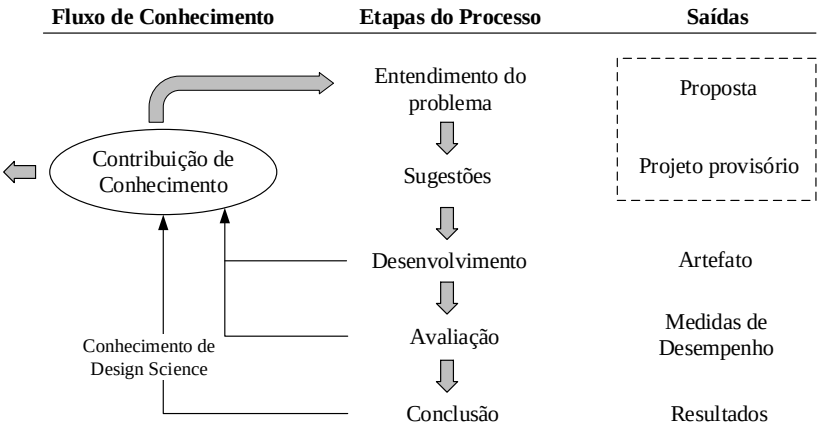
Este capítulo tem como objetivo apresentar o enquadramento metodológico usado neste trabalho, assim como as etapas e procedimentos definidos para atingir os objetivos estabelecidos.

2.1 MÉTODO DE PESQUISA

A pesquisa pode ser geralmente definida como uma atividade que contribui para a compreensão de um fenômeno. O fenômeno é tipicamente um conjunto de comportamentos de algumas entidades que são consideradas interessantes pelo pesquisador ou por um grupo de pesquisa. O conjunto de atividades que uma comunidade de pesquisa considera apropriadas para produzir conhecimento são seus métodos ou técnicas de pesquisa (VAISHNAVI; KUECHLER; PETTER, 2004).

O método de pesquisa adotado neste trabalho é *Design Science*, que consiste em tratar o problema de pesquisa com uma abordagem incremental e iterativa, por meio da criação de um artefato principal que permite obter resultados que são aprimorados a cada iteração. A Figura 1 apresenta o ciclo básico proposto no método *Design Science*.

Figura 1. Visão geral do método Design Science



Fonte: Adaptada de (VAISHNAVI; KUECHLER; PETTER, 2004)

Toda pesquisa inicia por uma fase de identificação/entendimento do problema (parte superior central). Como resultado desse processo é obtida a proposta inicial do trabalho, a qual é refinada segundo as sugestões feitas pelos colaboradores na pesquisa.

Assim que o projeto provisório do trabalho é criado, a fase de desenvolvimento do artefato central da pesquisa começa. Quando o artefato atingir um ponto estável, ele é avaliado considerando medidas de desempenho apropriadas ao problema tratado.

Segundo os resultados obtidos na etapa de avaliação podem acontecer duas coisas: o artefato realmente representa na prática o exposto na literatura e, além disso, é evidenciada uma solução/contribuição ao problema em questão, ou o resultado de avaliação do artefato não representa a teoria como era esperado. Na primeira situação, os primeiros resultados da pesquisa são obtidos gerando conhecimento; na segunda opção, é realizada uma nova iteração com base nos resultados obtidos até o momento, ajustando o entendimento do problema e a proposta do projeto para assim começar de novo o processo de pesquisa.

2.2 ENQUADRAMENTO METODOLÓGICO

O enquadramento metodológico é um instrumento importante que permite planejar a execução da pesquisa proporcionando os meios para definir as condições dentro das quais os experimentos de pesquisa serão realizados e delimitar o escopo do trabalho (GIL, 2010). Essas condições são:

- **Premissa de pesquisa:** Convalida e Afirmativa, de que a carência de um modelo de interoperabilidade ou as limitações dos atuais afetam diretamente o desempenho e sucesso das empresas-membro de uma EV nos negócios em que participam;
- **Método de Pesquisa:** *Design Science* (JÄRVINEN, 2007). Pretende-se encontrar uma solução a um problema concreto e específico intervindo diretamente na concepção de um modelo voltado para facilitar que as empresas de uma EV se integrem;
- **Natureza da pesquisa:** Aplicada, busca-se aproveitar as fundamentações teóricas e tecnologias existentes ao redor da temática de integração e interoperabilidade no cenário de EVs. Aplicando-as com o intuito de projetar e implementar o modelo de interoperabilidade desejado;

- **Abordagem do problema:** Quali-Quantitativa, já que o modelo que pretende-se desenvolver visa contribuir em aspectos quali e quantitativos do processo geral de integração e interoperabilidade dos membros das EVs, avaliando o resultado através de experimentos e análises parcialmente subjetivas;
- **Tipo de pesquisa:** Dedutivo; busca-se obter conclusões por meio de informações colhidas de experimentos orientados a representar a criação, operação, evolução e dissolução de EVs e seus processos de integração ao longo dessas etapas;
- **Paradigma de pesquisa:** Adota-se o paradigma Positivista, partindo da premissa que o problema de integração de sistemas em um cenário de EVs pode-se observar e medir por meio de dados específicos e que o objeto de pesquisa e o pesquisador encontram-se separados; parte-se do conhecimento prévio na área e tenta-se propor soluções para as lacunas identificadas na formulação do problema usando como base para provar que o modelo de interoperabilidade melhora a integração em um cenário de EVs.
- **Objetivo de pesquisa:** Ação, visa agir para projetar um modelo de interoperabilidade de sistemas em EVs sem pretender desvendar o fenômeno de integração de sistemas;
- **Tempo de pesquisa:** Estudo Longitudinal; devido à caraterística incremental do método *Design Science*, variados experimentos são executados ao longo do desenvolvimento do trabalho conforme os avanços obtidos em cada iteração.
- **Observação:** Sistemática, são realizadas observações no decorrer das diferentes etapas de desenvolvimento do modelo de interoperabilidade.
- **Procedimentos:** Variados, foram realizados diversos procedimentos ao longo das iterações e etapas do *Design Science*. Pesquisas bibliográficas, visando, através da leitura de artigos científicos, teses e dissertações obter uma base de conhecimento teórico que sirva de fundamentação para o modelo que se pretende propor. Além disso, foram realizados testes experimentais com o intuito avaliar tecnologias e ferramentas para assim identificar o estado da prática e utilizar as tecnologias mais adequadas na implementação do protótipo e os experimentos.

2.3 ROTEIRO BÁSICO DE PESQUISA

Considerando-se a natureza e os objetivos desse trabalho de pesquisa, foram definidas 9 etapas detalhadas a seguir. Em termos gerais a metodologia de desenvolvimento é sequencial, com algumas etapas podendo ser realizadas em paralelo.

1. **Revisão bibliográfica:** A abordagem adotada para efetuar a primeira parte da pesquisa foi o método SLR - *Systematic Literature Review* (KITCHENHAM; CHARTERS, 2007). O SLR é uma metodologia que permite se pesquisar por trabalhos relevantes na área e realizar a análise do estado da arte. Com isso é possível ao pesquisador identificar as lacunas na área e a contribuição científica pretendida. O SLR realizado é apresentado no capítulo 4;

2. **Projeto do modelo:** Baseando-se na revisão bibliográfica do estado da arte e identificação de requisitos funcionais e não funcionais mais usuais em um cenário de EV bem como o tipo de trabalho a ser realizado e seu enquadramento metodológico, foi projetado o modelo de suporte à interoperabilidade para EVs. O modelo é apresentado no capítulo 5;

3. **Seleção do modelo de referência de processos de negócio:** Levantamento de modelos de referência dentre os vários disponíveis, tanto na pesquisa quanto na indústria, para analisar seus escopos e focos setoriais. O modelo escolhido é utilizado na prova de conceito desenvolvida do modelo de interoperação;

4. **Seleção de tecnologias e ferramentas:** Via revisão bibliográfica e das práticas de integração empresarial, são analisadas as tecnologias, ferramentas e plataformas necessárias para suportar o modelo de interoperabilidade idealizado, com foco no uso daquelas que são abertas e usam padrões abertos;

5. **Implementação do Modelo:** implementação de um protótipo computacional considerando as análises efetuadas nas etapas 2 à 5 e, em um segundo momento, da etapa 7, dado o ciclo iterativo do método Design Science adotado na metodologia;

6. **Construção de cenários de testes:** Construção de cenários de testes para verificação e avaliação do modelo de interoperabilidade no protótipo desenvolvido frente aos objetivos e requisitos delineados;

7. **Experimentação e documentação:** São documentados os experimentos realizados sobre os cenários de teste construídos na etapa anterior, obtendo os resultados necessários para uma análise posterior;

8. **Análise final:** Baseado nas informações adquiridas é projetada uma análise apresentando as vantagens e desvantagens/restrições do modelo proposto;

9. **Escrita de publicações:** Por fim, os resultados consolidados das etapas anteriores são apresentados através de publicações. No caso, de um artigo em um congresso internacional da área de EVs.

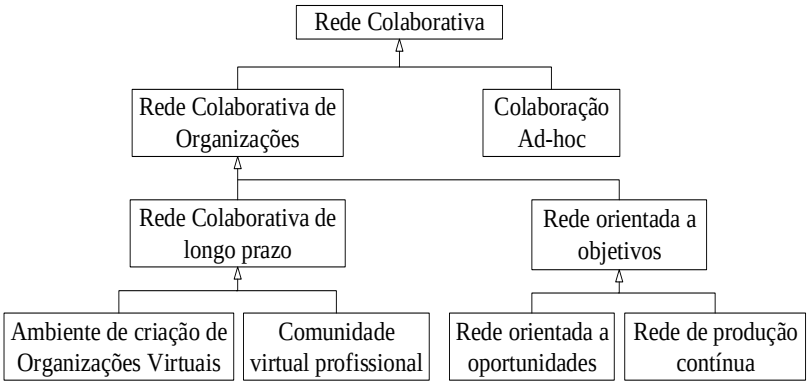
3 FUNDAMENTAÇÃO TEÓRICA

Como exposto anteriormente, o foco deste trabalho é conceber um modelo aberto de integração que suporte interoperabilidade sintática e semântica em ambientes de empresas virtuais ao longo do seu ciclo de vida numa abordagem geral baseada em SOA. Neste sentido, este capítulo sumariza as principais fundamentações teóricas de base que foram estudadas e utilizadas para a concepção do modelo proposto e que servem de base para o entendimento do trabalho.

3.1 REDES COLABORATIVAS DE ORGANIZAÇÕES

Uma RCO é definida como uma rede de variadas entidades (empresas, pessoas, instituições governamentais, etc.), geograficamente distribuídas, autônomas e heterogêneas em termos de seus ambientes de operação, cultura, objetivos e capital social, que usam a colaboração sistemática entre elas como estratégia para suplantar suas limitações individuais e atingir objetivos comuns, e cujas transações são fundamentalmente realizadas via redes de computadores (CAMARINHA-MATOS; AFSARMANESH, 2005; RABELO et al., 2004). Segundo o apresentado em Camarinha-Matos e Afsarmanesh (2008a) e como pode-se observar na Figura 2 é possível diferenciar dois tipos principais de RCOs: redes estratégicas de longo prazo e redes orientadas por objetivos.

Figura 2. Taxonomia de Redes Colaborativas



Fonte: Adaptada de (CAMARINHA-MATOS; AFSARMANESH, 2008a)

As *redes estratégicas de longo prazo* são parcerias estabelecidas com o intuito de oferecer as condições e ambientes de suporte necessários aos membros da rede para prepará-los e facilitar sua participação em futuras redes colaborativas que serão criadas quando surgirem novas oportunidades de negócio. Essa categoria abrange outro tipos de RCOs, como:

(i) *Ambiente de criação de organizações virtuais* (*ACV*): é considerado uma associação de empresas que acordam em estabelecer uma cooperação de longo prazo com vistas a melhorar sua preparação para atender oportunidades de negócio de forma colaborativa. Este conceito é abordado com maior profundidade na seção 3.1.1.

(ii) *Comunidade virtual profissional*: é tida como uma união de profissionais que visam se preparar para eventuais colaborações comerciais sob uma perspectiva orientada a negócios, e prover um ambiente que facilite a formação de *times virtuais* que respondam às oportunidades que surjam. Esse tipo de RCO oferece um suporte similar ao fornecido pelo ACV às EVs.

Já as *redes orientadas a objetivos* são um tipo de rede colaborativa onde seus membros interagem continuamente tendo como finalidade atingir um ou vários objetivos comuns. Esse tipo de rede tem duas subdivisões:

(i) *Redes de produção contínua*: são redes orientadas a atividades de produção ou fornecimento de serviços contínuos. São caracterizadas principalmente por manter a estabilidade na estrutura da rede com uma definição clara dos membros e papéis que desempenham nas atividades da colaboração. Pelo fato de serem estáveis, são estabelecidas como uma colaboração de longo prazo.

(ii) *Redes orientadas a atender oportunidades*: são criadas única e exclusivamente para atender uma oportunidade de negócio específica; portanto, mais dinâmicas e podem ser de bem curto prazo, se caracterizando porque são dissolvidas uma vez que o objetivo inicial seja atingido.

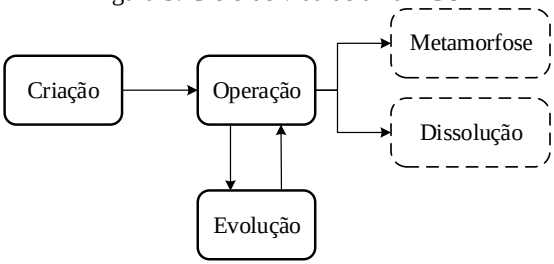
Considerando a classificação temporal das empresas (longo e curto prazo), é possível identificar duas vertentes em relação a como são gerenciadas suas atividades desde sua criação até sua operação. Por exemplo, analisando uma empresa típica que trabalha de forma isolada, de estrutura estável, planejada ao longo prazo, que teve sucesso e conseguiu se manter no mercado, a comparação de tempo de trabalho gasto na etapa de operação a respeito da criação/configuração ou dissolução é muito maior. Portanto, neste tipo de empresa a análise do

ciclo de vida e as etapas e atividades que o compõem é um aspecto menos crítico (CAMARINHA-MATOS; AFSARMANESH, 2007).

Já as *redes orientadas a atender oportunidades*, pelas suas características e dinamicidade, precisam operar rapidamente. Em consequência, é necessário estabelecer claramente as etapas do ciclo de vida e atividades executadas em cada uma delas para assim facilitar a transição de uma etapa para outra e atingir os objetivos definidos na criação da RCO; e por fim, uma vez todo o trabalho tenha sido concluído, a colaboração possa ser dissolvida efetivamente (CAMARINHA-MATOS; AFSARMANESH, 2007).

Segundo Camarinha-Matos e Afsarmanesh (2008b), um ciclo de vida genérico para RCOs é composto de 5 etapas principais (ver Figura 3), detalhadas na sequência:

Figura 3: Ciclo de vida de uma RCO



Fonte: Adaptada de (CAMARINHA-MATOS; AFSARMANESH, 2008b)

- **Criação:** nesta etapa devem ser executadas algumas tarefas relacionadas a seleção de membros e configurações iniciais, tais como a definição de infraestrutura de suporte, o compartilhamento de informação, os direitos de acesso, entre outras.

- **Operação:** é nesta etapa onde a RCO efetivamente realiza as atividades para atingir seus objetivos. Por exemplo, no caso da RCO ser do tipo EV, as atividades estão relacionadas principalmente à execução dos seus processos de negócio para desenvolver seus produtos ou serviços de forma colaborativa entre seus membros.

- **Evolução:** Etapa onde são executadas pequenas adaptações necessárias para o funcionamento da colaboração que surgem

durante sua operação, inclui aspectos como mudar os papéis dos membros da RCO, mudar a estrutura de seus processos de negócio, adicionar ou restringir permissões de acesso à informação, etc.

- **Metamorfose:** Implica uma mudança profunda na colaboração que não pode ser tratada na etapa de evolução devido à complexidade que envolve. Por exemplo, a transferência de conhecimento ou ativos a um terceiro. Esta etapa está relacionada com RCOs de longo prazo, já que considerando o grande valor e esforço dedicado ao estabelecimento dos recursos de suporte compartilhados ao longo da criação e operação, dissolver a colaboração é pouco provável.

- **Dissolução:** É a etapa onde a RCO encerra suas atividades e finaliza a colaboração.

Como apresentado na introdução do documento, este trabalho lidará essencialmente com RCOs dos tipos ACV e EV.

3.1.1 Ambiente de criação de empresas virtuais (ACV)

Um ACV representa uma associação de organizações e suas instituições de suporte relacionadas, que aderem a um acordo de cooperação de *longo prazo*. Com isso, adotam um conjunto de princípios operacionais e de infraestrutura em comum com o propósito principal de suportar e preparar seus participantes na criação de EVs (AFSARMANESH; CAMARINHA-MATOS; ERMILOVA, 2008).

Sua localização geográfica pode ser regional, nacional ou mesmo internacional, e ainda pode ser mono ou multi-setorial. Além disso, um ACV pode ser representado como uma Pessoa Jurídica (PJ) ou um agrupamento lógico de organizações, cada uma delas sendo uma PJ.

Cada ACV pode ter seu próprio modelo de governança, código de conduta, regimento interno, critérios de entrada/aceitação e retirada de empresas, convenções, padrões de gestão de qualidade, reputação, etc.

O processo de concretização de um ACV é composto de um conjunto de tarefas que devem ser executadas ordenadamente. Metodologias e *guidelines* de instanciação de ACVs têm sido propostas, como no trabalho de Baldo e Rabelo (2010) e no trabalho de Romero, Galeano e Molina (2008) que detalham as atividades principais

necessárias e o fluxo de execução delas ao longo do ciclo de vida das RCOs, por exemplo, planejamento estratégico, configuração de tecnologias de informação e comunicação, seleção de membros, criação de empresas virtuais, dissolução de ativos compartilhados, entre outras.

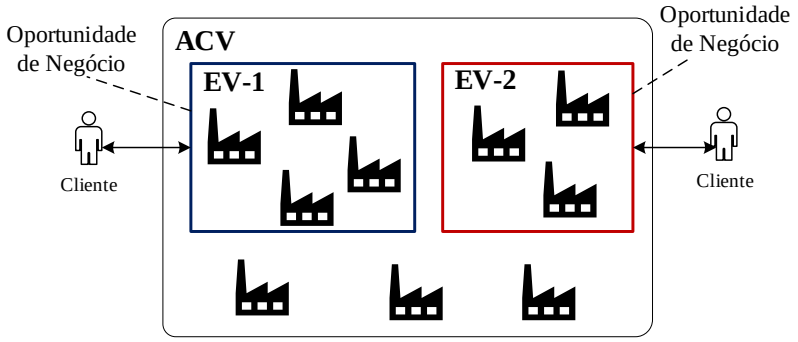
3.1.2 Empresa Virtual (EV)

Segundo (CAMARINHA-MATOS; AFSARMANESH, 2005) uma empresa virtual é uma aliança temporária e dinâmica de empresas legalmente independentes que compartilham recursos e habilidades para atingir um objetivo comum e que suportam as suas interações em redes de computadores.

Segundo o exposto, toda EV se origina/emerge a partir de um ACV. Apesar de existirem na literatura outros papéis, basicamente uma empresa é chamada de Membro (ou parceiro) quando apenas executa suas tarefas dentre as várias de uma EV; ou de coordenadora, quando ela gerencia e/ou é responsável legalmente pela EV (CAMARINHA-MATOS; AFSARMANESH, 2005). Todavia, uma empresa coordenadora também pode ser um membro da EV, tendo a possibilidade de participar em múltiplas EVs (ou seja, de negócios) simultaneamente.

Na Figura 4 pode-se observar os conceitos definidos anteriormente. Nela são formadas duas empresas virtuais (EV1 e EV2) no ambiente ACV com o intuito de responder a oportunidades de negócio diferentes.

Figura 4: Ambiente de criação de empresas (ACV) e Empresas Virtuais (EV)



Fonte: Própria.

3.2 MODELO DE REFERÊNCIA DE PROCESSOS DE NEGÓCIO

Um processo de negócio é um conjunto de atividades coordenadas que ocorrem com o intuito de atingir um objetivo. Pode ser a fabricação de um produto ou o oferecimento de um serviço. Geralmente, esses processos são empresariais, de gestão ou produtivos, e buscam atender os objetivos estratégicos da empresa ou de múltiplas empresas envolvidas (JOSUTTIS, 2007).

A importância dos processos de negócio no nível empresarial se evidencia no sentido de que é possível tornar uma organização mais eficiente melhorando seus processos. Para isso, deve ser realizada uma análise que permita conhecê-los detalhadamente, além de descobrir processos não explícitos ou de maior complexidade que envolvam diversas áreas da empresa ou até mesmo outras empresas (DUFRESNE; MARTIN, 2003).

Um processo de negócio pode ser expresso/documentado através de diversas linguagens ou formalismos, desde os tradicionais fluxogramas até notações mais atuais como BPMN (*Business Process Modeling and Notation*), que gradualmente está se tornando um "padrão". Existem diversas ferramentas de modelagem e gestão de processos de negócio (BPM – *Business Process Management*) no mercado. Um dos grandes avanços trazidos pelos modernos sistemas de BPM é a possibilidade de um processo, uma vez modelado em BPMN, poder posteriormente ser convertido em uma linguagem / código computacionalmente executável, por exemplo, a linguagem/padrão BPEL – *Business Process Execution Language* (DE SOUZA; RABELO, 2015; JOSUTTIS, 2007).

Modelos de Referência de processos de negócio têm surgido ao longo do tempo. Isso tem visado ajudar a melhor compreender uma organização como um todo, facilitar a comunicação e integração entre as suas áreas, a promover a criação de artefatos de TI que representam esses processos de negócio minimizando problemas de integração / interoperabilidade entre sistemas computacionais, e como uma alternativa para gerir de uma melhor forma a quantidade de processos de negócio que as empresas utilizam no seu funcionamento cotidiano.

Os modelos de referência de processos de negócio são modelos de informação que foram construídos para representar de forma completa a estrutura e os processos de um domínio de interesse. Esses modelos contêm conhecimento adquirido em anos de trabalho e experiência em projetos, condensando as melhores práticas de solução aos problemas recorrentes. Com isso, tornam-se genéricos o suficiente para ter uma

aplicação universal no domínio de interesse, favorecendo a reutilização e modificação segundo as necessidades de cada problema (FETTKE; LOOS, 2006).

Os modelos de referência podem ser vistos como modelos semanticamente generalizados, comumente usados como ponto de partida para o desenvolvimento de sistemas de informação empresariais que envolvem processos de negócio transversais à grande maioria das empresas como: compra, venda, faturamento, distribuição, etc.

Dentre os modelos de referência de processos de negócios empresariais podem-se citar:

3.2.1 ebXML

O projeto ebXML (*Electronic Business using eXtensible Markup Language*) foi iniciado em 1999 como uma iniciativa da OASIS (*Organization for the Advancement of Structured Information Standards*) e da agência das Nações Unidas / CEPE CEFACT. Ele fornece um conjunto modular de especificações que permite que empresas de qualquer tamanho e em qualquer localização geográfica conduzam negócios pela Internet. O principal ponto forte do ebXML é o de oferecer um método padrão para trocar mensagens de negócios, realizar relacionamentos comerciais, comunicar dados com terminologias comuns e definir e registrar processos de negócios (NAUJOK; HUEMER, 2008; OASIS, 2001). Foram projetadas cinco camadas de especificação de dados, incluindo padrões XML para:

1. Especificação de esquemas de Processos de negócios (*Business Process Specification Schema* - BPSS)
2. Componentes de dados principais (*Core Components* - CC)
3. Acordos de protocolo de colaboração e perfis (*Collaboration Protocol Profiles and Agreements* - CPP/A);
4. Mensagens (*Message Service Specification* - ebMS);
5. Registros e repositórios (*Registry Information Model* ebRIM / *Registry Services* ebRS)

Depois da finalização do conjunto de padrões, eles foram submetidos para aprovação ISO e aprovados como os padrões ISO 15000.

3.2.2 RosettaNet

O RosettaNet é um consórcio dirigido por membros da indústria que visa criar, implementar e promover padrões abertos de integração B2B. Os membros são principalmente dos setores de TI, componentes eletrônicos e fabricação de semicondutores.

A especificação define processos interempresariais onde ocorrem interações comerciais comuns e os documentos associados. É baseado no paradigma de Arquitetura Orientada a Serviços (SOA) e todos os documentos de negócios são expressos em XML (KOTINURMI; HALLER; OREN, 2011). O padrão é orientado às três seguintes áreas:

- *RossetaNet Business Dictionary*: Possui as propriedades necessárias para descrever as características de produtos e serviços.
- *RosettaNet Implementation Framework (RNIF):* Especifica o conteúdo das mensagens, os protocolos de transporte e mecanismos de segurança.
- *RosettaNet PIPs (Partner Interface Processes):* Especifica os processos interorganizacionais e a sequência em que as mensagens serão trocadas pelos participantes.

3.2.3 EDI/EDIFACT

O Intercâmbio Eletrônico de Dados (EDI) é usado para troca de informações em transações de negócios eletrônicos entre organizações colaboradoras sem intervenção humana. Um pré-requisito para a implementação bem sucedida de sistemas EDI são formatos padronizados que são tipicamente definidos para um domínio ou setor específico através de Organizações de Desenvolvimento Padrão (SDOs do inglês *Standard Development Organizations*), por exemplo, *Accredited Standards Committee* (ASC) X12 ou o Centro das Nações Unidas para Facilitação de Comércio e Negócios Eletrônicos (UN / CEFACT). Este último tornou-se um dos mais famosos na indústria por definir e manter o formato *United Nations/Electronic Data Interchange for Administration, Commerce and Transport* (UN/EDIFACT) (ENGEL et al., 2012).

EDIFACT é um padrão internacional aprovado pela ISO em 1987. Este padrão oferece um conjunto de regras de sintaxe (gramática) para estruturar dados baseado em um conjunto de caracteres acordado e um

vocabulário (*Data Elements*), um protocolo de troca de dados interativa (I-EDI) e um conjunto de mensagens padronizadas que permitem troca de informação entre empresas de múltiplos países e indústrias (BECKER, 1990).

3.2.4 UBL

UBL (*Universal Business Language*) é uma especificação gerenciada pela OASIS que define uma biblioteca aberta de documentos empresariais padronizados, baseados em XML, que suportam a digitalização dos processos comerciais e logísticos para cadeias de suprimentos nacionais e internacionais. Dentre os processos suportados estão a compra, venda, transporte, logística e outras funções de gestão da cadeia de suprimentos (OASIS, 2018).

UBL permite que as aplicações de negócios e as comunidades de comércio heterogêneas troquem informações ao longo de seus processos de negócio por meio de um formato de intercâmbio XML genérico. Ela define uma série de documentos de negócios que podem ser restritos ou estendidos para atender aos requisitos de indústrias específicas, além de detalhar as interações dos participantes em cada processo e os documentos que são trocados.

A última versão da especificação UBL é a 2.2, liberada em 2018. Essa versão estende os processos generalizados da cadeia de suprimentos da UBL 2.1, incluindo novos termos nos documentos, modificações nas cardinalidades de alguns dos objetos de negócio e alterações em processos de negócio. Entretanto, ao mesmo tempo ela mantém a compatibilidade com as versões predecessoras 2.0 e 2.1.

Das opções de modelos de referência de processo de negócio citadas acima, o modelo da especificação UBL é o que está mais atualizado, dado que a organização que o suporta vem trabalhando constantemente no seu aprimoramento, seguindo os avanços da indústria. Além disso, a especificação é gratuita, o que facilita sua adoção.

Por outro lado, segundo Oliveira (2011), as especificações ebXML e RosettaNet não estão sendo mais atualizadas, sendo que a última atualização feita nelas foi no ano 2001.

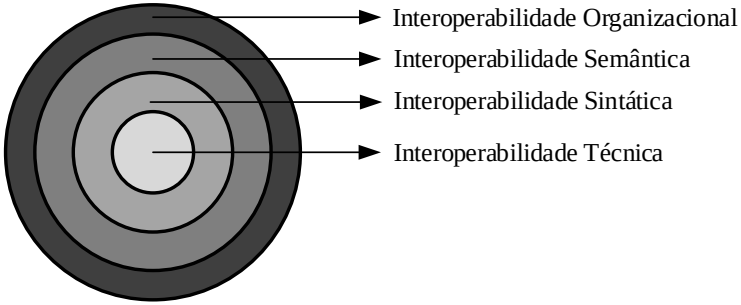
3.3 INTEROPERABILIDADE

Interoperabilidade é um conceito amplo e complexo que tem sido estudado ao longo do tempo em diferentes campos do conhecimento. Como é evidenciado em Rezaei et al. (2014) têm sido propostas diversas

definições deste termo. Neste trabalho é adotada a definição de interoperabilidade de Blair et al. (2011), onde é definida como a *habilidade que tem dois ou mais sistemas de trocar dados e serviços sem importar suas linguagens, implementações, entornos de execução ou modelos de abstração*.

Como apresentando em Rezaei et al. (2014), a literatura adota predominantemente 4 camadas de interoperabilidade: técnica, sintática, semântica e organizacional.

Figura 5: Camadas da interoperabilidade



Fonte: Adaptada de (DER VEER; WILES, 2006)

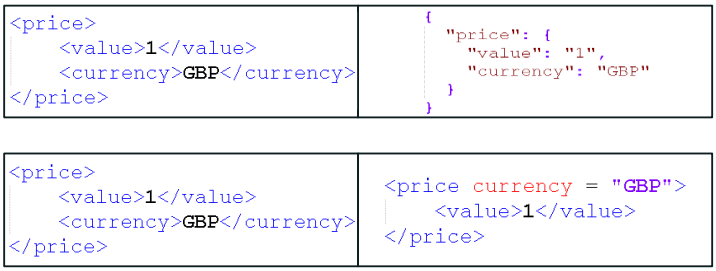
Na Figura 5, a camada interna representa a interoperabilidade técnica, que é geralmente associada a componentes, sistemas e plataformas de hardware/software que permitem a comunicação entre sistemas computacionais. É comumente centralizada em protocolos de comunicação e na infraestrutura necessária para que esses protocolos operem.

Na segunda camada encontra-se a interoperabilidade sintática, que agrupa os conflitos gerados pelas diferenças de esquemas nas diversas fontes de dados, como: tipo de dados, agregação, generalização e representação. É geralmente associada a formatos de dados como XML, HTML, JSON, etc.

Na Figura 6 podem-se observar dois exemplos de problemas de interoperabilidade sintática entre dois sistemas. A Figura 6.a mostra a divergência no formato usado para representar dados iguais, onde a informação mesmo tendo estrutura equivalente é representada em dois formatos de dados diferentes; do lado esquerdo XML e do direito JSON.

A Figura 6.b apresenta o problema de agregação da mesma informação, que mesmo sendo equivalentes, são organizadas em estruturas diferentes. Do lado esquerdo o nó *currency* está aninhado no nó pai, mas do lado direito essa informação é um atributo do nó pai. Portanto, para esses sistemas conseguirem se entender e interoperar deve existir no meio uma "transformação" que coloque os dados no formato e na estrutura esperados.

Figura 6: Problemas de interoperabilidade sintática



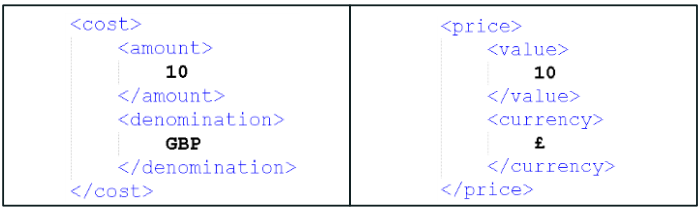
Fonte: Adaptada de (BLAIR et al., 2011)

A interoperabilidade semântica pode ser definida como a capacidade de operar em dados/mensagens trocados por sistemas de informações heterogêneos. O principal objetivo é garantir que dois sistemas de comunicação interpretem informações comuns ou compartilhadas de maneira consistente (VERNADAT, 2010), (ROMERO; VERNADAT, 2016).

Problemas de interoperabilidade semântica são comumente relacionados com diferenças de nomes, escalas, unidades, sinônimos, etc. A Figura 7 apresenta um caso típico de heterogeneidade semântica entre dois sistemas que usam a mesma estrutura e formato de dados para representar o custo de um item.

Embora os dois sistemas usem a mesma sintaxe e estrutura (XML), não é possível garantir que possam interoperar corretamente, já que não conhecem a equivalência de termos (eg. *price* ≡ *cost*, *value* ≡ *amount*, *currency* ≡ *denomination*). Portanto, mesmo que o sistema interprete o XML não vai conseguir entendê-lo devido à diferença dos termos para representar um mesmo dado (BLAIR et al., 2011).

Figura 7: Problema de interoperabilidade semântica



Fonte: Adaptada de (BLAIR et al., 2011)

Por fim, na última camada, está a interoperabilidade organizacional. Essa camada tem como foco o alinhamento de processos de negócios para aumentar a capacidade das organizações comunicar e transferir dados efetivamente, ainda que possam estar usando sistemas de informação diferentes sobre infraestruturas heterogêneas. Esse alinhamento de processos tem como resultado o agrupamento de subfunções em um único fluxo de trabalho automatizado e interorganizacional que usa o suporte fornecido pelas camadas inferiores de interoperabilidade mencionadas anteriormente (ROMERO; VERNADAT, 2016).

Um aspecto importante em relação à interoperabilidade entre sistemas é ter uma forma de medir a qualidade da mesma. Para isto têm sido propostas diferentes abordagens. O trabalho de Da Silva Serapião Leal, Guédria e Panetto (2019) apresenta um resumo das principais técnicas adotadas para tal fim.

Uma das técnicas que têm sido amplamente usadas para obter informações relacionadas ao *custo*, *duração* e *qualidade* da interoperação é a avaliação de desempenho. Essa técnica define um conjunto de métricas que permitem verificar esses três aspectos (DACLIN; CHEN; VALLESPIR, 2016):

- **Custo de interoperação:** Representa o custo envolvido pelos parceiros para realizar um ciclo de interoperação. A avaliação do desempenho da interoperação, em termos de custo, corresponde à comparação do custo real da troca de informação com o custo de operação que as empresas desejam empregar. Se um desses custos reais for maior do que os custos esperados, então há uma deficiência.

- **Duração da interoperação:** Corresponde à duração entre o *timestamp* de envio da informação e o *timestamp* em que a resposta é efetivamente utilizável pelo sistema solicitante, sendo desejável obter tempos curtos o suficiente para o problema de comunicação em questão.

- **Qualidade da interoperação:** Considera três tipos de qualidade: a qualidade da troca de informação, a qualidade da operação e a conformidade. A *qualidade de troca de informação* evidencia se a troca de informação é executada corretamente, ou seja, se as informações enviadas a um sistema foram bem-sucedidas. A *qualidade da operação* representa o número de recepções de um sistema em comparação com o número de requisições. Por exemplo, uma quantidade maior de recepções (dificuldade para processar toda a informação) ou menor quantidade (falta de informação) para o número de requisições significa uma deficiência. Finalmente, a *conformidade* corresponde à operação da informação, isto é, se de fato a informação recebida é diretamente utilizável ou não.

Considerando as camadas de interoperabilidade expostas anteriormente e recentes avaliações relacionadas à interoperabilidade (DA SILVA SERAPIÃO LEAL; GUÉDRIA; PANETTO, 2019; PANETTO et al., 2016), os atuais esforços de pesquisa consideram a interoperabilidade semântica como um dos problemas de maior relevância. Segundo Panetto et al. (2016), com a aplicação de interoperabilidade semântica se espera diminuir o número de pré-condições/pré-acordos necessários para que dois sistemas heterogêneos consigam interoperar.

Na prática, ainda existe uma lacuna em soluções a esse problema. Uma das abordagens é automatizar o alinhamento de conceitos semanticamente equivalentes, já que comumente esta atividade é feita de forma manual, consumindo muito tempo das atividades de integração, sendo susceptível a erros e em consequência, aumentando o custo dos projetos.

Na busca de uma solução para o problema de interoperabilidade semântica, muitas abordagens têm sido propostas; porém, existe uma tendência na comunidade científica de focar os esforços numa linha especifica: as ontologias (VERNADAT, 2010). Esse conceito é detalhado na seção a seguir.

3.4 ONTOLOGIAS

O conceito de ontologia tem sido usado ao longo do tempo em diferentes campos do conhecimento, como filosofia, linguística e ciências da computação. De forma geral, uma ontologia é usada para capturar conhecimento sobre uma porção de realidade (domínio de interesse) e descrever os conceitos e suas inter-relações no domínio representado.

Segundo Abadi et al (2016), Guarino, Oberle, Staab (2009) e Noy, McGuinness (2001), na área da computação uma ontologia é definida como um artefato de engenharia que permite representar o conhecimento por meio de um vocabulário compartilhado. Fornece uma classificação formal e explícita dos conceitos pertencentes a um domínio, especificando assim as propriedades e atributos de cada um desses conceitos, e detalhando as restrições usadas para criar relações que delimitem suas interpretações no domínio pretendido, evitando ambiguidades no significado.

Como é apresentado em Abadi et al, (2016), os benefícios esperados ao usar ontologias para resolver problemas de interoperabilidade podem ser classificados em três categorias:

- Integração e completude, proporcionadas pela expressividade da linguagem.
- Inteligência embarcada, devido à capacidade de raciocínio (*reasoning*) de linguagens de descrição lógica.
- Dinamismo e flexibilidade por meio de *queries* e *web services*.

A literatura tem adotado principalmente as arquiteturas de ontologias descritas nas próximas subseções para abordar problemas de integração (ABADI et al., 2016; WACHE et al., 2001):

3.4.1 Arquitetura de ontologias para integração

3.4.1.1 Ontologia única

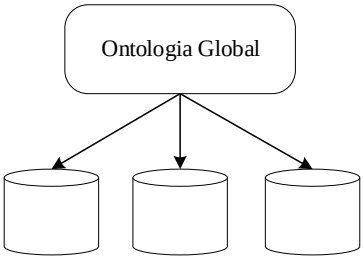
Usam uma ontologia global, fornecendo um vocabulário compartilhado para a especificação da semântica. Nessa abordagem, todas as fontes de informação estão relacionadas com a única ontologia global. Logo, elas podem ser aplicadas a problemas de integração onde todas as fontes de informação a serem integradas proveem quase a mesma visão em um domínio. Entretanto, se uma fonte de informação tem uma

visão diferente em um domínio, por exemplo, proporcionando um outro nível de granularidade, a integração se torna uma tarefa difícil. Além disso, as abordagens das ontologias individuais são suscetíveis a alterações nas fontes de dados, o que pode afetar a conceituação do domínio representado na ontologia.

A ontologia global pode ser uma combinação de várias ontologias especializadas (módulos). Uma razão para a combinação de várias ontologias pode ser a modularização de uma ontologia muito grande.

A combinação é suportada por formalismos de representação de ontologias, ou seja, a importação dos módulos da ontologia necessários. Na Figura 8 é apresentado o esquema básico dessa abordagem.

Figura 8: Abordagem de ontologia única



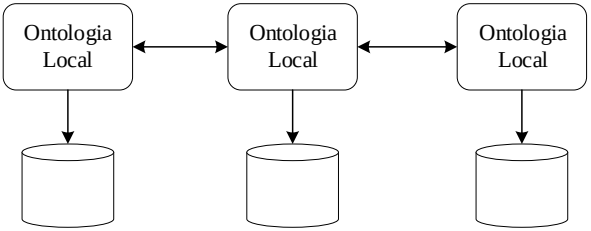
Fonte: Adaptada de (WACHE et al., 2001)

3.4.1.2 Múltiplas Ontologias

Na Figura 9 é possível observar o esquema dessa abordagem, onde cada fonte de informação é descrita por sua própria ontologia. Ela apresenta a vantagem de que cada ontologia das fontes pode ser desenvolvida independentemente, sem considerar as outras fontes ou ontologias, o que simplifica a integração e apoia a mudança, facilitando adicionar ou remover fontes de dados dinamicamente. No entanto, a falta de um vocabulário comum torna difícil comparar as ontologias de fontes diferentes. Como alternativa para superar esse problema é necessário um formalismo adicional de mapeamento entre ontologias, que serve para identificar termos correspondentes (iguais ou semelhantes) semanticamente em ontologias diferentes. A tarefa de mapeamento deve considerar aspectos conceituais da ontologia, como granularidade,

agregação de dados, etc., o que traz uma maior complexidade na integração.

Figura 9: Abordagem de ontologias múltiplas

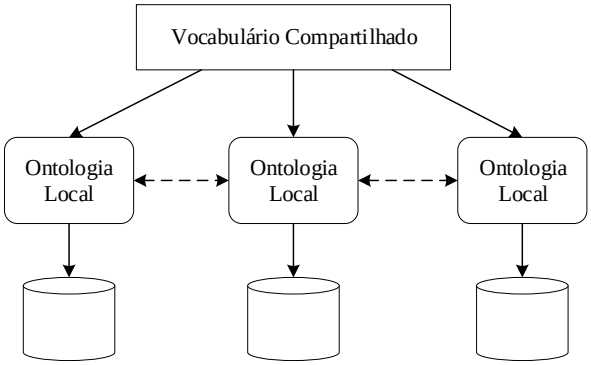


Fonte: Adaptada de (WACHE et al., 2001)

3.4.1.3 Abordagem híbrida

Desenvolvida para superar as desvantagens das abordagens de ontologia única e múltipla. Nessa abordagem, a semântica de cada fonte é descrita por sua própria ontologia. Mas, como pode ser observado na Figura 10 para tornar as ontologias locais comparáveis entre si, elas são construídas a partir de um vocabulário global compartilhado que contém os termos básicos do domínio, os quais são misturados nas ontologias locais, procurando descrever semânticas mais complexas.

Figura 10: Abordagem de ontologia híbrida



Fonte: Adaptada de (WACHE et al., 2001)

Essa estrutura e a utilização do vocabulário global na construção das ontologias locais evita as desvantagens de abordagens de ontologias múltiplas e oferece a vantagem da flexibilidade. A desvantagem dessa abordagem é que ontologias existentes não são facilmente reusadas, portanto, no caso de uma mudança, é necessário editar ou desenvolver uma nova a ontologia.

3.4.2 Criação e alinhamento de ontologias

Dado o ambiente heterogêneo e dinâmico das diversas fontes de dados em projetos de integração, duas das tarefas mais comuns são: a criação de ontologias para representar esses dados, e seu posterior alinhamento para conseguir integrá-los. A seguir são descritas brevemente essas tarefas.

3.4.2.1 Técnicas de criação de ontologias

Considerando a dinamicidade do ambiente tecnológico atual, onde a cada instante estão sendo gerados uma quantidade enorme de dados que tem que ser integrados em diferentes sistemas, como é o caso das RCOs, é necessário que esses sistemas sejam capazes de interpretar esses dados de forma ágil.

Levando em consideração que as ontologias são um meio que permite efetivar essa interpretação de dados, isso tem despertado o interesse da indústria e da comunidade científica para desenvolver métodos que acelerem (automatizem) o processo de criação delas.

Essa atividade é uma tarefa que tradicionalmente é executada com intervenção de um especialista no domínio, que nem sempre é fácil de encontrar.

Segundo Bedini e Nguyen (2007), o problema de criação de ontologias, ou como chamado na literatura *Ontology Learning*, pode ser classificado em 4 grandes categorias, descritas a seguir:

- **Conversão ou tradução:** Assume que a estrutura de uma ontologia já está bem definida por alguém ou em algum lugar, pode ser um esquema XSD, um diagrama UML, etc. Essa abordagem aproveita a definição da estrutura e estabelece um conjunto de regras de transformação, do formato específico para ontologia. Essa abordagem se caracteriza por ter um alto grau de automação, portanto agiliza o processo de criação.

- **Baseadas em mineração de dados:** Essas abordagens aplicam técnicas de mineração de dados com o intuito de recuperar informações suficientes para gerar uma ontologia. A maioria das abordagens se concentra em fontes de dados não estruturadas, como documentos de texto, páginas web, etc., e implementam técnicas de Processamento de Linguagem Natural (PLN). Dada a falta de uma estrutura bem definida, em contraste com a categoria de conversão, essa técnica de recuperação de conceitos estruturados a partir de documentos não estruturados se caracteriza por ter um alto grau de dificuldade, portanto, requer assistência humana.

- **Baseadas em conhecimento externo:** Corresponde às abordagens que criam ou enriquecem uma ontologia de domínio usando um recurso externo. Esta categoria pode por vezes se sobrepor à baseada em mineração de dados porque as técnicas aplicadas para recuperar informações podem ser as mesmas. No entanto, nesta categoria são usadas técnicas com uma abordagem mais próxima da integração de dicionários externos, ontologias existentes ou de recursos de conhecimento mais geral, como thesauri.

- **Baseada em Frameworks:** Essa categoria abrange os aplicativos que suportam o processo de criação, edição e visualização de ontologias, por exemplo, o bastante difundido *Protégé*[1].

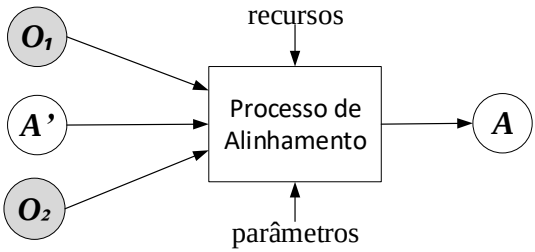
3.4.2.2 Técnicas de alinhamento de ontologias

Quando múltiplas ontologias que representam domínios iguais ou complementares, mas de fontes de dados heterogêneas, estão envolvidas em um mesmo projeto de integração, é necessário estabelecer as equivalências entre seus termos para poder efetivamente trocar informação. Esse é um dos principais problemas a ser resolvido pela interoperabilidade semântica. Esse processo de encontrar as correspondências entre entidades de ontologias semanticamente relacionadas é comumente conhecido como “alinhamento de ontologias” (do inglês *ontology matching*).

[1] https://protege.stanford.edu/

Sendo O_1 e O_2 um par de ontologias, o processo de alinhamento entre elas pode ser definido formalmente como um processo em que para cada elemento de O_1 busca se encontrar uma ou mais correspondências ou "mapeamentos" com elementos de O_2. Esse conjunto de correspondências pode ser denotado como A (SHVAIKO; EUZENAT, 2013). Além de ter as duas ontologias como entrada do processo de alinhamento, podem ser adicionados outros artefatos, como um alinhamento A' já feito que pode ser estendido, os parâmetros de alinhamento (como pesos ou limiares) e recursos externos, como uma base de conhecimento comum ou um thesauri, ver Figura 11.

Figura 11. Alinhamento de ontologias



Fonte: Adaptada de (EUZENAT; SHVAIKO, 2013)

Segundo Shvaiko e Euzenat (2013), uma correspondência entre 2 elementos, presente em um alinhamento A, pode ser definida como uma 4-upla da seguinte forma $< id, e1, e2, r >$, onde:

- id é o identificador da correspondência;
- $e1$ e $e2$ são entidades (classes, propriedades, etc.) na primeira e segunda ontologia respectivamente;
- r é a relação entre as duas entidades. Relações armazenam metadados. Normalmente uma medida usada é o grau de confiança na correspondência (intervalo de [0-1]). Quanto maior for a confiança, maior é a probabilidade que a relação seja correta.

Para executar o processo de alinhamento entre ontologias tem sido propostas diferentes técnicas, que segundo Euzenat e Shvaiko (2013) podem ser classificadas em 9 categorias:

- **Formal Baseada em recursos:** Usa recursos formais para suportar o processo de correspondência, como ontologias de nível superior, ontologias específicas de domínio ou alinhamentos de ontologias realizados previamente (reutilização de alinhamento).

- **Informal Baseada em recursos:** Como as técnicas da categoria anterior, também exploram um recurso externo; mas neste caso os recursos externos são informais. Esse conjunto de técnicas deduz as correspondências entre ontologias utilizando a relação que existe entre as ontologias e esses recursos informais.

- **Baseada em String:** Baseadas na similaridade das *strings*, que representam os nomes e descrições das entidades nas ontologias. Existem várias métricas de distância entre *strings* que podem ser usadas nesses métodos, por exemplo, Levenshtein, Jaccard, Jaro-Winkler, Euclidiano, TFIDF, etc.

- **Baseada na linguagem:** Baseadas na teoria do Processamento de Linguagem Natural (NLP – *Natural Language Processing*), elas não consideram os nomes das entidades simplesmente como strings, mas sim palavras em alguma linguagem natural. As técnicas nesta categoria são, por exemplo, *tokenization*, *lemmatization* ou *stopword elimination*. Essa categoria também considera as técnicas que aproveitam recursos externos para encontrar correspondências entre os termos, usando, por exemplo, lexicons como WordNet[2], dicionários ou thesauri.

- **Baseado em restrições:** Consideram critérios relacionados à estrutura interna das entidades das ontologias, como o domínio (*domain*) e o intervalo (*range*) das propriedades ou os tipos de atributos, para calcular a similaridade entre eles. É comum usar essas técnicas em conjunto com outras.

[2] https://wordnet.princeton.edu/

- **Baseado em grafos:** Consideram as ontologias como grafos marcados (*labelled graphs*) ou mesmo árvores, e tratam o problema de correspondência de ontologias como um problema de homomorfismo de grafos.

- **Baseado em taxonomia:** Também são algoritmos de grafos que consideram apenas a relação especialização. A ideia por trás das técnicas taxonômicas é que se existem links que conectam termos que já são semelhantes (sendo um subconjunto ou superconjunto do outro), possivelmente, seus vizinhos também possam ser de alguma forma semelhantes.

- **Baseado em instâncias:** Essas técnicas usam a extensão das classes nas ontologias, ou seja, os indivíduos, com a intuição de que se os indivíduos forem semelhantes, as classes às que pertencem também devem ser semelhantes. Essas técnicas podem usar princípios de teoria de conjuntos, mas também técnicas estatísticas mais elaboradas.

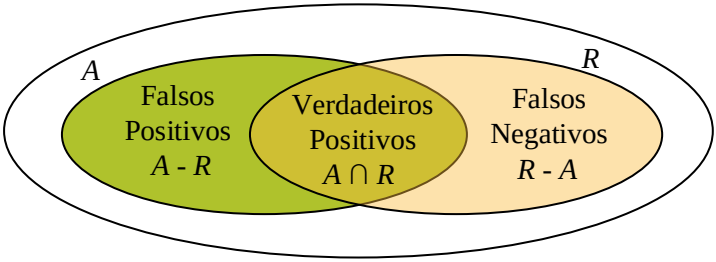
- **Baseado em modelo:** essas técnicas usam a interpretação semântica vinculada às ontologias de entrada.

Uma vez que as correspondências entre termos são realizadas e o *matcher* retorna o alinhamento *A*, é importante ter os meios para verificar se esse alinhamento é correto.

Na literatura, têm sido usadas para medir a qualidade do alinhamento de ontologias algumas métricas adaptadas principalmente da área de recuperação de informação (*information retrieval*). Dentre as medidas adotadas as principais são a distância de *Hamming*, Precisão (*Precision*), Cobertura (*Recall*), *F-measure* e *Overall* (DO; RAHM, 2002; EUZENAT; SHVAIKO, 2013).

A avaliação do desempenho no processo de alinhamento de duas ontologias é normalmente realizada contrastando o alinhamento *A* retornado pelo *matcher* contra um alinhamento de referência *R* previamente definido. Esses alinhamentos (conjuntos de correspondências) são expostos no diagrama de Venn da Figura 12.

Figura 12. Alinhamentos como conjuntos de correspondências e suas relações



Fonte: Adaptada de (EUZENAT; SHVAIKO, 2013)

Na Figura 12 podem-se observar as relações que existem entre dois alinhamentos *A* e *R*. Todos os elementos que fazem parte da intersecção entre *A* e *R* representam as correspondências do alinhamento *A* que foram corretamente geradas, dado que elas também estão no conjunto de correspondências no alinhamento de referência *R*, esses elementos são conhecidos como os *verdadeiros positivos*. As correspondências geradas no alinhamento *A* que não estão no conjunto de correspondências no alinhamento de referência *R* (correspondências falsas) são chamados de *Falsos Positivos*. Da mesma forma, as correspondências que estão no alinhamento de referência, mas que não foram encontradas pelo processo de alinhamento (correspondências perdidas), são chamadas de *Falsos Negativos*.

Baseado nos conjuntos apresentados na Figura 12, a seguir são detalhadas as principais métricas de confiabilidade de um alinhamento.

- **Distância Hamming:** Essa distância mede quão diferentes são dois alinhamentos considerando o número de correspondências corretas (*verdadeiros positivos*) sobre todas as correspondências presentes nos alinhamentos *A* e *R*. Se a distância calculada é zero indica que o alinhamento gerado *A* é igual ao alinhamento de referência *R*, portanto é correto. Caso contrário, se a distância tende a 1 sugere que a qualidade do alinhamento não é boa. A distância é descrita pela equação (1) apresentada a seguir:

$$H(A,R) = 1 - \frac{|A \cap R|}{|A \cup R|} \qquad (1)$$

- **Precisão:** A precisão mede a proporção de correspondências encontradas corretamente (*verdadeiros positivos*) sobre o número total de correspondências retornadas (*verdadeiros positivos e falsos positivos*). Em outras palavras, fornece uma estimativa da confiabilidade das correspondências preditas pelo *matcher*. Esta medida pode ser calculada aplicando a equação (2).

$$P(A,R) = \frac{|A \cap R|}{|A|} \qquad (2)$$

- **Recall:** Apresentada na equação (3), *Recall* mede a proporção de correspondências encontradas corretamente (*verdadeiros positivos*) sobre o número total de correspondências esperadas (*verdadeiros positivos e falsos negativos*). Ou seja, mede o grau de completude/perfeição do alinhamento.

$$R_C(A,R) = \frac{|A \cap R|}{|R|} \qquad (3)$$

- **F-measure:** Embora *precisão* e *recall* sejam das medidas mais usadas para avaliar alinhamentos, existem ocasiões onde essas medidas separadas podem não ser as mais apropriadas. Nesse sentido, tem sido propostas algumas medidas alternativas que combinam as duas gerando um resultado centralizado, entre elas, *F-measure*, apresentada a seguir na equação (4).

$$F_{\propto}(A,R) = \frac{P(A,R) \times R_C(A,R)}{(1-\propto) \times P(A,R) + \propto \times R_C(A,R)} \qquad (4)$$

O parâmetro $\propto$ controla a importância que tem a *precisão* e o *recall* no cálculo da medida *F-measure*. Se $\propto = 1$ então *F-measure* é igual à *precisão*; se $\propto = 0$, *F-measure* é igual ao *recall*, portanto, para ponderar as duas medidas na literatura é comumente escolhido um $\propto = 0.5$, que dá um peso equivalente às duas medidas, tendo assim a equação (5) apresentada na sequência.

$$F_{0.5}(A,R) = \frac{2 \times P(A,R) \times R_C(A,R)}{P(A,R) + R_C(A,R)} \qquad (5)$$

- **Overall:** Essa medida representa a proporção do número de erros sobre o total de correspondências esperadas (número de elementos em *R*). Pode ser considerada como uma distância de edição entre um alinhamento gerado (*A*) e um alinhamento de referência (*R*). Em outras palavras, essa medida indica o esforço necessário para corrigir o alinhamento *A* (remover correspondências falsas e adicionar as faltantes). *Overall* é sempre menor que a *F-measure* e varia entre [−1, 1]. Se a *Precisão* for inferior a 0,5, *Overall* retorna um valor negativo indicando que a reparação do alinhamento não vale o esforço (quanto mais alto, menor o esforço para reparar o alinhamento). Essa medida pode ser expressa como na equação (6) ou na equação (7).

$$O(A,R) = R_C(A,R) \times \left(2 - \frac{1}{P(A,R)}\right) \quad (6)$$

$$O(A,R) = 1 - \frac{|R - A| + |A - R|}{|R|} \quad (7)$$

3.5 ARQUITETURA ORIENTADA A SERVIÇOS (SOA)

A primeira definição da arquitetura orientada a serviços, SOA, do inglês *Service Oriented Architecture*, foi proposta no ano de 1996 por Roy Schulte e Yeffim V. Natiz, que a definiram como: “um estilo de computação multicamada que ajuda as organizações a compartilhar lógica e dados por meio de múltiplas aplicações e modos de uso” (apud ROTEM-GAL-OZ, 2012).

Atualmente, SOA é considerado um estilo arquitetônico que visa construir sistemas distribuídos baseados em componentes chamados de *serviços*. Esses serviços se caracterizam por serem autônomos e fracamente acoplados, expondo suas funcionalidades e as informações necessárias para se comunicar por meio de contratos que são usados por componentes externos (consumidores do serviço) para estabelecer a comunicação (PAPAZOGLOU, 2012; ROTEM-GAL-OZ, 2012).

Este paradigma surgiu como uma alternativa para superar os problemas da integração fortemente acoplada das aplicações tradicionalmente monolíticas, oferecendo mecanismos de flexibilidade, interoperabilidade e baixo acoplamento que permitem integrar e compor

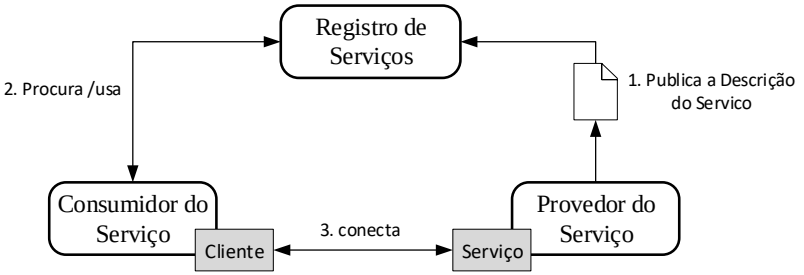
dinamicamente diferentes tecnologias independentemente da plataforma tecnológica subjacente. SOA promove a reutilização e reduz o tempo de implementação e acesso a novas funcionalidades do sistema (manutenção), permitindo dinamicamente publicar, descobrir e compor um conjunto de serviços através da internet (JARDIM-GONCALVES; GRILO; STEIGER-GARCAO, 2006). Segundo Papazoglou (2012), SOA é composta por 3 artefatos principais apresentados na Figura 13:

- **Registro de Serviços:** Também conhecido como *Service Broker*, ele disponibiliza a infraestrutura para que os provedores registrem e publiquem as descrições dos seus serviços e para que os consumidores possam procurar, filtrar e obter as informações detalhadas necessárias para iniciar uma comunicação com os serviços previamente registrados.

- **Provedor do Serviço:** Componente que cria o serviço, encarregado de registrar a descrição dele no registro de serviços e disponibilizar a implementação correspondente.

- **Consumidor do Serviço:** Componente cliente que procura o serviço desejado no registro de serviços. Uma vez obtida a resposta com a lista de serviços compatíveis com os seus critérios de busca, escolhe o serviço que ofereça melhores informações e caraterísticas e utiliza a informação necessária disponível na sua descrição para fazer a sua invocação.

Figura 13: Blocos da Arquitetura Orientada a Serviços



Fonte: Adaptada de (JARDIM-GONCALVES; GRILO; STEIGER-GARCAO, 2006)

O ponto de partida da interação entre os blocos da arquitetura começa quando um provedor disponibiliza a descrição de um dado serviço no registro com o objetivo de que possa ser descoberto e invocado pelos possíveis consumidores. Quando um cliente precisa invocar um serviço, procura no registro fornecendo um parâmetro de busca. O registro retorna uma lista de descrições relacionadas com o parâmetro de busca recebido. Então, o cliente escolhe uma das descrições para, finalmente, obter o endereço do serviço, detalhes das mensagens e assim conseguir realizar a invocação remota. No caso de um novo serviço ser adicionado na solução, primeiramente ele deve ser disponibilizado no registro central para que então possa ser invocado pelos outros serviços (CHEN, 2012).

Embora SOA seja independente da plataforma tecnológica, a indústria tem adotado fortemente tecnologias baseadas em padrões abertos, trazendo benefícios na interoperabilidade e aumentando a flexibilidade dos sistemas que estão transacionando. Esse fato está diretamente relacionado à facilidade de trocar um serviço por um outro tecnologicamente equivalente. No entanto, chegar até uma solução completamente compatível com as diretrizes SOA é uma tarefa difícil, principalmente quando é necessária uma mudança tanto tecnológica, quanto cultural, para conseguir migrar de sistemas monolíticos aos sistemas distribuídos fracamente acoplados baseados em serviços (JOSUTTIS, 2007; PAPAZOGLOU, 2012).

Nas seguintes seções serão apresentados os tipos de tecnologias de maior relevância usadas na implementação de um sistema orientado a serviços.

3.5.1 Serviços Web

Serviços web, ou *web services* (WS) são uma representação específica do conceito de *serviço* utilizado em SOA. De forma geral, um serviço web pode ser definido como um sistema de software projetado para suportar a interação entre máquinas conectadas em uma rede de comunicações (W3C, 2004).

Como apresentado anteriormente, SOA é um estilo arquitetural que não está vinculado a nenhuma tecnologia em particular. Porém, o conceito de *web services* é uma alternativa amplamente aceita pela comunidade para a implementação de arquiteturas SOA. Atualmente, na indústria existem duas grandes categorizações de tecnologias para implementar *web services*: serviços web baseados no protocolo *SOAP* e serviços web *RESTful*.

3.5.1.1 Serviços web baseados em SOAP

Esse tipo de serviço é implementado baseado na arquitetura e no conjunto de padrões definidos pela W3C, fornecendo os meios para construir aplicações distribuídas interoperáveis, baseadas em XML. O grupo encarregado definiu *web service* como:

> Um sistema de software projetado para suportar a interação máquina a máquina em uma rede. Ele possui uma interface descrita em um formato processável por máquinas (especificamente WSDL). Outros sistemas interagem com o serviço web da maneira especificada pela sua descrição usando mensagens SOAP, normalmente transmitidas usando HTTP e serializadas no formato de dados XML em conjunto com outros padrões relacionados à Web (W3C, 2004).

Na definição acima, é importante identificar alguns padrões (como WSDL, SOAP e XML) que fazem parte do conjunto de tecnologias que viabilizam a implementação de sistemas seguindo a arquitetura SOA. Esses padrões são apresentados na Figura 14, conhecida como "pilha" de WS-*.

Figura 14: Pilha da arquitetura de web services

Fonte: Adaptada de (W3C, 2004)

Na parte inferior da Figura 14 estão os protocolos de comunicação que dão suporte às trocas de mensagens entre serviços. Como é possível observar, existem vários protocolos que podem ser utilizados, mas o padrão de facto é o protocolo HTTP. Subindo na pilha estão as tecnologias de base que servem para formatar as mensagens e descrições dos serviços, como XML e XSD. Essas tecnologias abrangem e suportam o protocolo SOAP que é utilizado para criar um envelope onde são estruturadas as informações que o serviço recebe e retorna. O padrão WSDL descreve as operações, os dados e os detalhes de localização do serviço para que os clientes possam invocar suas funcionalidades. Por fim, na parte superior da pilha, se encontram os padrões relacionados às atividades de mais alto nível, como descoberta de serviços, coreografia e orquestração. Na sequência, são abordados os principais padrões da pilha WS-* relacionados à implementação de sistemas sob a ótica SOA.

Segundo Josuttis (2007), são 5 os padrões fundamentais para *web services*. Dois deles são padrões gerais que já existiam antes e serviram de suporte: HTTP e XML. Os outros três são: SOAP, WSDL e UDDI, apresentados a seguir.

3.5.1.1.1 SOAP

SOAP é um protocolo de comunicação baseado em XML para trocar mensagens entre clientes e provedores de serviços, independentemente de plataforma de implementação. A função principal desse protocolo é definir uma estrutura padrão de mensagens que são enviadas usando protocolos padrão de comunicação, como HTTP, SMTP, FTP, RMI/IIOP entre outros (JOSUTTIS, 2007; PAPAZOGLOU, 2012).

Existem duas versões principais do padrão, na primeira versão (1.1) o acrônimo oficial de SOAP foi definido como *Simple Object Access Protocol*, porém foi retirado na última versão (1.2).

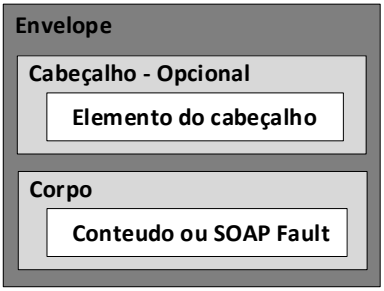
As mensagens SOAP são basicamente documentos formatados em XML, compostos de três elementos principais apresentados na Figura 15:

- **Envelope:** O elemento <*Envelope*> é obrigatório e é a raiz da mensagem. Ele é usado geralmente para declaração dos *namespaces* (definição dos esquemas XSD) usados na mensagem. O envelope tem dois elementos internos, o cabeçalho e o corpo;

- **Cabeçalho:** O <*Header*> é um elemento opcional e tem como objetivo colocar contexto ou enviar dados de controle na mensagem, por exemplo tokens de autenticação, autorização, roteamento, parâmetros de qualidade de serviço QoS (*Quality of Service*) entre outros;

- **Corpo:** O elemento <*Body*> é obrigatório e contém a representação XML dos dados a serem transmitidos (*payload*), tanto da requisição quanto da resposta. No caso de um erro na execução da lógica do serviço invocado, o <*Body*> serve para armazenar a informação detalhada da falha.

Figura 15: Estrutura de uma mensagem SOAP



Fonte: Adaptada de (COULOURIS; DOLLIMORE; KINDBERG, 2012)

A Listagem 1 apresenta um exemplo do XML de uma mensagem SOAP com os 3 elementos principais descritos anteriormente.

Listagem 1: Exemplo de envelope SOAP

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
3     <Header> <!-- opcional -->
4         <!-- informação do cabeçalho -->
5     </Header>
6     <Body>
7         <!-- payload ou SOAP Fault no caso de erro -->
8     </Body>
9  </Envelope>
```

3.5.1.1.2 WSDL

WSDL (*Web Service Description Language*) é uma linguagem de definição de interfaces (*IDL*), baseada em XML, usada para criar descrições de serviços web que sejam legíveis por maquinas. Essa descrição representa as mensagens, tipos de dados, estrutura da informação e os detalhes de implementação de um serviço web (PAPAZOGLOU, 2012). Existem duas versões 1.1: de 2001, que foi implementado para trabalhar unicamente com serviços SOAP, e a 2.0, de 2007, que além de trabalhar com SOAP inclui a possibilidade de descrever outro tipo de serviços, como os REST, sendo está última a recomendação oficial da W3C (W3C, 2007).

Um documento WSDL descreve um serviço web em duas partes principais: uma abstrata e outra concreta. A parte abstrata inclui um conjunto de definições sobre os tipos de dados das informações trocadas nas mensagens e suas funcionalidades. Entretanto, a concreta descreve informações referentes à implementação técnica, como o protocolo utilizado para invocar o serviço e o endereço onde está publicado (COULOURIS; DOLLIMORE; KINDBERG, 2012; W3C, 2007).

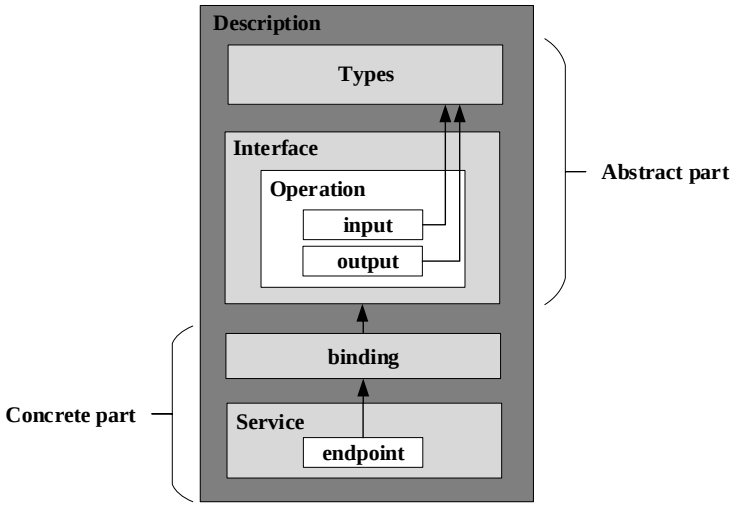
Além do WSDL descrever o serviço web, ele serve para automatizar a criação dos códigos necessários (*stubs*) que usam os clientes para fazer requisições ao serviço corretamente. Portanto, ele facilita a interoperabilidade e a implementação da comunicação. Uma vez estabelecida a comunicação, a descrição WSDL serve como um acordo entre o consumidor e o provedor do serviço (COULOURIS; DOLLIMORE; KINDBERG, 2012; PAPAZOGLOU, 2012).

De forma geral, um documento WSDL 2.0 é composto por 7 elementos principais, apresentados na Figura 16 e descritos a seguir:

- ***Description*:** Serve como um contêiner para as definições de tipos de dados trocados e os elementos próprios do padrão WSDL 2.0 usados para descrever o serviço;
- ***Types*:** Permite definir os tipos de dados que serão usados nas operações expostas pelo serviço; geralmente é feito usando XSD;
- ***Operation*:** É uma interação com o serviço que consiste em um conjunto de mensagens (podem ser ordinárias ou de falhas) trocadas entre o serviço e as outras partes envolvidas na interação (normalmente o cliente do serviço);

- ***Interface*:** Descreve o conjunto de mensagens que um serviço envia e recebe. As mensagens relacionadas são agrupadas logicamente nas operações da interface.
- ***Binding*:** Descreve de forma concreta o protocolo de transmissão e formato da mensagem que podem ser usados para definir um *endpoint*. Ou seja, define os detalhes de implementação necessários para acessar o serviço;
- ***Endpoint*:** Define a localização (URL) onde o serviço está disponível para ser invocado, sempre associado a um elemento *binding* que especifica o protocolo requerido;
- ***Service*:** Descreve um conjunto de *endpoints* que oferecem uma implementação particular do serviço.

Figura 16: Elementos de uma descrição WSDL 2.0



Fonte: Própria

Na Figura 16 podem se observar os conceitos definidos anteriormente, sua classificação e como se relacionam entre eles. Também é possível observar na Listagem 2 um exemplo simplificado do código de um documento WSDL destacando os principais elementos expostos.

Listagem 2. Exemplo simplificado de um documento WSDL 2.0

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl" ... >
   <!-- Abstract Part -->
   <types>
     <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       <xs:element name="request"> ... </xs:element>
       <xs:element name="response"> ... </xs:element>
     </xs:schema>
   <types>

   <interface name="Interface_1">
     <operation name="myOper" pattern="http://www.w3.org/wsdl/in-out">
        <input messageLabel="InMsg" element="request"/>
        <output messageLabel="OutMsg" element="response"/>
     </operation>
   </interface>

   <!-- Concrete Part -->
   <binding name="SoapBinding" interface="tns:Interface_1"
      type="http://www.w3.org/wsdl/soap"
      soap:protocol="http://www.w3.org/soap/bindings/HTTP/"
      soap:mepDefault="http://www.w3.org/soap/mep/request-response">
      <operation ref="myOper" />
   </binding>

   <service name="Service1" interface="tns:Interface1">
     <endpoint name="SoapEndpoint" binding="SoapBinding"
               address="http://www.my-domain.com/soap/"/>
   </service>
</description>
```

3.5.1.1.3 UDDI

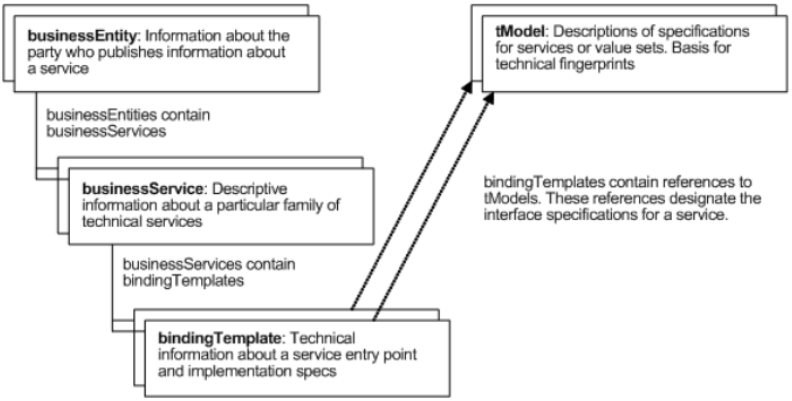
A especificação UDDI (*Universal Description, Discovery and Integration*) foi criada pela OASIS e tem como objetivo fornecer uma forma padronizada para publicar, classificar, administrar e descobrir serviços web. Representa o artefato "*registro de serviços*" dos componentes da arquitetura orientada a serviços apresentados na Figura 13.

Um registro UDDI provê um mecanismo para categorizar empresas e serviços usando taxonomias junto com metadados, facilitando que os clientes possam localizar de uma forma muito mais apurada os serviços que correspondam aos seus requisitos (PAPAZOGLOU, 2012).

Na Figura 17 podem ser observadas as quatro estruturas de dados definidas na especificação para armazenar as informações no registro (COULOURIS; DOLLIMORE; KINDBERG, 2012; OASIS, 2004):

- ***BusinessEntity*:** Também conhecidas como páginas brancas (*White Pages*). Armazenam informações gerais, que incluem nome, contato e traduções em diferentes línguas, tanto da empresa quanto dos serviços que disponibiliza;
- ***BusinessService*:** Estas estruturas são chamadas de páginas amarelas (*Yellow Pages*) e armazenam informações mais detalhadas em relação às categorias nas quais estão classificados os serviços do registro;
- ***BindingTemplate*:** Chamadas de páginas verdes (*Green Pages*)**,** contém os detalhes técnicos sobre o acesso ao serviço, fornecendo a informação como a URL e os protocolos necessários para efetuar a invocação. Geralmente referencia diretamente as descrições dos serviços armazenadas nos *tModels*.
- ***tModel:*** É a abreviação de *Technical Models*. Normalmente contém os documentos WSDL armazenados fora do banco de dados do registro e que são acessados via URL. Da mesma forma que os *BindingTemplate*, os *tModels* fazem parte das chamadas páginas verdes.

Figura 17: Estruturas de dados da especificação UDDI

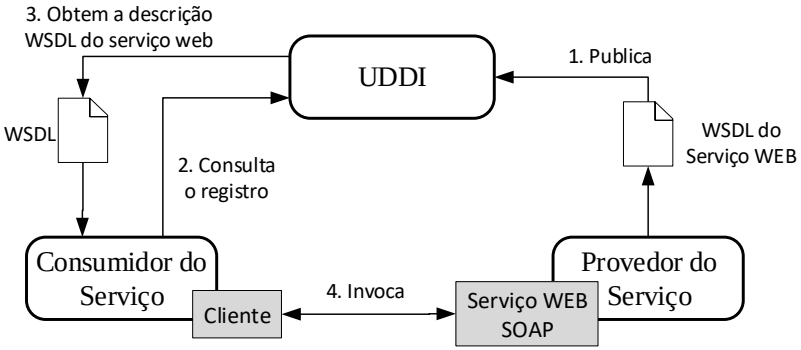


Fonte: (OASIS, 2004)

A última versão da especificação UDDI é a 3.0.2, de 2004 (OASIS, 2004). Embora essa especificação tenha sido suportada por empresas de grande influência na indústria, como Microsoft, IBM e SAP, em 2006 os servidores públicos que essas empresas mantinham foram desligados (JOSUTTIS, 2007). Atualmente existem implementações de código aberto da especificação como jUDDI[3], usada principalmente para criação de servidores privados.

Como observado na Figura 18, com os três padrões apresentados anteriormente da pilha WS-* (SOAP, WSDL, UDDI), é possível implementar as três operações principais de SOA: publicação no registro de serviços, busca/localização de serviços e invocação baseada na descrição do serviço.

Figura 18: Implementação de SOA com serviços web



Fonte: Própria

3.5.1.1.4 BPEL

Uma das principais vantagens de uma implementação baseada na filosofia da arquitetura SOA é a possibilidade de obter interoperabilidade entre serviços. Isso permite agrupar e organizar um conjunto de serviços e apresentá-los como um serviço só (composição de serviços), que executa tarefas de maior complexidade.

Essa composição de serviços levanta novos desafios no âmbito da integração de sistemas, o que requer o projeto de mecanismos que forneçam controle das possíveis interações que possam ocorrer em

[3] Apache jUDDI - https://juddi.apache.org

resposta a diferentes eventos como, por exemplo, controlar o fluxo das invocações (orquestração), recuperação de erros, memória do estado do processo, intervenção de agentes externos, entre outros (PAPAZOGLOU, 2012).

O padrão BPEL apresenta uma solução aos desafios expostos anteriormente. Ele define um modelo e uma gramática em XML para descrever o comportamento de um processo de negócio (representado como uma composição de serviços) com base nas interações entre seus parceiros (serviços). O processo BPEL estabelece como devem ser coordenadas as interações entre esses parceiros para atingir um objetivo de negócio bem como o estado e a lógica necessária para essa coordenação. O padrão também introduz mecanismos sistemáticos para lidar com exceções de negócios e falhas de processamento (OASIS, 2007).

A seguir são descritos alguns dos elementos de maior relevância disponibilizados no padrão WS-BPEL para representar processos de negócio (OASIS, 2007):

- ***Receive****:* Atividade que recebe mensagens*;*
- ***Reply****:* Retorna uma reposta a uma mensagem recebida;
- ***Invoke****:* Invoca uma operação de um determinado Web Service participante do processo de negócio;
- ***Assign****:* Atribui os valores às variáveis do processo;
- ***Throw****:* Lança exceções controladas quando o processo entrar em uma condição de erro;
- ***Exit****:* Atividade usada para encerrar imediatamente uma instância do processo de negócio;
- ***Wait****:* Especifica um atraso até que um determinado prazo seja atingido;
- ***Sequence****:* Contêiner que engloba uma ou mais atividades que são executadas sequencialmente;
- ***If****:* Provê comportamento condicional, fornecendo a possibilidade de seguir caminhos diferentes segundo condições especificas.

3.5.1.2 Serviços Web RESTful

O termo REST, do inglês (*REpresentational State Transfer*), foi definido por Roy Fielding na sua tese de doutorado apresentada no ano 2000 (FIELDING, 2000). Fielding define REST como um estilo arquitetural baseado em um conjunto de restrições para sistemas distribuídos de hipermídia (Cliente - Servidor) como a World Wide Web (WWW).

O REST adota uma abordagem centrada em *recursos*, definindo-os como uma abstração de qualquer conceito coerente, por exemplo, um livro, uma foto, uma pessoa, um sentimento, etc. Uma característica desses recursos é que são detectáveis pelos clientes de forma única por meio de *identificadores* chamados URIs (*Uniform Resource Identifier*) e acessíveis por uma *interface unificada* que, no caso de REST é o protocolo HTTP. Atualmente, os mais usados são GET, POST, PUT e DELETE (FIELDING, 2000; PAPAZOGLOU, 2012; RICHARDSON; RUBY, 2007; ROTEM-GAL-OZ, 2012).

A combinação da semântica do método HTTP usado para interagir com o recurso e, adicionalmente, a URI associada a ele, fazem com que a ação a ser executada sobre o recurso na invocação seja única no servidor, eliminando assim a possibilidade de ambiguidades.

Considerando que o recurso é apenas uma abstração de um conceito do mundo real, a *representação* dele é feita como um encapsulamento das suas informações e do estado da interação com o servidor. Essa representação é serializada em um dado formato (XML, HTML, JSON, RDF), que tanto o cliente quanto o servidor combinam e entendem em cada requisição e/ou resposta que ocorre na interação (ALLAMARAJU, 2010).

Uma das restrições que mais caracteriza o estilo de arquitetura REST é que o estado e representação dos recursos nunca são mantidos no servidor (*Stateless*). Isso é, a cada requisição o cliente é responsável por fornecer todos os dados que dão o contexto a ela (transferência de estado – *State Transfer*), permitindo interação sem sobrecarregar o servidor, que com isso não precisa manter na memória os dados de sessão.

Conforme Richardson e Ruby (2007), o termo serviços "RESTful" é utilizado para se referir aos serviços que adotaram os requisitos definidos pelo estilo arquitetural REST. Esse tipo de serviços tem ganhado muita adoção na indústria na última década devido às caraterísticas expostas anteriormente: sua facilidade de implementação, aproveitamento do já amplamente conhecido protocolo HTTP e pelo suporte que as linguagens de desenvolvimento mais populares oferecem

(COULOURIS; DOLLIMORE; KINDBERG, 2012; RICHARDSON; RUBY, 2007).

Em contraste com os serviços web baseados em SOAP que são orientados a mensagens e que definem uma descrição formal padronizada, os serviços RESTful são orientados a recursos e não definem formalmente uma descrição completa do serviço, sendo essa uma das diferenças mais relevantes entre as duas abordagens. Em consequência, isso tem gerado várias discussões entre os membros da indústria, cada qual tentando impor uma abordagem sobre a outra. No entanto, alguns autores, como Papazoglou (2012), consideram que cada abordagem tem seu campo de atuação e a escolha depende das necessidades do projeto. Consequentemente, classifica os serviços web em dois grandes tipos: os voltados para consumo humano e os voltados para serem consumidos por máquinas.

Na primeira categoria, os serviços RESTful são os mais usados devido à facilidade de implementação. Os desenvolvedores têm adotado as integrações com APIs externas, como as da *Google*, *Facebook*, *Amazon* de uma maneira pontual, sendo responsabilidade do implementador ler e entender as especificações e os detalhes técnicos de como interagir com essas APIs.

Por outro lado, quando o cenário envolve um grau de dinamicidade em tempo de execução entre os serviços, é muito importante que as maquinas consigam interpretar as descrições dos serviços automaticamente. É nessa categoria que os serviços web baseados em SOAP, junto com as outras tecnologias já apresentadas na seção 3.5.1.1, são relevantes; porém, com a desvantagem do aumento da complexidade no processo de serialização das mensagens, aumento do consumo de banda, entre outros.

3.6 BARRAMENTO DE SERVIÇOS EMPRESARIAIS (ESB)

Em uma aplicação SOA, os seus módulos são vistos como serviços que interagem por meio de troca de mensagens para executar uma tarefa. Considerando a complexidade que impõe essa natureza colaborativa, além da possibilidade de que cada serviço seja implementado em tecnologias diferentes, surgiu a necessidade de conceber uma abordagem visando diminuir os esforços necessários na implementação, manutenção e gerenciamento da integração dos diversos serviços.

Segundo Bhadoria, Chaudhari, Tomar (2017) e Papazoglou (2012), o barramento de serviços empresariais ou ESB, do inglês *Enterprise Service Bus*, é uma estrutura de software baseada em padrões

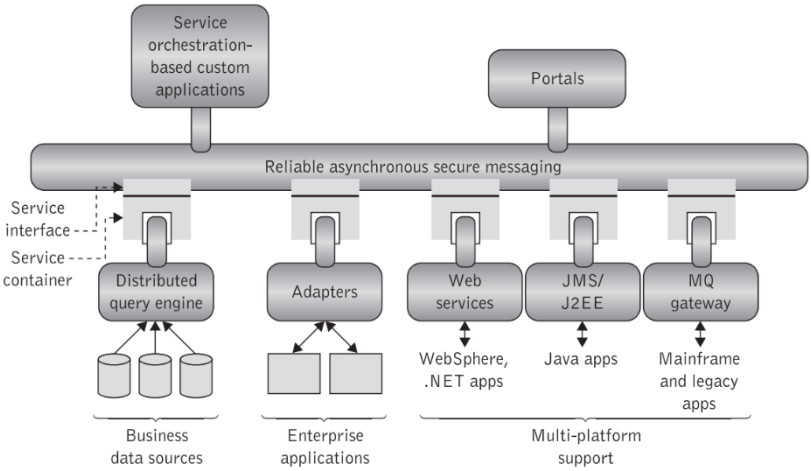
abertos e projetada para prover uma camada de comunicação intermediária que integra serviços heterogêneos por meio de uma plataforma comum. Fornece o suporte de infraestrutura necessário para implementar tarefas habituais em aplicações de grande escala orientadas a serviços, como: roteamento de mensagens, conversão de protocolos, transformação de mensagens, confiabilidade, segurança, entre outras. Ele possibilita uma interoperabilidade nos serviços de aplicativos legados e ao mesmo tempo mantém um baixo acoplamento entre eles.

Uma das principais funcionalidades do ESB é atenuar os problemas de heterogeneidade entre aplicativos executados em plataformas heterogêneas e que usam diversos formatos de dados. O barramento funciona como facilitador de transporte e transformação para permitir a distribuição desses serviços em diferentes sistemas e ambientes de computação por meio de adaptadores e interfaces baseadas em padrões.

A vantagem de oferecer adaptadores baseados em padrões é que para um serviço se comunicar com os outros disponíveis na colaboração só precisa de um adaptador que implemente o mesmo protocolo que ele usa. Tarefas como roteamento, transformação da mensagem e do formato de dados são realizadas diretamente pelo ESB, sem a necessidade do implementador escrever código (MENGE, 2007). Isso agiliza a integração e diminui o trabalho necessário, dado que se o ESB não servisse de mediador entre os *n* serviços heterogêneos de uma colaboração, cada serviço teria que criar *n*-1 adaptadores para se comunicar com os outros, abordagem típica de uma integração ponto a ponto que incrementa dificuldade de manutenção, escalabilidade da solução e em consequência o custo do projeto.

Na Figura 19 é apresentado um diagrama de integração de múltiplos sistemas heterogêneos baseada em ESB. Pode-se observar como diferentes aplicativos, como web services, *Java Message Services* (JMS), aplicativos legados, fontes de dados, entre outros, são conectados ao barramento central usando os adaptadores específicos a cada um deles, permitindo assim sua comunicação.

Figura 19: ESB conectando diferentes sistemas e tecnologias



Fonte: (PAPAZOGLOU, 2012)

A seguir são listadas as principais caraterísticas que um ESB deve suportar (MENGE, 2007):

- **Invocação:** Tida como a capacidade de um ESB enviar requisições a serviços/recursos integrados e receber suas respectivas respostas. Isso significa que um ESB deve suportar uma série de padrões de comunicação, como os dos serviços web citados na seção 3.5.1, a API JMS, e também deve ser capaz de lidar com os protocolos subjacentes que suportam as comunicações, como HTTP, RMI, SSL, SMTP, JDBC, FTP, TCP, UDP, entre outros;

- **Roteamento:** É a capacidade de decidir o destino de uma mensagem durante seu transporte. Os serviços de roteamento são um recurso essencial de um ESB porque permitem desacoplar a fonte de uma mensagem do destino final. A decisão sobre a qual destino uma mensagem é enviada pode ser feita com base em várias condições (regras de roteamento), o que leva a que um ESB ofereça diferentes tipos de roteadores, tais como: o roteador baseado em conteúdo, baseado em lista de destinatários, sequenciador, entre outros;

- **Mediação:** Refere-se a todas as transformações ou traduções entre recursos diferentes, incluindo protocolo de transporte, formato de mensagem e conteúdo da mesma. Essas transformações são muito importantes para a integração, porque é um problema muito comum. O ESB deve fornecer componentes de transformação genéricos e configuráveis com a capacidade de invocar outros recursos que estão conectados ao barramento para cumprir com sua tarefa;

- **Adaptadores:** São artefatos de software que implementam a lógica de integração do ESB com diferentes APIs de terceiros e protocolos proprietários. Exemplos do uso desse tipo de adaptadores é a integração com aplicativos empresariais, como ERP (*Enterprise Resource Planning*), SCM (*Supply Chain Management*) e CRM (*Customer Relationship Management*);

- **Segurança:** Uma infraestrutura de integração precisa ter mecanismos de segurança adequados para fornecer troca de mensagens segura. Isso inclui tarefas como criptografar e descriptografar o conteúdo das mensagens, gerenciar autenticação e autorização implementando os protocolos de segurança reconhecidos na indústria;

- **Gerenciamento:** Outra funcionalidade de grande importância que um ESB deve fornecer é o gerenciamento de recursos de auditoria e log, que permitem monitorar a infraestrutura de integração e também a execução do processo;

- **Processamento de eventos:** O ESB deve incluir mecanismos de interpretação e correlação de eventos que viabilizam implementações assíncronas baseadas em arquiteturas orientadas a eventos. Por exemplo, o suporte ao mecanismo de comunicação Publish/Subscribe;

- **Ferramental de integração:** Finalmente, é de grande importância que o ESB ofereça os meios para utilizar de forma amigável todas as características descritas anteriormente. Portanto, deve prover uma interface gráfica de projeto que permita facilitar o processo de construção, teste e implantação dos processos de integração.

4 REVISÃO DO ESTADO DA ARTE

Este capítulo tem como objetivo apresentar o procedimento e resultados da revisão sistemática da literatura realizada.

4.1 REVISÃO SISTEMÁTICA DA LITERATURA

Para a elaboração da revisão do estado da arte foi utilizado o método SLR - *Systematic Literature Review* (KITCHENHAM; CHARTERS, 2007). Um SLR é executado em três fases principais.

A primeira fase é a de planejamento, que de forma geral, consiste em identificar o objetivo e a justificativa da revisão da literatura no problema que está tentando-se resolver. No capítulo 1 são abordados esses tópicos.

A segunda fase consiste em estabelecer o processo que será executado na revisão da literatura. Para isso foram selecionadas as bases de dados de pesquisa, definida uma *string* de busca geral contendo as palavras-chave relacionadas ao tema do trabalho e foram escolhidos os critérios de exclusão usados no refinamento dos artigos retornados na primeira busca. A *string* de busca e o resultado da sua aplicação nas bases de dados selecionadas são apresentadas a seguir.

Considerando as áreas de pesquisa que abrangem o objetivo desta dissertação, foram escolhidas bases de dados voltadas principalmente para temas de engenharia, software e computação: *IEEExplore*, *Sciencedirect*, *SpringerLink* e *ACM*.

Uma vez selecionadas as bases de dados, e considerando os objetivos e a definição do problema desta dissertação, foi elaborada uma *string* de busca que contem termos (em inglês) relacionados com "Empresas Virtuais", "Interoperabilidade Semântica" e "Arquitetura Orientada a Serviços" para ser aplicada em cada uma das bases, considerando também alguns dos seus termos equivalentes.

A *string* geral obtida pode ser apreciada na Figura 20. Ela é composta de duas partes principais, restringindo os resultados apenas aos relacionados com os termos (e sinônimos) citados acima.

Contudo, é importante mencionar que cada base de dados define sua funcionalidade de busca avançada com seus próprios filtros. Assim, a sintaxe da *string* foi ajustada (sem modificar sua lógica) antes de ser aplicada em cada base de dados.

Figura 20. *String* geral de busca na SLR

```
[
    (
        ("semantic" AND ("interoperability" OR "integration"))
        AND
        ("Ontology" OR "Ontologies")
    )
    AND
    (
        "Networked Organizations" OR "Networked Organisations"
        OR "Collaborative Networks" OR "virtual enterprise"
        "virtual organization" OR "virtual organisation"
    )
]
AND
[
(
    (
        "web service"
        OR "service composition"
    )
    AND
    (
        "service oriented architecture"
        OR "SOA"
        OR "service-oriented architecture"
    )
]
```

Fonte: Própria

Os trabalhos retornados pela aplicação da *string* de busca nas diferentes bases de dados foram filtrados para obter só artigos de revistas e congressos publicados entre 2000 e 2017. Os resultados podem ser observados no Quadro 1, apresentado a seguir.

Quadro 1: Trabalhos retornados na SLR

Base de dados	**Trabalhos retornados**
IEEExplore	247
Sciencedirect	116
SpringerLink	228
ACM	114
Total	705

Fonte: Própria

Mesmo que seja aplicada a *string* de busca alguns resultados retornados não atendiam ao tema específico desta pesquisa. Portanto,

foram definidos alguns critérios de exclusão para eliminar esses trabalhos da revisão. Os critérios definidos foram:

1. Artigos que não tratavam efetivamente de interoperabilidade;
2. Artigos focados unicamente em algoritmos de alinhamento de ontologias;
3. Artigos que não sejam escritos em inglês,
4. Artigos que sumarizavam um trabalho realizado em menos de 4 páginas.
5. Artigos duplicados do mesmo trabalho (existem alguns trabalhos que são publicados em diferentes níveis de detalhamento em diferentes bases de dados).

Os trabalhos retornados depois dos critérios de exclusão, foram avaliados utilizando quatro filtros principais, começando com o título, palavras-chave, abstract e conclusões. Uma vez selecionados os trabalhos, foram lidos e estudados com maior profundidade. Finalmente, como resultado do processo de revisão sistemática da literatura foram identificados oito trabalhos (Quadro 2) considerados os mais relevantes e fortemente relacionados com a proposta de pesquisa desta dissertação.

Quadro 2. Trabalhos selecionados como resultado da SLR

	Título do trabalho	**Referência**
1	A process-oriented service infrastructure for networked enterprises	(LUKÁČ et al., 2017)
2	Semantic integration via enterprise service bus in virtual organization breeding environments	(SCHRATZENSTALLER; BALDO; RABELO, 2016)
3	An ontology-based framework for virtual enterprise integration and interoperability	(ABADI et al., 2016)
4	A cloud-based platform to ensure interoperability in aerospace industry	(KHALFALLAH et al., 2016)
5	Integration framework with semantic aspect of heterogeneous system based on ontology and ESB	(SHI et al., 2014)
6	Semantic Mediation Bus An Ontology based runtime Infrastructure for service interoperability	(ZHU, 2012)

7	Support of semantic interoperability in a service-based business collaboration platform.	(FURDÍK et al., 2011)
8	Smart configuration of dynamic virtual enterprises	(RABELO et al., 2004)

Fonte: Própria

No processo de revisão sistemática da literatura, a última fase consiste em gerar um relatório que apresente os pontos principais dos trabalhos selecionados com o intuito de poder realizar uma análise comparativa em relação à própria proposta.

A seguir é apresentada a última fase da revisão da literatura (relatório), iniciando com uma breve descrição de cada trabalho selecionado e, por fim, apresentando um quadro comparativo entre eles.

A PROCESS-ORIENTED SERVICE INFRASTRUCTURE FOR NETWORKED ENTERPRISES

O trabalho apresentado em Lukáč et al (2017) descreve as soluções de software que foram desenvolvidas dentro dos projetos de pesquisa europeus SPIKE e VENIS. Os resultados da pesquisa evidenciam que a combinação de serviços web semânticos, modelos de processos de negócios e barramento de serviços oferecem uma plataforma tecnológica adequada para a infraestrutura de TI em RCOs.

A tecnologia ESB foi aplicada em ambas soluções implementadas e foi essencial para o gerenciamento de serviços, eventos e recursos de dados fornecidos e consumidos pelos participantes da RCO.

Em relação aos modelos semânticos usados, os autores afirmam que uma abordagem leve (*light-weight*) com a construção semiautomática da base de conhecimento e anotações livres de esquemas mostrou-se mais vantajosa que os modelos semânticos estáticos em relação à flexibilidade e adaptabilidade das colaborações de negócios.

A abordagem empregada no projeto VENIS foi baseada em serviços web RESTful, fornecendo funcionalidades comparáveis às tecnologias SA-WSDL (*Semantic Annotations for WSDL*) e WSMO (*Web Service Modeling Ontology*), usadas no SPIKE.

Os protótipos implementados dos sistemas SPIKE e VENIS foram testados em uma série de aplicações-piloto e implantados em ambientes industriais envolvendo projeto CAD e CAE aeroespacial, serviços de documentação, desenvolvimento de sistemas de TI, e serviços de

transporte público. Os autores concluem que os resultados da avaliação demonstram a aceitação pelos usuários da tecnologia fornecida.

As abordagens desenvolvidas nesse trabalho foram as mais próximas em relação à proposta desta dissertação. Porém, os trabalhos não incluem um modelo de referência genérico, o que impacta diretamente no processo de estabelecimento de mapeamentos semânticos, tendo eles que ser realizados para todas as fontes da colaboração cada vez que uma nova fonte de dados é adicionada dado que a arquitetura de ontologias adotada é de múltiplas ontologias (apresentada na Figura 9). Além disso, essa abordagem não considera diferenças de esquemas de dados, que é um ponto relevante para se estabelecer interoperabilidade estrutural.

SEMANTIC INTEGRATION VIA ENTERPRISE SERVICE BUS IN VIRTUAL ORGANIZATION BREEDING ENVIRONMENTS

Schratzenstaller, Baldo e Rabelo (2016) propuseram a criação de uma ontologia global para o ambiente virtual de empresas (ACV), onde os dados das empresas participantes são adicionados à ontologia quando é executada a fase de configuração do ACV.

Portanto, caso seja necessário participar de uma eventual empresa virtual, o estabelecimento da comunicação pode ser feito facilmente, já que todos os alinhamentos semânticos dos parâmetros trocados entre as invocações dos serviços já são conhecidos.

A abordagem foi implementada usando uma ferramenta ESB aberta, e o componente que resolve as diferenças semânticas (mapeador semântico) foi desenvolvido em Java como um plug-in para ser adicionado ao ESB.

Algumas limitações observadas na abordagem proposta dizem respeito à necessidade de atualizar a ontologia caso o modelo de dados de uma das empresas participantes do ACV mude, processo que teria que ser feito manualmente. Além disso, os mapeamentos semânticos dos diferentes conceitos na ontologia são feitos manualmente e, portanto, conforme cresce o número de participantes do ACV, também cresce o número de mapeamentos necessários, dificultando escalar e realizar manutenção na solução.

AN ONTOLOGY-BASED FRAMEWORK FOR VIRTUAL ENTERPRISE INTEGRATION AND INTEROPERABILITY

Abadi et al. (2016) propuseram um framework que tenta suportar integração e interoperabilidade em uma empresa virtual por meio de um modelo semântico baseado em ontologias relacionada a 3 grandes áreas: produto, processo de manufatura e sua cadeia de suprimentos.

Os autores propõem uma arquitetura de ontologia hierárquica com 4 camadas visando garantir interoperabilidade semântica, incorporação de novo conhecimento, e troca de informação efetiva e dinâmica entre aplicações, usando as capacidades de inferência que as ontologias proveem e suportando o processo de tomada de decisões no ciclo de desenvolvimento do produto.

A arquitetura proposta começa no nível superior com uma ontologia geral, que abarca toda a informação relevante no ciclo de desenvolvimento do produto, e aumenta o nível de detalhamento conforme se desce as camadas. Na camada seguinte é criada uma ontologia com as suas regras de inferência por cada atividade principal feita no ciclo (Projeto do produto, manufatura, logística, etc.). Na terceira camada estão as ontologias que representam tarefas específicas das atividades da camada anterior, para suportar o processo de tomada de decisão automática. Por fim, tem-se a camada de aplicação, que é instanciada em cada EV baseada na ontologia geral da primeira camada, possibilitando resolver buscas mais detalhadas indo de dados gerais até dados específicos.

O trabalho descrito foca em uma arquitetura conceitual orientada à solução do problema de interoperabilidade semântica usando ontologias. Porém, não define outros aspectos importantes que estão envolvidos no processo de interoperabilidade, como o suporte para efetivar a integração, entrada e saída de fontes de dados, entre outros. A proposta de uma arquitetura de 4 camadas de ontologias implica a criação tanto das ontologias superiores quanto das ontologias de domínio, tarefas, e aplicação. Todas essas ontologias e a integração entre elas têm que ser realizadas por uma pessoa especialista na área, o que dificulta a adoção desta arquitetura em cenários onde as PMEs são parte importante da colaboração. O trabalho não detalha a criação nem a integração dessas ontologias. Na data de publicação do artigo não havia sido implementada uma prova de conceito para avaliar a proposta.

A CLOUD-BASED PLATFORM TO ENSURE INTEROPERABILITY IN AEROSPACE INDUSTRY

Khalfallah et al. (2016) propuseram uma abordagem para tratar problemas de interoperabilidade sintática, estrutural e semântica na área de desenvolvimento colaborativo de produtos em RCOs. O trabalho faz uso de um modelo de mediação semântica de dados em duas fases e orientado a serviços para assegurar a interoperabilidade. Além disso, o modelo é projetado e implementado para usar uma plataforma baseada em nuvem que permita cenários mais complexos de colaboração. O modelo implementado é voltado para a indústria aeroespacial.

O fluxo de integração é dividido em duas fases: mapeamento de esquema e vocabulário, e transformação de dados. Para o mapeamento de dados foi usada uma arquitetura de ontologia híbrida, onde existe uma ontologia global de referência criada com base no padrão AP214 da indústria aeronáutica e uma ontologia associada a cada empresa. Essas ontologias são mapeadas com a ontologia global com o objetivo de alinhar os diferentes modelos de dados das empresas com o modelo central. Uma vez gerados os mapeamentos segue a etapa de transformação, que usa essa informação para transformar as mensagens que são enviadas entre sistemas em tempo de execução.

O modelo criado nesta proposta usa uma ontologia de referência voltada para a indústria aeroespacial em específico, o que restringe a sua aplicabilidade em colaborações fora desse campo.

Por outro lado, a criação de ontologias e alinhamento entre elas é um processo complexo que demanda tempo e normalmente requer uma pessoa com conhecimentos avançados na área, o que para PMEs pode ser um obstáculo ao adotar o modelo proposto.

Além disso, o modelo assume que todas as fontes de dados usam XML para representar suas informações, portanto, restringe outros tipos de dados que são muito comuns na indústria, como JSON.

Na data da publicação do artigo os autores ainda não tinham implementado uma prova de conceito do modelo proposto.

INTEGRATION FRAMEWORK WITH SEMANTIC ASPECT OF HETEROGENEOUS SYSTEM BASED ON ONTOLOGY AND ESB

O trabalho proposto por Shi et al. (2014) descreve um framework orientado a serviços para a integração de sistemas heterogêneos baseado em ontologias e suportado por uma ferramenta ESB (*JBossESB*).

O framework proposto é baseado em uma base de regras de inferência e em uma estrutura de ontologias de 3 camadas. A camada de sistema contém as ontologias que representam cada sistema na integração. A camada de funções consiste em uma ontologia global usada para atingir os requisitos do negócio. Finalmente, a camada da ontologia comum representa a base de dados da informação trocada na integração.

Foram identificadas algumas limitações no trabalho aqui proposto, principalmente quanto à integração das ontologias usadas no framework. Isto porque, para integrar um novo sistema, é necessário desenvolver manualmente cada ontologia local e adicioná-la na estrutura de ontologias. Isso implica a intervenção de um especialista no domínio da aplicação para realizar os mapeamentos necessários entre os conceitos da base e a nova ontologia adicionada, o que causa um maior tempo de configuração para introduzir um novo sistema na colaboração e, portanto, também dificulta a adoção dessa arquitetura proposta em PMEs. Outro aspecto importante, que é o estabelecimento de alinhamentos entre os termos das diferentes ontologias, é feito manualmente.

SEMANTIC MEDIATION BUS: AN ONTOLOGY-BASED RUNTIME INFRASTRUCTURE FOR SERVICE INTEROPERABILITY

Neste trabalho, o autor Zhu (2012) propôs uma infraestrutura de tempo de execução orientada a serviços chamada de barramento de mediação semântica. Ela tem como alvo permitir a interoperabilidade semântica entre serviços web através de ontologias comuns mesmo que os serviços sejam implementados usando modelos de dados e padrões de mensagens diferentes.

O objetivo é introduzir mecanismos que possibilitem mediação semântica em um barramento de serviços ESB. A proposta se baseia em uma arquitetura de ontologias híbrida. Cada serviço é representado por uma ontologia, a qual é mapeada manualmente com uma ontologia global que representa um vocabulário intermediário na colaboração.

A proposta adota o padrão OWL para representar as ontologias, SA-WSDL e WSMO para anotações semânticas dos serviços web, e XSLT, RDF e XML na mediação das mensagens que viajam no ESB. Foi implementado um protótipo usando como ESB a ferramenta *Apache Synapse*.

A abordagem proposta assume que todos os serviços que vão ser integrados usam como linguagem de descrição WSDL. Ainda que essa descrição já seja semanticamente anotada, isto é, que cada informação do esquema que representa os dados trocados pelo serviço (*payload*) já seja

mapeada com a ontologia global de alguma forma. Em termos práticos, essa suposição transfere a complexidade da realização dos mapeamentos semânticos aos provedores dos serviços, já que são eles que devem atualizar suas interfaces adicionando “enlaces” entre seus termos e os termos presentes na ontologia global usada dentro do ESB como vocabulário intermediário.

SUPPORT OF SEMANTIC INTEROPERABILITY IN A SERVICE-BASED BUSINESS COLLABORATION PLATFORM

Furdík et al. (2011) apresentam o projeto SPIKE, que tem como finalidade melhorar a interoperabilidade de serviços e processos em RCOs. Concentra-se na concepção e desenvolvimento de uma plataforma de software que possibilite de forma fácil, ágil e segura a criação de alianças de negócios virtuais.

Os autores colocam uma ênfase especial aos processos de negócios semânticos que possibilitam a integração de serviços heterogêneos fornecidos e consumidos pelos membros da aliança. Também, consideram aspectos de segurança de acesso aos serviços disponibilizados. A integração dos diferentes serviços e processos de negócio é suportada pelo uso de ontologias e infraestrutura de comunicação baseada em ESB.

Para representar as ontologias relacionadas aos processos de negócio e domínio os autores usaram WSMO. A solução proposta admite o uso unicamente de serviços web SOAP; portanto, para adicionar semântica aos dados trocados entre serviços da colaboração usaram SA-WSDL. Os processos de negócio foram modelados usando BPMN e transformados em BPEL para serem de fato executados.

O trabalho propõe um framework que abarca as fases principais do processo de integração, começando desde a configuração até o encerramento de atividades. Porém, algumas atividades ao longo dessas fases não são abordadas na proposta, como a reconfiguração dinâmica dos serviços que participam nos processos colaborativos caso um deles precise ser trocado. Além disso, a solução proposta não considera um modelo de referência, portanto, o processo de anotação semântica de serviços e o alinhamento dessas ontologias geradas (WSMO) é difícil, dado que para obter um processo de mediação correto tem que ser alinhadas as *n* ontologias criadas na colaboração, processo que deve ser realizado manualmente.

SMART CONFIGURATION OF DYNAMIC VIRTUAL ENTERPRISES

A pesquisa desenvolvida por Rabelo et al (2004) visa automatizar algumas das atividades necessárias na etapa de configuração da EV. Foram propostas três ferramentas que suportam a especificação da informação a ser trocada pelos parceiros da EV durante seu ciclo de vida, a integração dos sistemas legados com a plataforma EV, a identificação da topologia na EV e a definição dos direitos de acesso à informação compartilhada.

A solução adota um modelo de referência proprietário de dados chamado *Distributed Business Process* (DBP) usado para construir o banco de dados geral com conceitos correlatos a manufatura, vendas, distribuição, etc. Foi desenvolvida uma ferramenta que obtém o esquema XML correspondente ao modelo de dados DBP e que gera automaticamente as 26 classes e tabelas do banco de dados que vai servir de referência. Posteriormente, com ajuda do software desenvolvido, são lidos os bancos de dados de cada um dos participantes na EV cujos campos de informação são apresentados de uma forma amigável ao usuário que é encarregado de estabelecer o mapeamento entre cada campo de dados dos participantes e o modelo geral. Com isso, resolvendo manualmente os problemas de semântica, como relação entre conceitos sinônimos. O trabalho está voltado para a etapa de criação e pré-operação da EV e não contempla as etapas de evolução e dissolução do ciclo de vida.

O framework foi desenvolvido usando programação de sistemas multiagentes nas linguagens Java e C++. Para o funcionamento distribuído foi adotado CORBA e como formato de representação de dados o padrão XML.

Foram identificadas algumas limitações no trabalho proposto. A primeira diz respeito à necessidade de fornecer acesso ao esquema de banco de dados de cada membro à ferramenta de integração. Isso é considerado um risco de segurança dado que nem todas as informações disponíveis no banco de dados devem ficar expostas na integração. Para isso, os autores propõem um mecanismo de autorização para a leitura de informação entre membros; porém, o sistema de integração consegue acessar informações que poderiam ser sensíveis. Além disso, mesmo que o processo de alinhamento de modelos de dados seja assistido por uma ferramenta, ela só apresenta os esquemas e deixa a responsabilidade do processo ao usuário da plataforma, o que, dependendo do tamanho e heterogeneidade dos modelos pode ser um processo demorado.

4.2 CONSIDERAÇÕES FINAIS

Fundamentado no relatório da revisão sistemática da literatura apresentado anteriormente e considerando o exposto por Panetto et al (2016) e Picard et al (2010), foram identificados um conjunto de critérios de comparação entre os oito principais trabalhos estudados. O resultado dessa comparação é apresentado no Quadro 3. As colunas correspondem a cada um dos trabalhos enumerados anteriormente no Quadro 2 e a última coluna (representada com a letra P) corresponde à proposta desta dissertação. São marcados com X cada um dos requisitos que são abordados nos trabalhos mesmo sendo de forma parcial.

Quadro 3. Comparação de trabalhos relacionadas e proposta

Trabalhos SLR	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**P**
Interoperabilidade									
Sintática	X	X	X	X	X	X	X	X	**X**
Estrutural				X					
Semântica	X	X	X	X	X	X	X	X	**X**
Criação de mapeamentos semânticos									
Manual	X	X	X	X	X	X	X	X	**X**
Semiautomático	X			X					**X**
Uso de Ontologias	X	X	X	X	X	X	X		**X**
Serviços Web									
Orientação a Serviços	X	X		X	X	X	X	X	**X**
REST	X					X			**X**
SOAP	X	X		X		X	X		**X**
Segurança	X						X		
Processos de negócio									
Usa modelo de referência				X				X	**X**
Fase do ciclo de vida da RCO									
Criação	X	X	X	X		X	X	X	**X**
Operação	X	X	X	X	X	X	X	X	**X**
Evolução	X		X						**X**
Dissolução	X					X	X		**X**

Fonte: Própria

Segundo a comparação feita, pode-se observar que a proposta desta dissertação visa abranger a maioria dos critérios de avaliação

selecionados a partir dos trabalhos analisados. Identificou-se que poucos trabalhos relacionados usam modelos de referência na suas soluções, fazendo com que suas abordagens sejam menos eficientes na interoperação, já que não permitem aproveitar possíveis integrações com os padrões existentes. Além disso, a maioria das abordagens dos trabalhos, mesmo sendo recentes, optaram por suportar apenas serviços web baseados em SOAP e formato de dados XML, o que pode diminuir as possibilidades de formação de RCOs considerando que os serviços REST/JSON atualmente têm uma ampla participação e aceitação na indústria. Também foi possível evidenciar que a maioria de trabalhos estão focados na etapa de operação das RCOs, ao contrário da proposta desta dissertação que considera todas as 4 etapas do ciclo de vida. Optou-se também por adotar uma abordagem de mapeamento semiautomático, que é um tema pouco abordado nos trabalhos relacionados. O fato de ser semi e não completamente automático se deve a que o resultado desse processo é de vital importância na integração, não podendo haver margem para erros computacionais. Assim, a estratégia adotada foi de se requerer uma pessoa encarregada para essa atividade, verificando e aprovando o resultado do mapeamento antes que de fato os sistemas dos membros da EV passem a trocar informação entre si.

Finalmente, como resultado da comparação dos trabalhos relacionados, se evidenciou que a interoperabilidade estrutural é um aspecto igualmente pouco abordado. Porém, considerando que a interoperabilidade estrutural traz alguns complexos desafios em termos de alinhamento de esquemas, integração de bancos de dados, entre outros, optou-se por manter o escopo do trabalho na interoperabilidade sintática e semântica.

Vários trabalhos apresentaram abordagens importantes, como em Khalfallah et al (2016), que propuseram um modelo em duas fases. Nesta dissertação, por exemplo, se pretende estender isso adicionando uma fase para a dissolução das EVs. Também nesta dissertação se considera a proposta de criação automática de ontologias como um facilitador do processo de integração. Além disso, a arquitetura de ontologias híbrida descrita em Abadi et al (2016) fornece um bom desempenho e se adapta bem ao cenário pretendido nesta dissertação. Essas contribuições serviram para dirigir a pesquisa e auxiliaram em algumas considerações do projeto da proposta que será detalhada a seguir.

5 MODELO DE INTEROPERAÇÃO

Fundamentado na contextualização feita nas seções anteriores, este capítulo apresenta o artefato principal da dissertação, representado por um modelo que suporte interoperabilidade sintática e semântica em um ambiente de EVs, permitindo que as empresas-membro possam entrar, participar na execução de processos de negócio e sair de uma EV.

O capítulo é dividido em 4 partes principais:

A primeira parte (seção 5.1) retoma de forma sucinta a definição do problema e apresenta de forma geral o modelo proposto nesta dissertação.

A segunda parte (seção 5.2) detalha os requisitos do modelo identificados como resultado do processo de levantamento de informações apresentado nos capítulos 3 e 4, incluindo um detalhamento dos atores e casos de uso do modelo.

A terceira parte (seção 5.3) descreve a arquitetura geral do modelo proposto.

Finalmente, a quarta parte (seção 5.4) apresenta de forma detalhada o modelo proposto.

5.1 INTRODUÇÃO

Como foi apresentado nos capítulos 1 e 3, RCOs trazem consigo uma série de vantagens, como o compartilhamento de recursos, de conhecimento, entre outras. Porém, para concretizar de forma efetiva essas colaborações, uma série de questões devem ser resolvidas. Neste trabalho são abordadas especificamente as questões atreladas à integração e interoperação dos diferentes sistemas e processos das empresas envolvidas em uma EV.

O primeiro problema que emerge é o da heterogeneidade das tecnologias (protocolos e formatos de dados) e modelos de dados que cada empresa-membro possui. Como já abordado, atualmente a solução desses problemas requer intervenção da pessoa encarregada da integração (um especialista no assunto) para alinhar os modelos de dados de todos os membros e fazer as adaptações necessárias dos diferentes protocolos, fazendo com que a integração seja um processo lento, propenso a erros e de alto custo financeiro.

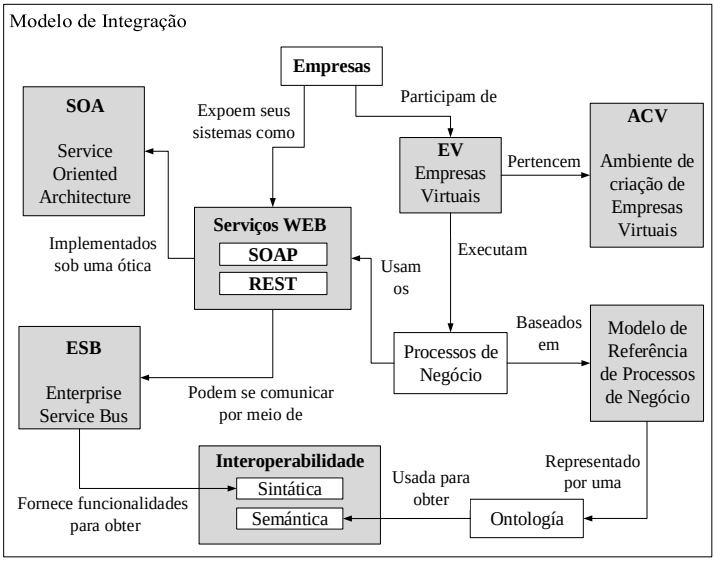
Como alternativa para resolver esse tipo de problema a comunidade industrial e científica tem adotado algumas abordagens que demonstraram uma maior efetividade de solução técnica, pelo menos para parte dos vários problemas envolvidos. Entre elas, podem-se citar

modelos de referência, ontologias, barramento de serviços e arquitetura orientada a serviços (explanadas no capítulo 3), que são as adotadas para o projeto do modelo.

Por outro lado, na medida em que mais empresas fazem parte da colaboração, também aumenta a complexidade de gerenciamento. Portanto, é de vital importância estabelecer um fluxo ordenado de atividades que permitam que todos os membros estejam preparados e disponíveis para atender as oportunidades que surgirem. Nesse sentido, a teoria de RCOs e seu ciclo de vida (ver Figura 3) permitem estabelecer essas atividades, que são adotadas como base para o modelo para proposto em termos do processo geral de integração, tanto na etapa de configuração, quanto na etapa de execução e saída das empresas das EVs das quais participam. A salientar que, nesse contexto, a entrada e saída das empresas de EVs dá-se dinamicamente, e que uma mesma empresa pode participar de várias EVs simultaneamente, i.e., seus sistemas devem ter que interoperar com vários outros diferentes ao mesmo tempo.

Tomando esse cenário acima como referência e os conceitos apresentados no capítulo 3, a Figura 21 evidencia a relação dos temas abordados na fundamentação teórica (blocos cinza) e uma abstração de alto nível do modelo de interoperabilidade pretendido.

Figura 21: Relação das áreas de conhecimento abordadas no trabalho



Fonte: Própria

Na Figura 21 a relação, começa do conceito das empresas que participam em EVs. As EVs (i.e., suas empresas-membro) pertencem a um ACV. Cada empresa-membro do ACV (e portanto qualquer EV que seja criada nele) expõe seus sistemas na colaboração como serviços web, que compõem os processos de negócio (baseados em modelos de referência) executados pelas EVs. O modelo de referência é usado para representar um vocabulário que sirva de base para viabilizar a interoperabilidade semântica. Já a interoperabilidade sintática e o suporte da arquitetura orientada a serviços são fornecidos pelo barramento (ESB), que viabiliza a troca de informação e a comunicação entre os sistemas computacionais de cada empresa-membro das EVs em operação.

5.2 REQUISITOS DO MODELO

Partindo do exposto na fundamentação teórica (Capítulo 3), na seção 5.1 e no Quadro 3 (Capítulo 4) são definidos os seguintes requisitos funcionais para o modelo de integração desejado.

Quadro 4. Requisitos do modelo

ID	Requisito	Descrição
R1	Suportar interoperabilidade sintática	Permitir que sistemas que usam diferentes protocolos de comunicação possam interagir, providenciando os meios para realizar as adaptações necessárias na comunicação.
R2	Suportar interoperabilidade semântica	Fornecer os meios para que as informações trocadas na colaboração sejam entendidas corretamente pelo destinatário sem necessidade de combinar previamente nenhum tipo de regras.
R3	Suportar mapeamentos semânticos semiautomáticos	Suporte aos mecanismos que permitam automatizar a representação e alinhamento das informações dos membros da colaboração.
R4	Ser baseado em orientação a serviços	Seguir as diretrizes da arquitetura orientada a serviços (SOA)
R5	Ser baseado em padrões abertos	Usar padrões abertos na solução
R6	Suportar modelos de referência de processos de negócio	Usar especificações padrão de processos de negócio na definição dos processos colaborativos.
R7	Suportar monitoramento	Permitir monitorar a execução dos processos de negócio da EV.
R8	Suporte ao ciclo de vida de EVs	Deve considerar as etapas do ciclo de vida de colaboração entre empresas: criação, operação, evolução e dissolução.

Fonte: Própria

5.3 ATORES E CASOS DE USO

Os casos de uso basicamente são descrições sucintas que representam um conjunto de ações que um sistema pode executar quando interage com um usuário externo ao sistema, sendo um ponto de partida central na análise de requisitos. Este tipo de artefato apresenta as relações entre os atores (usuários) e as principais funcionalidades que o sistema fornece (LARMAN, 2001).

Foram identificados 4 atores principais para o modelo de integração pretendido. Cada um deles desempenha um papel específico ao longo da colaboração, sendo um coordenador e um membro, tanto para o ACV quanto para cada EV. A seguir são listados e detalhados os casos de uso identificados. O diagrama que expõe as suas relações com os autores é apresentado na Figura 22.

Os casos de uso estão baseados nos requisitos definidos na seção 5.2. Na sequência se detalha cada caso e se estabelece a relação com cada um dos requisitos:

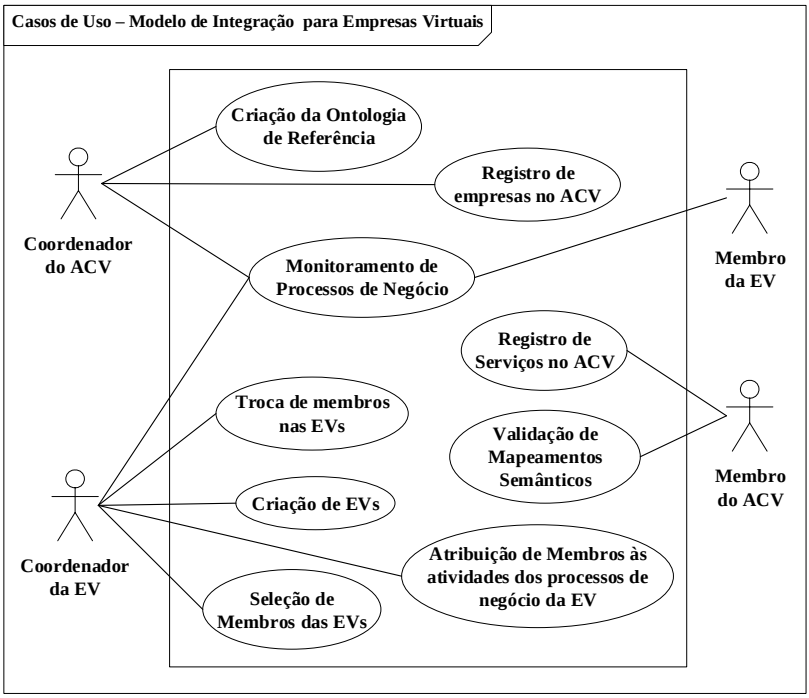
- **Criação da Ontologia de referência:** Esse caso de uso fornece a funcionalidade de criação automática da ontologia que representa o modelo de referência usado na colaboração visando facilitar a utilização de modelos de referência. O usuário que participa neste caso de uso é o coordenador do ACV. Está relacionado aos requisitos R2 e R6.
- **Registro de Empresas no ACV:** Representa o processo de registro das empresas que vão participar no ACV. Esse processo é realizado pelo coordenador do ACV, e está relacionado ao requisito R8.
- **Registro de Serviços no ACV:** Representa o processo de cadastro dos sistemas por parte de cada empresa (membro do ACV) selecionada para participar do ACV. É uma das atividades mais relevantes, já que dessa atividade depende que esses sistemas possam ser depois integrados nas possíveis EVs que sejam criadas dentro do ACV. Além do registro dos sistemas no modelo de interoperabilidade, também realiza o alinhamento do modelo de dados dos sistemas com o modelo de referência. Os requisitos relacionados neste caso de uso são R2, R3 e R4.
- **Validação de Mapeamentos Semânticos:** Tem como objetivo permitir que o membro do ACV que fez o cadastro dos seus sistemas faça uma validação/correção do resultado do

alinhamento dos modelos de dados realizado no caso de uso de *registro de serviços no ACV*. Esse caso de uso atende ao requisito R2.

- **Criação de EVs:** Retrata o processo de criação de EVs onde são definidas as suas informações básicas. Esse processo é realizado pelo coordenador da EV e está relacionado com o requisito R8.
- **Seleção de Membros das EVs:** Fornece a funcionalidade ao coordenador da EV para realizar a seleção entre os diferentes membros já cadastrados que vão participar da EV. Esse caso de uso atende ao requisito R8.
- **Atribuição de Membros às atividades dos processos de negócio da EV:** Esse caso de uso reflete o processo de seleção dos sistemas dos membros da EV que vão participar da execução de um dado processo de negócio. Esse procedimento é realizado pelo coordenador da EV. Esse caso de uso referencia o requisito R8.
- **Troca de membros nas EVs:** Representa a funcionalidade de troca de membros que atuam em um dado processo de negócio de uma EV. Essa funcionalidade pode ser utilizada unicamente pelo coordenador da EV que, segundo o desempenho dos seus membros, pode decidir tirar um membro e substituí-lo por um outro ou desfazer a EV quando ela atinge seu objetivo de criação. Esse caso de uso se relaciona com o requisito R8.
- **Monitoramento de Processos de Negócio:** O último caso de uso identificado representa o monitoramento da operação das diferentes instâncias dos processos de negócio da EV. Permite aos membros da EV e aos coordenadores acompanhar a execução dos processos onde participam. Esse caso reflete a funcionalidade definida no requisito R7.

Na Figura 22 se expõem os 9 casos de uso descritos e suas relações com os 4 atores identificados. Pode-se observar que os coordenadores têm maior número de privilégios no sistema, interagindo em 5 de 9 dos casos de uso definidos.

Figura 22: Diagrama de casos de uso



Fonte: Própria

Com base nas principais funcionalidades apresentadas é detalhada na seguinte seção a arquitetura conceitual geral do modelo.

5.4 DESCRIÇÃO GERAL DA PROPOSTA

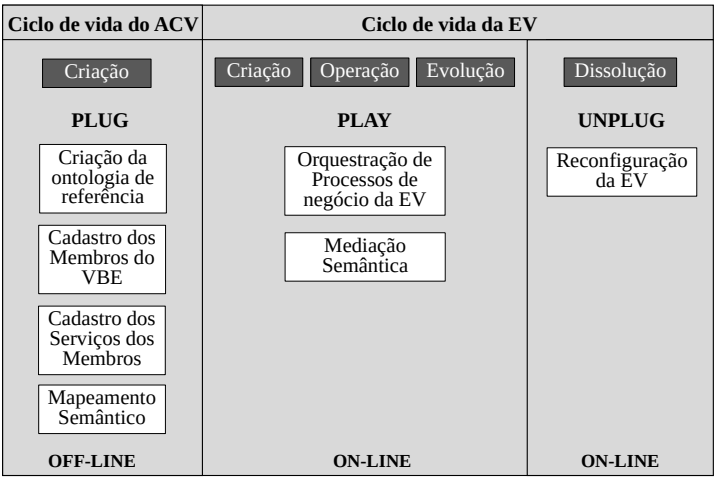
Uma premissa fundamental no modelo proposto é que todos os membros da EV pertencem a uma aliança de longo prazo do tipo ACV. Esse ACV deve ser intrinsecamente fundamentado na confiança, no compartilhamento de recursos, na autonomia, nos princípios de trabalho comuns e na disposição para colaboração dos participantes. Portanto, o modelo não considera etapas específicas de instanciação do ACV como definição de contratos, processos de negócio, busca e seleção de membros, entre outras.

A proposta do modelo é baseada nos critérios de avaliação apresentados no Quadro 3 identificados na revisão do estado da arte, no

ciclo de vida de ACVs e EVs, e nos requisitos definidos no Quadro 4, que condensam as principais tendências sobre a integração de RCOs semelhantes ao ACV e à EV.

Considerando o exposto, de forma sucinta o modelo geral de interoperabilidade para EVs proposto divide conceitualmente o problema e o resolve através de fases principais: *plug-in*, *play* e *unplug*, a seguir descritas. Os nomes dados têm o objetivo de retratar um dos objetivos da proposta, de permitir uma integração ágil das empresas quando da formação de uma EV e, ao mesmo tempo, pontuar que cada uma das fases do ciclo de vida de uma EV requer soluções diferentes, em separado. Esses nomes têm também sido usados cada vez mais em trabalhos na área que visam dar uma noção de algo mais rápido, fácil, etc., quando da integração ou uso de sistemas computacionais, como por exemplo em Rabelo et al, (2008). Uma visão geral do modelo pode ser observada na Figura 23.

Figura 23. Modelo proposto de Interoperabilidade para Empresas Virtuais



Fonte: Própria

- **Fase de plug-in:** fase de preparação executada quando um membro é recrutado por um ACV e seus sistemas devem ser preparados para trabalhar em possíveis colaborações quando as EVs forem criadas. Nessa fase são executados 4 casos de uso: a criação automática da ontologia de referência, o cadastro de

empresas no ACV, o cadastro dos sistemas das empresas membro do ACV e a validação de mapeamentos semânticos.

- **Fase de play:** fase em que os membros que já pertencem ao ACV são selecionados para participar das EVs. É nessa etapa que as empresas transacionam entre si, ou seja, onde seus sistemas interoperam para executar os processos de negócio da EV com base nas configurações e integrações incialmente feitas na fase de plug-in. Considerando o framework genérico de referência para EVs proposto em Camarinha-Matos, Afsarmanesh e Ollus (2005), essa fase pressupõe que algumas das etapas iniciais do processo geral de operação de EVs já tenham sido realizadas, como o planejamento de EV, a busca e seleção de membros, a negociação e a contratação legal final. Essa fase abrange os casos de uso correspondentes a criação de EVs, seleção de membros das EVs, atribuição de membros às atividades dos processos de negócio, troca de membros e monitoramento.

- **Fase de Unplug:** fase em que os sistemas dos membros de uma EV são logicamente desconectados quando finalizam a operação ou quando um determinado membro deixa a colaboração. O caso de uso relacionado nessa fase é a troca de membros nas EVs.

Cada uma dessas três fases será detalhada nas seções a seguir.

Considerando as atividades da metodologia de instanciação do ACV apresentada por Romero, Galeano e Molina, (2008), o modelo proposto executa a fase de "*plug-in"* na etapa de criação do ACV especificamente nas atividades de Configuração das TICs e seleção de membros.

Já a fase *"play"* é executada quando a EV é criada (etapa de criação da EV) e quando está em operação (etapas de operação e evolução da EV). A fase "*unplug"* é executada nas etapas de evolução e dissolução.

Importante ressaltar que, uma vez que uma empresa (seus sistemas) se "plugar" na EV, todo processo de operação e dissolução se dá dinamicamente e de forma transparente (abstraindo detalhes técnicos), sem interferência alguma de usuários ou de mapeamentos adicionais. Além disso, o fato de que uma mesma empresa poderá estar envolvida em várias EVs ao mesmo tempo, significa que ela (seus sistemas) estará permanentemente em diferentes processos de "*plugagem*" e "*desplugagem*", em diferentes momentos.

5.5 O MODELO DE INTEROPERAÇÃO PARA EVs

O modelo se baseia em duas abordagens básicas em relação a seus princípios e objetivos de projeto: i) o uso de modelos de referência para modelagem de processos de negócio e ii) orientação a serviços (SOA).

Neste trabalho, por ter uma perspectiva também de prova de conceito, optou-se como modelo de referência de processos de negócio o padrão UBL suportado pela OASIS (OASIS, 2018). Do ponto de vista técnico, ele foi adotado principalmente porque, como descrito na seção 3.2, dentre os modelos de referência consultados ele é o mais atualizado. Tem um catálogo de processos de negócio bastante completo (68 processos) e detalha cada processo com as suas atividades, documentos e as estruturas de dados a serem trocadas quando executados. Além do fato de ser um padrão aberto, o UBL foi concebido primordialmente para um tipo de RCOs muito semelhante com EVs do ponto de vista funcional, que são as cadeias de suprimentos (*supply chains*). De forma bastante geral, a diferença básica entre EVs e Cadeias de Suprimento é que nestas a rede de empresas é fixa, pré-determinada, com modelos de dados e terminologias comuns e previamente acordadas, e usualmente há uma empresa grande que determina padrões e práticas de trabalho e processos. No caso de EVs, esses aspectos se dão praticamente todos "em tempo de execução" dado que as parcerias se estabelecem dinamicamente quando um dado negócio é estabelecido, e onde cada empresa é completamente independente (incluindo em termos de TI) e diferente uma das outras.

Sob a estratégia de ser um modelo o menos intrusivo possível, a abordagem adotada é de não fazer com que todos os membros do ACV tenham que adotar a UBL nos seus sistemas internos para participar de EVs. A especificação UBL é usada apenas internamente na arquitetura do modelo como um meta-modelo, representando uma linguagem genérica e intermediária (usada como "referência"), onde todas as mensagens trocadas nas transações entre os diferentes membros das EVs são transformadas por meio de um processo de *mediação*.

De qualquer forma, o modelo de interoperabilidade foi construído para ser aberto para trabalhar com qualquer modelo de referência de processos de negócio, seja padrão ou proprietário. Cabe à organização do ACV, ou mesmo da EV, decidir sobre qual modelo de processos de negócio deve ser adotado.

Como apresentado na seção 3.5, a natureza distribuída de SOA traz vantagens em relação à flexibilidade, baixo acoplamento e independência de tecnologia de implementação, vantagens que se alinham com as características das EVs e dos requisitos identificados. Por essa razão,

SOA foi escolhida como arquitetura de suporte para o projeto e implementação do modelo de interoperabilidade proposto.

A escolha de SOA impõe uma primeira restrição nos sistemas que serão aceitos pelo modelo de interoperabilidade. Seguindo os princípios de SOA, cada sistema que seja incluso na colaboração deve ser encapsulado como um serviço (podendo ser implementado com diferentes granularidades, tecnologias e arquiteturas) e deve ter a sua descrição correspondente. Essa descrição é um artefato importante no modelo de interoperabilidade desejado, já que é a forma de descrever o modelo de dados usado por cada sistema disponibilizado pelos membros.

Da mesma forma, o fato de se adotar SOA não implica que os sistemas dos membros do ACV precisam adotar todos a mesma tecnologia de implementação, pois a arquitetura do modelo é aberta para suportar várias tecnologias. Essa flexibilidade é obtida pelo uso de outra tecnologia, o ESB, que atua como uma infraestrutura de computação que fornece comunicação segura e confiável envolvendo ambientes heterogêneos e de acoplamento flexível.

Contudo, os sistemas ESB não são nativamente preparados para lidar com problemas de interoperabilidade semântica. Como apresentado na seção 3.3, uma alternativa considerada como a abordagem mais adequada para superar essa deficiência tem sido as ontologias. No caso do modelo de interoperabilidade proposto e o cenário de EV desejado, é adotada a arquitetura de ontologia híbrida, apresentada na Figura 10, onde é definida uma ontologia global usada como referência e múltiplas ontologias locais que alinham seu modelo de dados contra a referência.

O modelo proposto suporta a criação automática das ontologias citadas usando a linguagem da web semântica (padrão) OWL (*Web Ontology Language*) e contempla o mapeamento automático delas. Tais processos são descritos em maior detalhe a seguir.

5.5.1 A fase de Plug-in

Esta é uma fase inicial de configuração, portanto é realizada *off-line*. Está principalmente relacionada com a preparação dos sistemas de TI (e.g. ERP) das empresas para elas se habilitarem a serem membros de um ACV. Isto tem a ver com um dos requisitos básicos de pertencimento a um ACV, que é o de "*IT preparedness*" (CAMARINHA-MATOS; AFSARMANESH, 2005).

O objetivo principal da fase é preparar a infraestrutura de suporte para a integração dos diferentes modelos de dados (de referência e de cada membro) e tornar os membros do ACV "prontos para participar" em EVs.

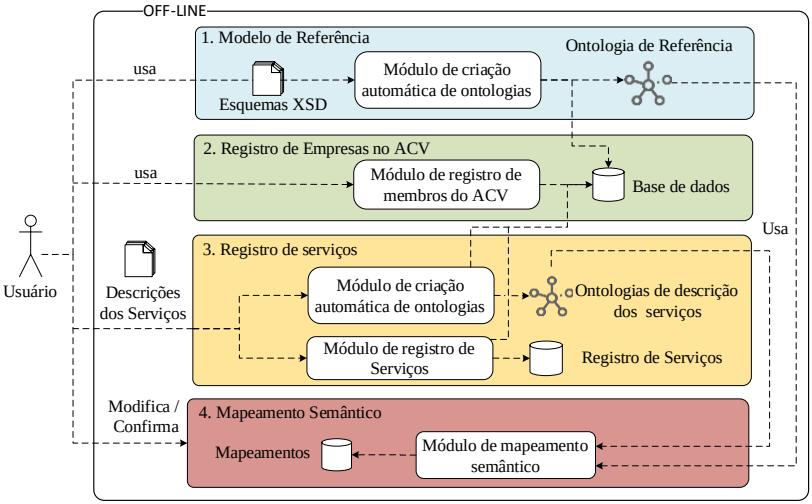
Para atingir esse objetivo essa fase é dividida em 4 subfases sequenciais: a criação do modelo de referência, o registro de empresas-membro no ACV, o cadastro dos serviços de cada membro do ACV e o alinhamento entre os modelos de dados dos serviços registrados do membro e o modelo de referência. A Figura 24 expõe o fluxo lógico das subfases realizadas nesta fase.

Figura 24: Fluxo de atividades principais na etapa de plug-in

Fonte: Própria

A Figura 25 detalha cada uma dessas subfases, apresentando os módulos que nelas são utilizados, suas entradas/saídas e suas interligações.

Figura 25: Etapas da fase de Plug-in



Fonte: Própria

5.5.1.1 Criação do Modelo de referência

Esta primeira etapa da subfase consiste na construção da ontologia global baseada no modelo de referência de processos de negócio (bloco 1

na Figura 25), essa etapa é executada uma vez só na configuração da comunicação.

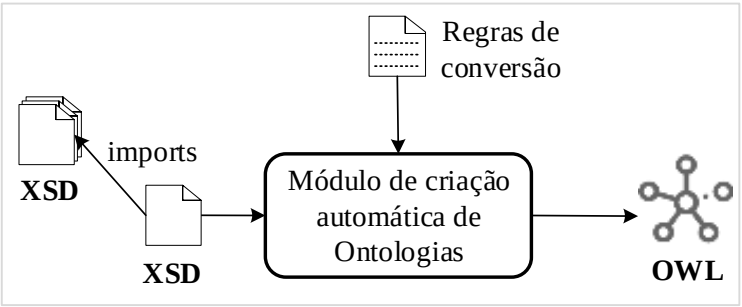
Como apresentado na seção 3.4.2.1, o processo de criação de uma ontologia costuma ser complexo, demorado e normalmente requer de um especialista no domínio com amplo conhecimento em modelagem de ontologias. Além disso, levando em consideração o apresentado em Bedini et al. (2011), XML e XSD tem sido usado amplamente para representação de dados na Web e também tem uma grande popularidade na representação de especificações B2B (*Business to Business*), como por exemplo os apresentados na seção 3.2, *ebXML*, *RossetaNet* e *UBL*.

Partindo desses dois pontos, e com o objetivo de facilitar esse processo de criação da ontologia de referência, se propõe um módulo que forneça a criação automática de ontologias.

Esta atividade é suportada pelo módulo de *Criação Automática de Ontologias*, que adota a técnica de criação automática de ontologias por meio de *conversão* (ver seção 3.4.2.2). O módulo recebe como entrada os esquemas XSD que definem o modelo de referência de processo de negócio escolhido e retorna a ontologia OWL equivalente. Este módulo foi projetado para ser usado pelo coordenador do ACV, e o artefato gerado (a ontologia global) como resultado desse processo será usado como modelo de dados global internamente no ACV.

A transformação dos esquemas XSD para ontologia OWL é realizada seguindo a técnica de *conversão* (ou *tradução*) descrita na seção 3.4.2.1. É utilizado um conjunto de regras de conversão que são aplicadas a cada elemento do esquema XSD de entrada. Como resultado, o processo retorna os elementos equivalentes da ontologia em formato OWL. A Figura 26 mostra conceitualmente como é esse processo.

Figura 26: Diagrama do processo de criação automática de ontologias



Fonte: Própria

5.5.1.2 Registro de empresas no ACV

A segunda subfase corresponde ao cadastro das empresas no ACV (Bloco 2). A sua principal função é capturar e salvar na infraestrutura da colaboração as informações básicas das empresas que vão participar do ACV, possibilitando a disponibilização dos seus sistemas e sua participação nas possíveis EVs.

5.5.1.3 Representação semântica e registro de serviços

Uma vez que a ontologia global é definida e os membros do ACV são cadastrados, o próximo passo é registrar os serviços (sistemas) de cada membro que vão ser disponibilizados para a integração, no *registro de serviços* central presente no modelo proposto.

Essa atividade é suportada pelo *módulo de registro de serviços*, que lê as descrições dos serviços disponíveis e as usa para registrá-los. Como se pode observar no diagrama de casos de uso definido na Figura 22, essa atividade é realizada pela própria empresa-membro, que é quem de fato conhece seus sistemas e sua infraestrutura interna.

Ainda nesta subfase, uma vez registrados os serviços, é utilizado o *módulo de criação automática de ontologias* (descrito anteriormente) para criar uma ontologia de cada serviço registrado pelos membros. Isso visa criar uma representação do modelo de dados usado por cada serviço e assim conseguir alinhá-lo com a ontologia de referência criada na primeira subfase. Esse processo descrito é representado na Figura 25 no bloco 3.

5.5.1.4 Mapeamento semântico de serviços

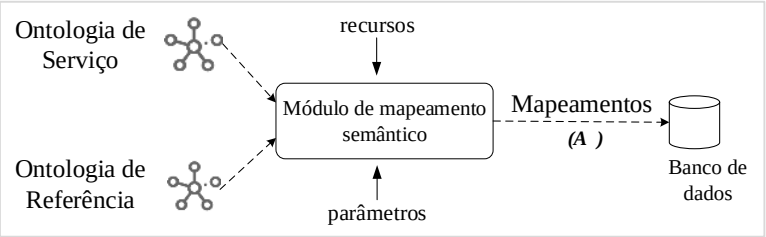
A última subfase refere-se a usar as ontologias criadas de cada serviço para realizar um processo de mapeamento em relação à ontologia global do modelo de referência. Isso é suportado pelo *módulo de mapeamento semântico*. Esta topologia, que combina uma ontologia central e as muitas outras representando os serviços dos membros, considera os benefícios de desempenho no processo de mediação, segundo apontado em Abadi et al. (2016).

Esse processo de mapeamento é de vital importância no modelo proposto porque dos resultados dele depende que a comunicação entre membros que usam diferentes modelos de dados seja efetuada corretamente. Assim, a ontologia de referência serve como linguagem intermediária, onde cada mensagem dos membros é transformada usando

os alinhamentos gerados entre sua ontologia e a de referência. Depois, antes da mensagem ser entregue ao destinatário, ocorre o processo inverso.

Como se pode observar na Figura 25, e seguindo as técnicas de alinhamento de ontologias descritas na seção 3.4.2.2, o *módulo de mapeamento semântico* recebe como entrada a ontologia de um serviço, gerada na etapa anterior, e a ontologia de referência gerada na primeira etapa, estabelecendo-se um alinhamento que contém as correspondências que relacionam seus conceitos. Esse processo é ilustrado na Figura 27.

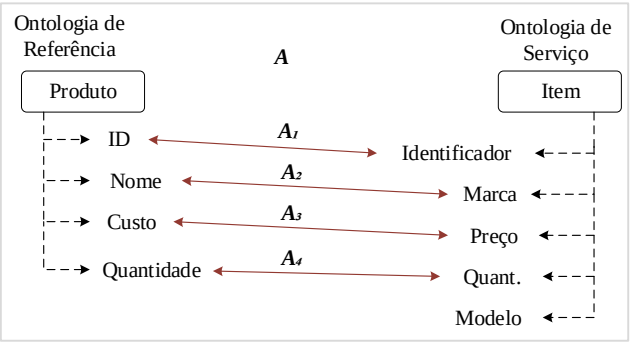
Figura 27. Diagrama do processo de mapeamento de ontologias



Fonte: Própria

A Figura 28 ilustra o mapeamento (*A*) resultado do processo de alinhamento de ontologias mostrado na Figura 27. Se observam dois trechos de cada ontologia e as correspondências semânticas (setas contínuas) entre as entidades de cada ontologia, indicando que podem ser consideradas como entidades equivalentes.

Figura 28. Exemplo do alinhamento entre duas ontologias



Fonte: Própria

Ao finalizar as quatro subfases de *Plug-in*, todas as empresas-membro do ACV e seus sistemas estarão corretamente cadastrados na infraestrutura da integração e terão os seus modelos de dados proprietários corretamente alinhados semanticamente com o modelo de dados de referência.

5.5.2 A fase de Play

É uma fase *on-line* (em tempo de execução), quando uma empresa é selecionada para fazer parte de uma EV e, assim, a interagir com as demais empresas-membro. Esta fase oferece o suporte necessário a cada empresa membro (seus sistemas) para obter integração e interoperabilidade no ambiente de computação do ACV, e portanto, nas EVs que dentro dele surgirem.

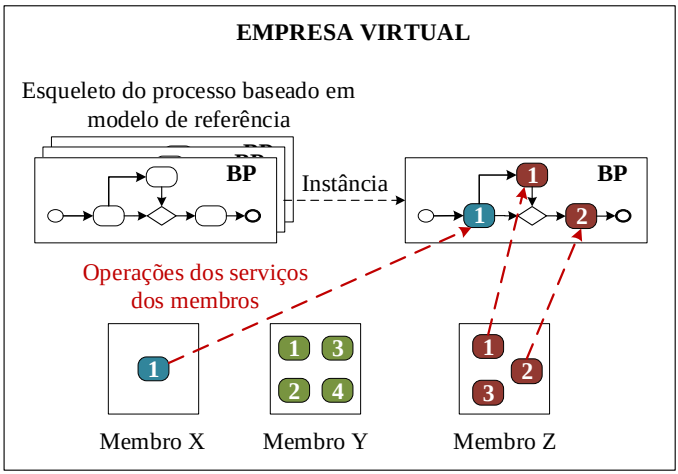
É importante destacar que muitas EVs podem ser criadas simultaneamente e que uma determinada empresa (ou seja, seus sistemas) pode pertencer a várias EVs ao mesmo tempo.

Considerando o ciclo de vida de uma EV (Figura 3), essa fase é executada em dois eventos principais: quando um membro é selecionado para participar em uma EV (etapa de *criação* no ciclo de vida da EV) e quando as empresas executam seus processos de negócio distribuídos entre seus membros (etapa de *operação* e de *evolução*).

Uma das atividades principais na criação de uma EV é definir o que ela vai fazer e como isso será feito. Nessa etapa são estabelecidos os processos de negócio que estarão disponíveis na colaboração e os membros que serão participantes desses processos. Segundo as etapas definidas em Camarinha-Matos, Afsarmanesh e Ollus (2005) essa atividade corresponde ao *planejamento detalhado de organizações virtuais*.

Para representar os processos de negócio, são utilizados "esqueletos" cuja estrutura é modelada seguindo o modelo de referência selecionado. Esses esqueletos servem como base para criar instâncias específicas associando os membros responsáveis por executar as atividades do processo. A Figura 29 ilustra isso, como exemplo, na figura se representa uma EV composta por 3 membros, cada um deles com diferente número de serviços. Além disso, são mostrados os esqueletos dos processos de negócio disponíveis na EV e o procedimento de como um processo é instanciado selecionando operações dos serviços dos membros para executar atividades especificas dele. É essa instância criada que finalmente é executada na operação da EV.

Figura 29: Associação de membros às atividades do processo de negócio



Fonte: Própria

Na sequência são apresentados os 2 módulos principais que suportam a integração entre os sistemas dos membros de uma EV:

5.5.2.1 Módulo de orquestração de serviços:

Os processos de negócio normalmente são modelados por analistas administrativos que usam linguagens conceituais de alto nível, definindo as atividades e o fluxo que deve seguir a informação segundo as condições próprias na lógica do processo. Como explicado anteriormente, foi adotada a especificação UBL que já define o fluxo, atividades e estruturas de dados dos processos de negócio.

Em outras palavras, UBL é apenas uma especificação textual. Ela pode ser modelada em formato digital, e diferentes linguagens ou formalismos podem ser usados para uma representação gráfica de um processo UBL numa empresa; por exemplo, em BPMN.

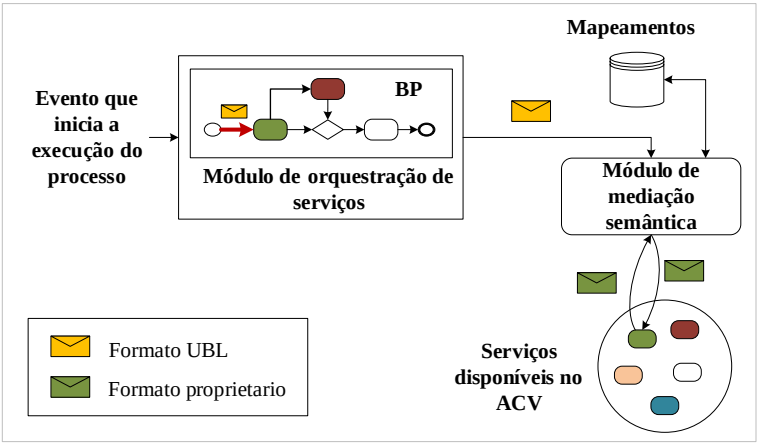
No nível de implementação, é necessário garantir que essas atividades do processo de negócio sejam executadas como foram planejadas. Portanto, nesta proposta se adota a abordagem de orquestração de serviços. É justamente isso que é o *módulo de orquestração*: um artefato central que garante que os processos de negócio sejam executados respeitando a ordem das suas atividades

conforme foram modelados (Figura 30). Nesta fase adota-se como abordagem representar esses processos de negócio por meio de um modelo equivalente ao modelo de alto nível, mas que permita detalhar informações relacionadas à implementação e tecnologia dos sistemas que irão ser invocados (os serviços dos membros das diferentes EVs).

5.5.2.2 Módulo de mediação semântica

Considerando que cada serviço tem o seu próprio modelo de dados e que normalmente há divergências entre eles, para conseguir que esses sistemas troquem informações corretamente é necessário que seja realizada uma transformação dessa informação. É aqui onde os alinhamentos feitos entre o modelo proprietário de cada serviço com a ontologia de referência (resultado da fase de *plug-in*) ganham relevância. Esses alinhamentos e as informações trocadas pelos serviços são as entradas de informação do módulo de mediação que é encarregado de realizar a transformação dessa informação. Esse processo é ilustrado na Figura 30.

Figura 30. Processo de mediação de mensagens



Fonte: Própria

A Figura 30 mostra como é o processo de mediação proposto no modelo. O processo de negócio pode ser iniciado por um evento externo; por exemplo, um click em um portal web, uma transação em um ERP,

entre outros. Pode-se observar na figura uma invocação (seta vermelha) feita à primeira atividade do processo de negócio, que irá ser executada por um dos serviços disponíveis no ACV; no caso o serviço com cor verde.

A mensagem inicial (em amarelo) enviada na invocação está formatada em UBL. Portanto, antes dela chegar no serviço destinatário, é interceptada pelo *módulo de mediação*, que busca os mapeamentos gerados na fase de *plug-in*. Uma vez obtidos os mapeamentos, pode-se transformar a mensagem na representação equivalente (em verde), mas nos termos do serviço destinatário, possibilitando assim uma correta interpretação.

O processo inverso é realizado com a reposta do serviço. Ela deve ser transformada para o formato UBL, já que é essa representação que serve como "língua" intermediária entre serviços. Desta forma, na seguinte invocação a um serviço diferente é possível realizar o mesmo processo de conversão.

Todo esse processo de comunicação ente módulos e serviços precisa de um suporte tecnológico que garanta que todas as mensagens que são enviadas cheguem ao seu destinatário de uma forma segura e confiável. O modelo proposto fornece esse suporte adotando o uso da tecnologia de barramento de serviços empresarias ESB, apresentada na secção 3.6.

5.5.3 A fase de Unplug

Esta também é uma fase *on-line* e é executada em duas situações: quando um membro de uma dada EV finaliza todas suas funções e sai naturalmente da EV (na etapa de dissolução); e quando um dado membro da EV apresenta algum tipo de problema que impossibilita o cumprimento das suas obrigações e deve deixar a EV após uma análise geral (durante a etapa de operação) por parte do coordenador da EV. O principal artefato que suporta esta atividade é o módulo de reconfiguração de EV, descrito a seguir.

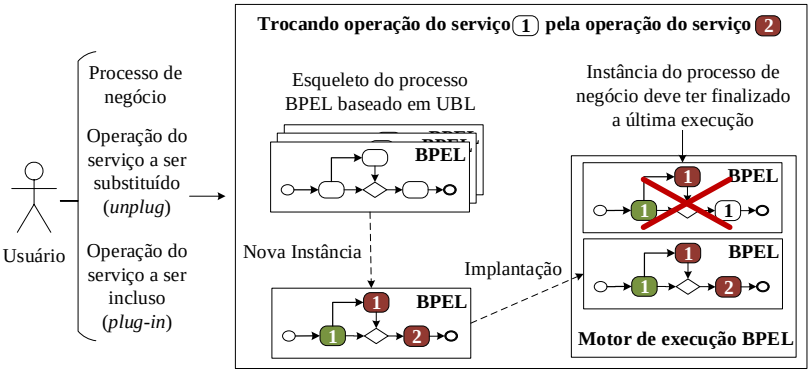
5.5.3.1 Módulo de reconfiguração da EV:

Este módulo tem como objetivo suportar as condições de troca de membros ou dissolução de uma EV. É também uma fase *on-line*. Porém, ele é executado apenas na primeira situação, quando um membro de uma EV deve ser forçosamente retirado dela e seus sistemas devem ser "desplugados". Nessa situação, o coordenador da EV deve atualizar o

planejamento da mesma e, conforme necessário, buscar novo(s) parceiro(s) para substituir o membro que está saindo.

Considerando a complexidade legal de saída de uma empresa de um negócio e de alguns aspectos tecnológicos associados à tecnologia de web services (como o de não preservação de estado), neste trabalho foram feitas algumas simplificações. É proposta uma alternativa simplificada de substituição de serviços, suportando unicamente a substituição de um serviço por apenas um outro que esteja devidamente cadastrado seguindo as etapas apresentadas na fase de *plug-in*. Essa mudança é efetivada quando o processo de negócio concluir todas suas atividades e os serviços do membro a sair não tenham mais processos dependentes. O modelo provê reconfiguração automática do processo de negócio que é afetado pela mudança e a desconexão lógica dos serviços antigos do membro a sair. O diagrama conceitual desse processo é apresentado na Figura 31.

Figura 31: Processo de Unplug



Fonte: Própria

A Figura 31 ilustra o processo de *unplug*. Todos os processos de negócio que são executados na EV tem o seu "esqueleto" disponível para gerar instâncias com invocações específicas às operações dos serviços que o compõem. No caso do exemplo, apresenta-se um processo de negócio que está sendo executado pelo *módulo orquestrador de serviços*. Esse processo está composto por três serviços representados por diferentes cores (verde, vermelha e branca), cada um deles executando a operação número 1. Supõe-se que o serviço "branco" não pode seguir participando de futuras execuções e deve ser trocado, fazendo com que seja necessário esperar a finalização das suas atividades nos processos

onde o serviço participa. Uma vez que o serviço esteja sem responsabilidades, é criada automaticamente uma nova instância do processo com a nova operação do participante (serviço vermelho, operação 2). Finalmente o novo processo é implantado no orquestrador.

Cabe esclarecer que a substituição de serviços é suportada unicamente para serviços que tenham sido devidamente cadastrados e cujos modelos de dados tenham sido alinhados com a ontologia de referência, segundo o apresentado na fase de *plug-in*.

5.5.4 Fluxo das mensagens

Para finalizar essa seção é apresentada a Figura 33, que expõe todos os módulos da fase de *play* e *unplug* (destacados em cinza) em conjunto, oferecendo um panorama geral das interligações deles e um exemplo do fluxo das informações através dos diferentes artefatos.

Nesta fase o fluxo de informação começa quando um cliente cria um pedido por meio do portal Web onde a EV oferece seus produtos/serviços. Essa demanda é atendida dentro da EV executando um processo de negócio que foi previamente modelado e instanciado com os membros participantes na etapa de criação da EV. A execução desse processo de negócio é gerida pelo *módulo orquestrador de serviços*. Todas as mensagens que são trocadas entre os membros das EVs são suportadas pelo ESB. Para tal é exposto um ponto de entrada de integração (*Proxy*) o qual é utilizado pelo orquestrador para se comunicar com os destinatários. Essas mensagens enviadas pelo orquestrador contêm quatro informações-chaves para o processo de interoperabilidade entre serviços:

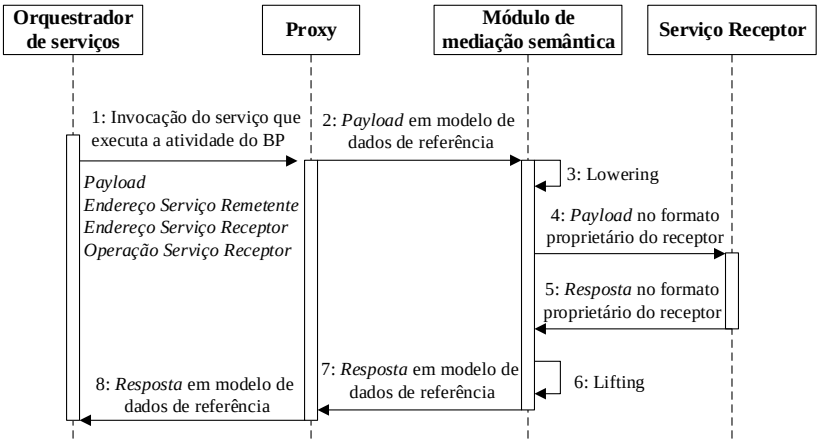
1. Os dados da mensagem ou *payload* (informações enviadas);
2. O endereço da descrição do serviço remetente;
3. O endereço da descrição do serviço receptor;
4. A operação a ser invocada no serviço receptor.

Quando o *proxy* recebe uma mensagem enviada pelo *orquestrador*, ele faz um redirecionamento para o *módulo de mediação semântica*, que usa os endereços das descrições enviadas na mensagem para consultar no banco de dados os mapeamentos necessários para transformar a mensagem. O processo de mediação está dividido em duas operações principais, que na literatura são comumente chamadas de *Lowering* e *Lifting* (FURDÍK et al., 2011; ZHU, 2012).

Lowering corresponde ao processo encarregado de transformar uma mensagem que está no modelo de dados de referência usado no ACV para o modelo de dados de um serviço particular. Esse processo é importante já que o orquestrador de serviços usa o modelo de dados de referência entre as invocações que ele faz às operações dos serviços que compõem o processo de negócio. Portanto, para que essa requisição seja interpretada corretamente pelo serviço receptor (membro da EV), ela precisa estar formatada no modelo de dados específico desse serviço.

O processo de *Lifting* realiza a operação inversa, transformando de volta a mensagem do formato particular do serviço ao formato de referência do ACV. Na Figura 32 pode-se observar o processo de troca de mensagens descrito anteriormente.

Figura 32: Diagrama de sequência da transformação da mensagem

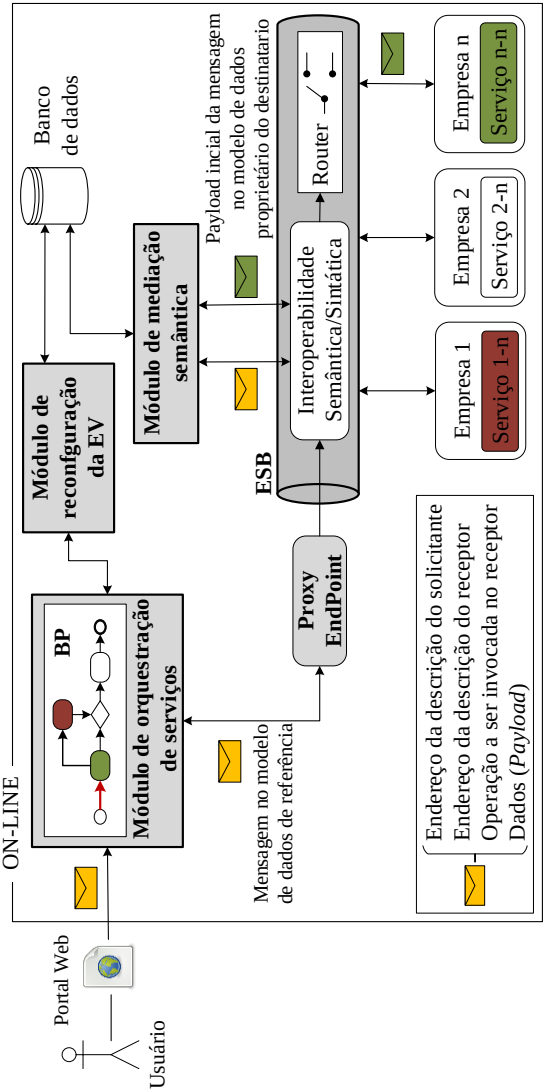


Fonte: Própria

Uma vez que os mapeamentos das informações na mensagem forem resolvidos, as transformações de protocolo/formato necessárias são gerenciadas pelo ESB. O ESB usa o endereço da descrição do serviço e o nome da operação recebidos para invocar o serviço desejado, usando o padrão de integração para roteamento de mensagens (representado como *Router* na figura), concluindo assim a comunicação entre os membros da EV e suportando tanto interoperabilidade sintática quanto semântica.

No caso em que no meio da operação da EV seja necessário trocar uma empresa-membro da EV de um dado processo de negócio, a EV deve ser reconfigurada, entrando na etapa de *evolução*.

Essa troca de serviços dos membros é gerenciada pelo *módulo de reconfiguração da EV*, que é responsável por instanciar e implantar um novo processo com os novos membros selecionados pelo coordenador da EV.



Fonte: Própria

Figura 33: Etapas da fase de Play

5.5.5 Exemplo de funcionamento

Para uma melhor compreensão da interação entre os módulos apresentados, nesta secção é mostrado um exemplo do funcionamento.

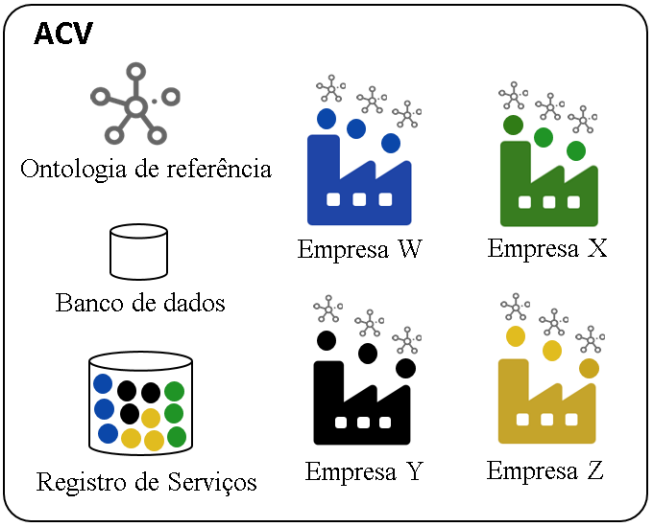
Para o exemplo são consideradas 4 empresas identificadas pelas letras (W, X, Y, Z) que são membros do ACV, cada uma delas disponibiliza três serviços que executam atividades relacionadas à recepção de pedidos. O processo de negócio de pedidos por definição têm três atividades lineares (recepção, validação e confirmação).

Como apresentado na secção 5.5.1 a primeira fase consiste na execução das atividades de configuração necessárias para uma colaboração entre empresas, isso consiste basicamente de três tarefas principais.

- Criação da ontologia de referência
- Cadastro de empresas e serviços no ACV
- Registro e mapeamento de serviços com a ontologia global

A Figura 34 representa o estado do ACV uma vez executada a etapa de plug-in.

Figura 34. Estado do ACV depois da fase de plug-in

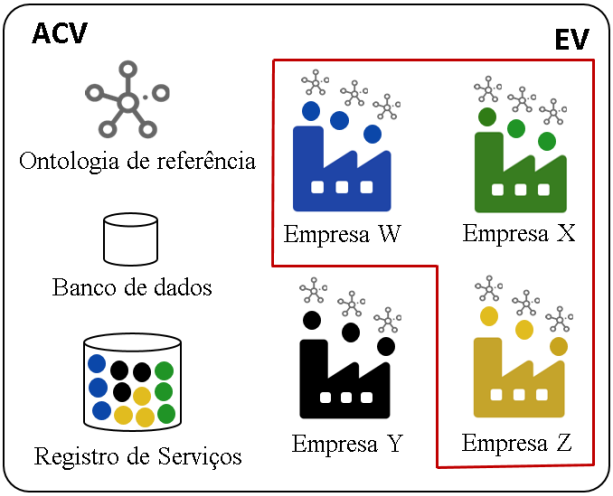


Fonte: Própria

Na Figura 34 se observam os itens adicionados pela execução da fase de plug-in, começando pela ontologia de referência baseada em um modelo padrão de processos de negócio, as empresas com suas informações cadastradas, os serviços disponibilizados pelas empresas já registrados (tanto no registro de serviços central quanto com a ontologia que os descreve) e alinhados com a ontologia de referência, e finalmente o banco de dados que armazena todas as informações relevantes do processo.

Nesse estado as empresas estão prontas para criar/participar de empresas virtuais. Dito processo ocorre na fase de *play* descrita na secção 5.5.2. A Figura 35 representa a criação de uma EV composta pelas empresas W, X e Y para atender uma oportunidade de negócio.

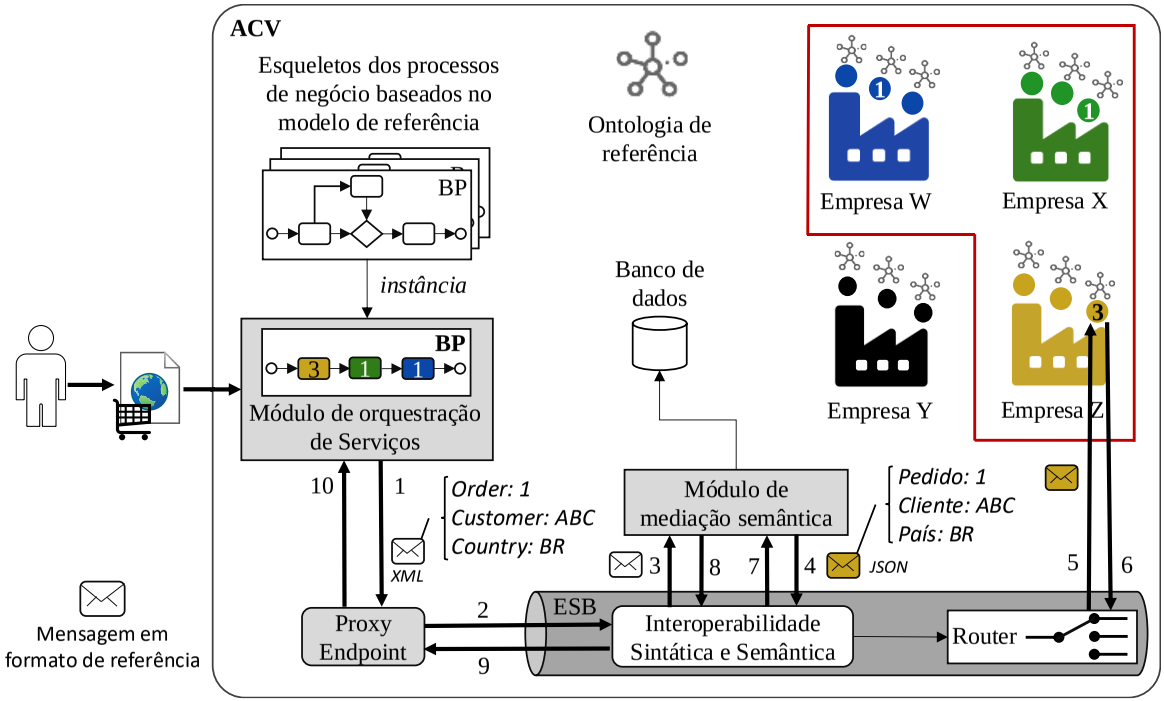
Figura 35. Exemplo de EV criada no ACV



Fonte: Própria

Nesse ponto a EV está pronta para receber pedidos de eventuais clientes que vierem para adquirir produtos ou serviços. A Figura 36 justamente representa o fluxo de mensagens entre os módulos que suportam a funcionalidade na fase de *play*. Especificamente representa o fluxo da informação desde que um cliente realiza um pedido até a execução da primeira atividade do processo de negócio (bloco amarelo dentro do *módulo de orquestração de serviços*).

Figura 36. Exemplo do fluxo de mensagens no processo de negócio de pedidos



Fonte: Própria

O fluxo começa com o cliente acessando o site de venda oferecido pela EV (criado em fases previas não consideradas no escopo deste trabalho). O pedido emitido pelo cliente no site é enviado ao *módulo de orquestração de serviços* que executa a primeira atividade do processo de negócio (recepção) representada pelo bloco amarelo e alocada para ser executada pelo *serviço 3* da *empresa Z*. Como o processo de negócio é baseado no modelo de referência adotado pelo ACV, as mensagens trocadas entre suas atividades também são padrões (representadas na figura na cor branca) e utilizam o modelo de dados definido no padrão, isto é, termos como, *Order*, *Customer*, *Country*, entre outros. Além disso, para efeitos do exemplo essa mensagem padrão é representada em XML.

Como pode-se evidenciar na Figura 36 o orquestrador de serviços não realiza uma invocação direta do serviço que irá executar essa atividade, ao invés disso ele invoca (seta número 1) o *endpoint* de entrada da integração que age como *Proxy* e encaminha a mensagem (seta número 2) ainda usando o formato de referência para o *ESB*, esse último, é encarregado de realizar as primeiras conversões de formato considerando o serviço destinatário, na sequência o *ESB* repassa a mensagem (seta número 3) para o *módulo de mediação semântica* que é o encarregado de "traduzir" essa mensagem para o formato e modelo de dados proprietário (mensagem amarela) do serviço destinatário, utilizando os mapeamentos semânticos gerados na fase de *plug-in* entre o serviço e a ontologia de referência. Pode-se observar na figura que os termos da mensagem original *Order*, *Customer* e *Country* foram mapeados com os termos *Pedido*, *Cliente* e *País*, também o formato da mensagem original foi trocado de XML para JSON que é o formato utilizado pelo serviço destinatário.

Assim que finaliza a "tradução" da mensagem ela é encaminhada (seta número 4) de novo para o *ESB* que finalmente gerencia a invocação final ao serviço destinatário (seta número 5).

O serviço destinatário recebe a requisição (no seu formato proprietário), realiza seu trabalho e retorna ao *ESB* (seta número 6) a informação resultante da execução da operação invocada. Como a mensagem retornada pelo serviço está no seu formato e modelo de dados proprietário é necessário realizar de novo a "tradução" para o modelo de referência (mensagem branca), essa mensagem é repassada para o *módulo de mediação* (seta número 7) que retorna a formata e transforma usando o modelo de dados de referência (seta número 8), finalmente o ESB a retorna ao Proxy (seta número 9) e da mesma forma ele repassa a informação (seta número 10) para o orquestrador de serviços que a utiliza

para seguir executando as duas atividades faltantes do processo de negócio.

O processo descrito anteriormente é executado para cada uma das atividades do processo de negócio até finalmente entregar a resposta do pedido para o cliente.

Neste capítulo, foram apresentados os conceitos principais e modelos gerais que fazem parte da proposta de interoperação para EVs, que engloba também um processo de integração dinâmica. No capítulo seguinte é apresentado o protótipo computacional que foi desenvolvido como prova de conceito para avaliar o modelo. São apresentados e discutidos detalhes de implementação como as tecnologias usadas e a interligação entre elas.

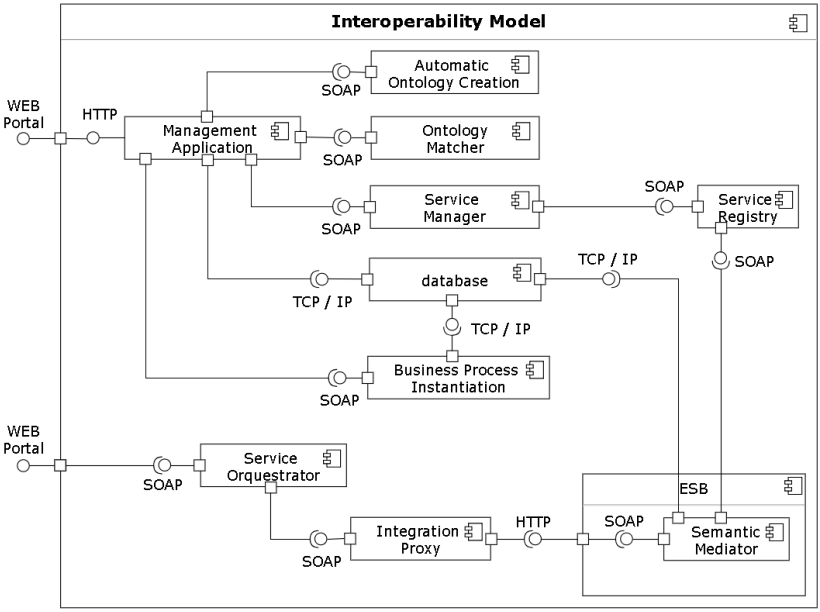
6 IMPLEMENTAÇÃO E RESULTADOS

Este capítulo apresenta a implementação do modelo de interoperabilidade proposto. Trata-se de um protótipo computacional usado como prova de conceito para avaliar o modelo de interoperabilidade/integração e os resultados alcançados em cada fase.

Todos os módulos da proposta foram implementados como serviços web baseados no protocolo SOAP/XML usando a linguagem Java, e implantados em uma rede local e ambiente controlado.

A implementação do modelo segue a abordagem clássica de BPM-SOA, sendo cada processo de negócio modelado em BPMN tomando como referência o padrão UBL, e posteriormente transformado em um arquivo BPEL que é executado no módulo de orquestração de serviços. Na Figura 37 apresentada a seguir se observa, de forma geral, os componentes que fazem parte da implementação do modelo de interoperabilidade e seus enlaces de comunicação.

Figura 37: Diagrama de componentes do protótipo



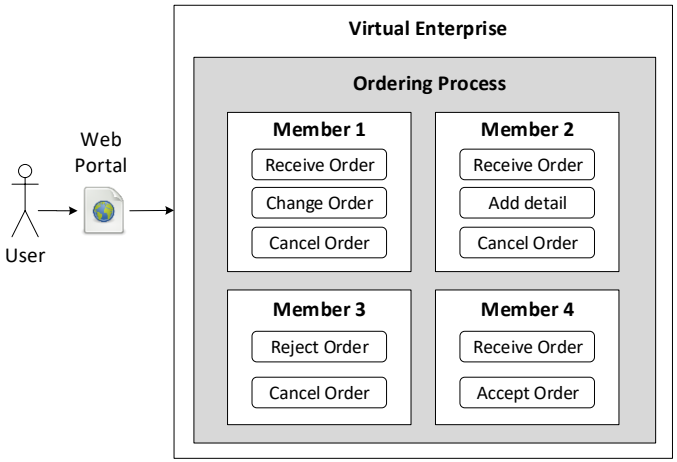
Fonte: Própria

Para representar os sistemas dos membros do ACV e considerando os pressupostos apresentados no capítulo anterior, foram desenvolvidos um conjunto de serviços web com modelos de dados heterogêneos (inclusive usando modelos de dados em diferentes línguas). As tecnologias usadas permitidas para participar da colaboração são serviços web baseados em SOAP ou REST e usando formatos de dados XML ou JSON. Além disso, foi desenvolvida uma interface web que permite aos participantes do ACV/EV executar todos os casos de uso da Figura 22.

Para poder avaliar o funcionamento foram implementados os processos de negócio baseados no padrão UBL correspondentes à criação de pedidos (*Ordering*), faturamento (*Billing*) e pagamento (*Payment*). Esses processos foram modelados incialmente em BPMN e posteriormente transformados à sintaxe BPEL. Esses últimos serão utilizados para de fato instanciar uma colaboração entre os serviços dos membros.

Na sequência, são retomadas cada uma das três fases da proposta apresentadas na seção 5.5, e detalhadas desde o ponto de vista da implementação em um cenário de teste que envolve uma EV conformada por 4 membros que dividem as atividades dos 3 processos de negócio implementados. No total, cada membro expõe 3 serviços na integração, um para cada processo de negócio. No entanto, nem todos implementam o total das atividades. Na Figura 38 é apresentada a distribuição de atividades entre os 4 membros para o processo de *ordering.*

Figura 38: Estrutura de atividades do processo *ordering* nos membros da EV

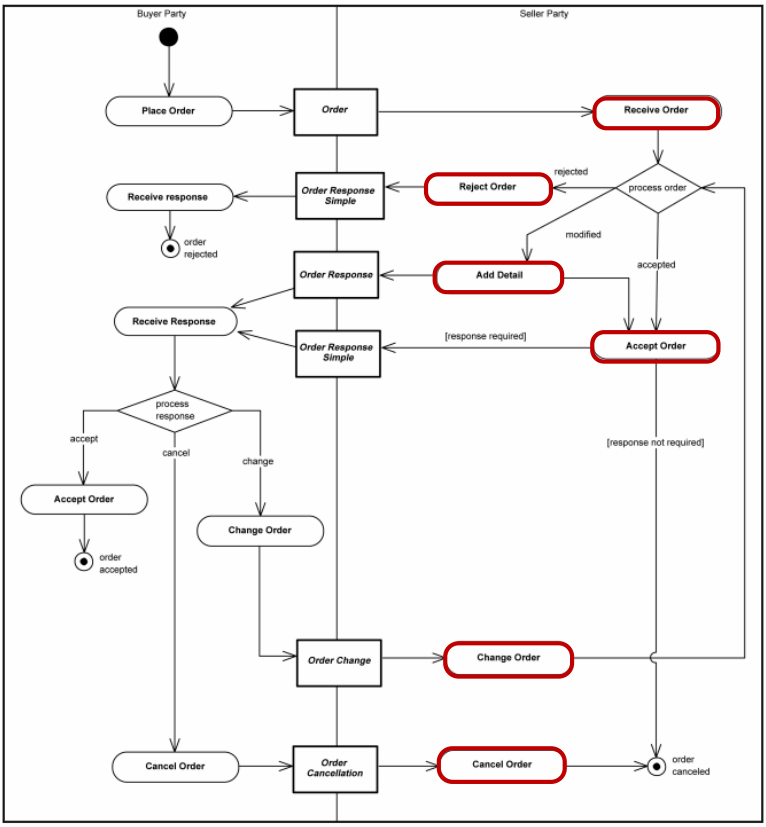


Fonte: Própria

O processo *Ordering* consiste em uma colaboração que cria uma obrigação contratual entre o cliente e o vendedor ou prestador do serviço (EV). Basicamente, permite interações de negociação entre as partes para definir um pedido. Entre as possíveis ações estão: aceitar, recusar e modificar, do lado da parte vendedora e criar, trocar ou cancelar do lado do cliente.

Como pode-se observar na Figura 38, os membros da EV definida nesta seção oferecem algumas dessas atividades; portanto, é necessário realizar uma alocação de membros às atividades que eles suportam. O modelo que define esse processo de negócio é ilustrado na Figura 39.

Figura 39: Modelo do processo de negócio *Ordering* na especificação UBL



Fonte: (OASIS, 2018)

Tendo definido o processo de negócio, a estrutura da EV e as atividades que os membros dela disponibilizam, se apresenta na sequência os detalhes de implementação de cada uma das 3 fases principais do modelo.

6.1 A FASE DE PLUG-IN

Na fase de Plug-in, o ponto de partida do processo de integração é definir o modelo de dados de referência que vai ser usado no ACV. Uma questão-chave é a criação da ontologia global que representa o modelo escolhido. Segundo foi exposto na seções 3.2 e 3.4.2.1, e considerando que os modelos de referência de processos de negócio mais usados na indústria estão implementados usando esquemas (XSD), optou-se por automatizar o processo de criação da ontologia utilizando a técnica de conversão de XSD/XML para RDF/OWL. Como apresentado em Hacherouf, Bahloul e Cruz (2015), na literatura têm sido exploradas várias abordagens para implementar essa tarefa. A proposta desenvolvida por Yüksel (2013) se adaptou às necessidades do protótipo e portanto foi escolhida, ajustada e implementada como um serviço web baseado no protocolo SOAP para representar o *módulo de criação automática de ontologias* definido na seção 5.5.1.1.

No algoritmo de criação da ontologia, Yüksel (2013) define um conjunto de regras de conversão de XSD para OWL, detalhadas no Quadro 5.

Quadro 5: Regras de conversão de XSD para OWL

Elemento XSD	Elemento OWL
`xs:simpleType`	Cria-se um `rdfs:Datatype` com o sufixo "*Datatype*" no nome do tipo de dado.
`xs:simpleType` com `xs:enumeration`	`rdfs:Datatype` com o sufixo "Datatype" no nome. Adicionalmente, para cada *element enumerations*, uma **owl:Class** como uma subclasse de *EnumeratedValue* é criada. Para cada valor enumerado são criadas instâncias.
`xs:complexType` sobre `xs:complexContent`	Cria-se uma `owl:Class`
`xs:complexType` sobre `xs:simpleContent`	Cria-se uma `owl:Class`

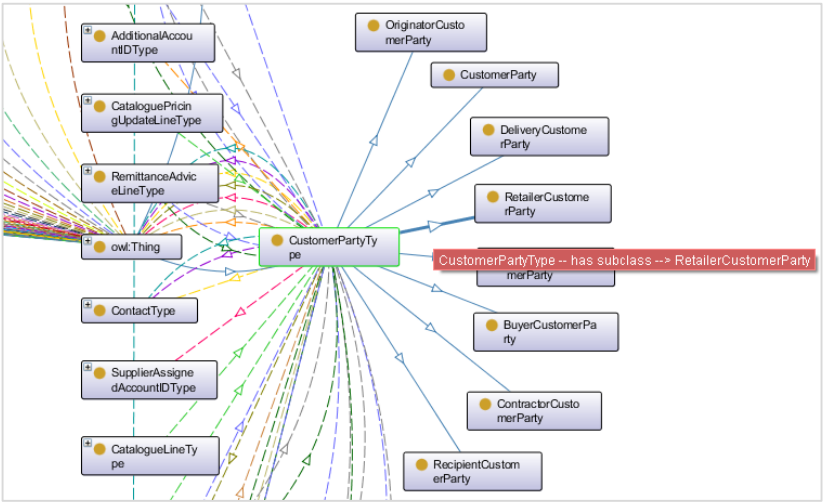
xs:element (global) com **xs:complexType**	Cria-se uma **owl:Class** e uma subclasse dela com referência ao tipo de dado complexo
xs:element (global) com **xs:simpleType**	Cria-se um **owl:Datatype**
xs:element (local a um tipo de dado)	**owl:DatatypeProperty** ou **owl:ObjectProperty** dependendo do tipo de elemento
xs:group	Cria-se uma **owl:Class**
xs:attributeGroup	Cria-se uma **owl:Class**
xs:complexType anónimo	Cria-se uma **owl:Class,** porém a IRI é construída como **Anon_#** e a classe é criada como uma subclasse de **Anon**.
xs:simpleType anónimo	Cria-se um **rdfs:Datatype,** porém a IRI é construída como **Anon_#**
substitutionGroup	Subclasses são geradas para cada membro
xsi:type em um elemento XML	Substitui o tipo abstrato do esquema pelo tipo especificado.

Fonte: Adaptada de Yüksel (2013)

Como apresentado na seção 5.5.1.1, o *módulo de criação automática de ontologias* recebe como entrada arquivos XSD; no caso específico da implementação são os da especificação UBL. O resultado de aplicar as regras de conversão apresentadas no Quadro 5 é uma ontologia em formato OWL contendo os termos e relações definidas no esquema XSD.

A seguir, na Figura 40, é apresentada uma pequena parte da ontologia gerada onde pode-se observar a classe *CustomerPartyType* criada e suas relações com outras entidades. O link azul à direita da classe com espessura maior representa uma relação do tipo *subclass* entre as duas classes que interliga. No caso que está sendo representado na figura, a entidade *RetailerCustomerParty* é uma subclasse de *CustomerPartyType*. O bloco vermelho mostra a relação descrita.

Figura 40: Parte da ontologia de referência (UBL) criada

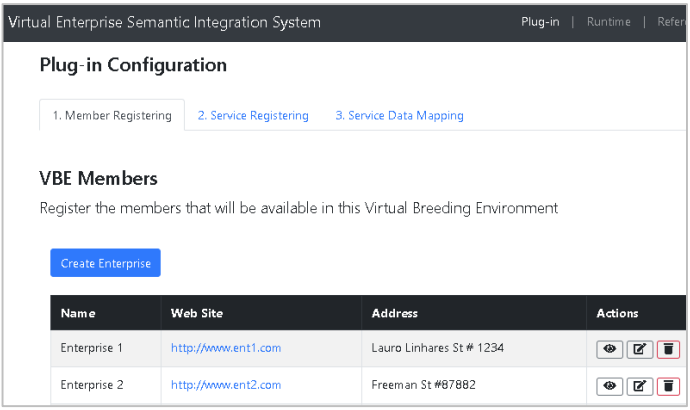


Fonte: Própria

Uma vez definido o modelo de referência de dados do ACV e representado como uma ontologia, o seguinte passo no fluxo é o cadastro dos membros e seus serviços. Nesta atividade cada membro que foi selecionado para participar do ACV deve se registrar na plataforma como uma empresa e adicionalmente deve realizar o processo de registro e representação semântica de todos os seus serviços que vão ser disponibilizados na plataforma de integração. Essa funcionalidade é suportada pelos *módulos de registro de empresas* e de *registro de serviços*, apresentados nas seções 5.5.1.2 e 5.5.1.3 respectivamente. Ainda no módulo de *registro de serviços*, se reusa o módulo de *criação automática de ontologias* como ferramenta de transformação das descrições dos serviços dos membros da EV (expressas como documentos WSDL ou WADL de serviços SOAP e REST respectivamente) em ontologias que são armazenadas na infraestrutura do ACV. Aproveitar o *módulo de criação de ontologias* é possível dado que ditas descrições são implementadas em XML e carregam o seu próprio esquema XSD que descreve os dados que esse serviço utiliza. Além do mais, a implementação do módulo usa a mesma descrição do serviço para registrá-lo no servidor de registro de serviços, que no caso do protótipo é implementado pelo apache jUDDI (que suporta a versão UDDI 3.0). As

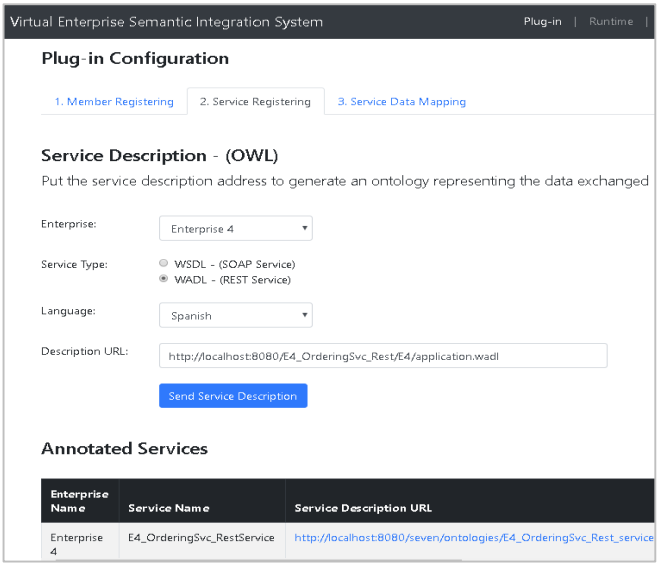
telas do protótipo que implementam estas funcionalidades podem ser observadas na Figura 41 e na Figura 42.

Figura 41: Registro de empresas no ACV



Fonte: Própria

Figura 42: Criação de ontologia e registro de cada serviço disponibilizado pelos membros do ACV



Fonte: Própria

Após registrar cada serviço e criar sua ontologia o seguinte passo é realizar o mapeamento entre a ontologia de referência do ACV e a ontologia do serviço que foi cadastrado. Esta funcionalidade é fornecida pelo terceiro artefato implementado, o serviço de *mapeamento de ontologias (Matcher)*. Esse serviço implementa o módulo descrito na seção 5.5.1.4 e tem como principal responsabilidade encontrar automaticamente o conjunto de equivalências semânticas (alinhamento) entre os termos dos serviços (modelos de dados proprietários das empresas) e os respectivos termos na ontologia de referência (UBL).

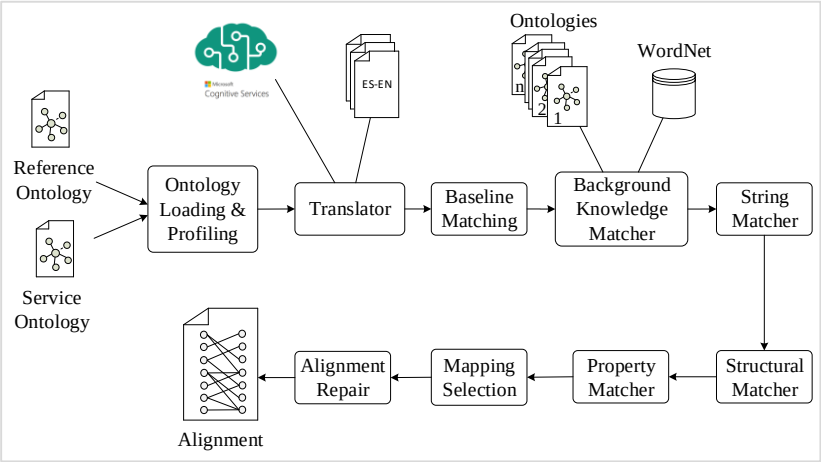
O mapeamento de ontologias é um tópico que tem sido bastante estudado e ainda hoje se mantém como uma linha de pesquisa ativa que atrai o interesse da comunidade científica devido às múltiplas oportunidades de aplicação em diferentes áreas e pela possibilidade de reduzir a lacuna semântica presente entre duas ontologias do mesmo domínio (OTERO-CERDEIRA; RODRÍGUEZ-MARTÍNEZ; GÓMEZ-RODRÍGUEZ, 2015).

Nas últimas décadas têm sido propostas algumas abordagens para resolver o problema de mapeamento de ontologias. Como resultado da revisão da literatura realizada, encontrou-se o trabalho de Achichi et al. (2017), que sumariza as principais ferramentas propostas (*matchers*) e as classifica segundo seu desempenho analisando os mapeamentos gerados em diferentes domínios de teste. Das 23 ferramentas avaliadas a que na média teve melhor desempenho foi *Agreement Maker Light* (AML) (FARIA et al., 2013).

Considerando os resultados citados, o *matcher* AML foi escolhido por causa da relação de tempo de execução e nível desejado de precisão nos mapeamentos. Neste trabalho o AML foi customizado ajustando alguns algoritmos de mapeamento, integrando e o expondo como um serviço web baseado em SOAP para conseguir trabalhar em conjunto com os outros artefatos do protótipo implementado.

Como explicado anteriormente, o processo de *matching* é executado para cada ontologia de serviço criada que representa um serviço disponibilizado por algum dos membros do ACV. De forma geral, esse processo de mapeamento de ontologias realizado pelo *matcher* escolhido (AML) é dividido em 9 etapas principais, as quais executam variados algoritmos encadeados que vão adicionando as correspondências encontradas do resultado da sua operação. Cada etapa executada pelo *matcher* é apresentada na Figura 43 e descrita na sequência.

Figura 43: Etapas do processo de *matching* de ontologias



Fonte: Própria

1. **Carregamento de ontologias**: O AML usa como base para a manipulação de ontologias a biblioteca OWL-API[4]. Nesta etapa a usa para ler as ontologias de entrada e recuperar as informações necessárias para preencher suas próprias estruturas de dados, o *Lexicon* e o *RelationshipMap*.
   O *Lexicon* é uma estrutura de dados interna do AML que vincula cada classe em uma ontologia com seus "nomes", (nomes locais, *labels* e sinônimos) e com a fonte desses nomes (ou seja, se eles vêm de um nome ou *label* local). Por outro lado, *RelationshipMap* é uma estrutura de dados que vincula cada classe às classes relacionadas a ela por meio de relações do tipo *is a*, *part of* ou *disjoint*:

   - Nomes locais, *labels* e anotações de sinônimos de Classes, *Object Properties* e *Data Properties* são normalizados e armazenados no *Lexicon* da ontologia correspondente. O *AML* obtém automaticamente novos sinônimos para cada nome, removendo palavras de

[4] https://github.com/owlcs/owlapi

parada (*Stop Words*) iniciais e finais e removendo as seções de nomes entre parênteses.

- *Domains* e *Ranges* de *Object Properties* e *Data Properties* são armazenados nos objetos *Property* da Ontologia.
- As relações entre classes e entre propriedades são armazenadas em um *RelationshipMap* global.
- Casos de disjunção implícita entre classes que possuem restrições incompatíveis em sua definição são inferidos e também explicitados no *RelationshipMap*.
- O AML não armazena nem usa comentários, definições ou instâncias.

Após o carregamento das ontologias, o problema de *matching* é analisado considerando o tamanho delas, seus idiomas e sua proporção de propriedades/classes.

2. **Tradução:** Esta etapa oferece a funcionalidade de tradução automática, usando o serviço de tradução da Microsoft. Esse serviço é chamado quando são identificadas ontologias com línguas diferentes. AML emprega este serviço para traduzir os nomes de todas as classes e propriedades da primeira ontologia para as da segunda e vice-versa. A tradução é feita enviando para o serviço tradutor o nome completo da entidade. Para melhorar o desempenho, o AML emprega uma estratégia de cache, armazenando localmente todos os resultados de tradução em arquivos de dicionário e consulta o tradutor somente quando nenhuma tradução armazenada é encontrada.
3. **Matching base:** O AML emprega um algoritmo ponderado de equivalências de *strings* eficiente e preciso, o *Lexical Matcher* (FARIA et al., 2013), para obter um alinhamento de base das classes disponíveis nas ontologias de entrada.
4. **Matching com conhecimento background:** O AML fornece a funcionalidade de importar fontes de conhecimento externo que podem ser usadas como mediadores entre as ontologias de input, por exemplo *WordNet*. Considerando o tema foco do trabalho, nesta etapa foi realizada uma busca de ontologias disponíveis na web (usando bases de dados de artigos, o portal de busca semântica *swoogle*[5] e *google*, filtrando por arquivos com

[5] http://swoogle.umbc.edu

extensão *.rdf* e *.owl*) relacionadas a negócios eletrônicos e integração empresarial. Como resultado desse processo foram selecionadas 4 ontologias, entre elas a ontologia ENIO (*ENterprise application Integration Ontology*) proposta por Bouras, Gouvas e Mentzas (2007). Dado que WordNet é uma base de dados de termos em inglês é usado apenas para ontologias nesta língua.

5. **Matching de palavras e strings:** Para ampliar ainda mais o alinhamento base, o AML emprega um algoritmo de similaridade baseado em palavras (o *WordMatcher*) e um algoritmo de similaridade de strings (o *ParametricStringMatcher* que implementa medidas de similaridade como: *Levenshtein*, *Jaro-Winkler*, *Q-gram*, etc.).
   Para pequenas ontologias, o AML também emprega o *Multi-Word Matcher*, que combina nomes de várias palavras altamente relacionadas com palavras que tem sinônimos comuns em WordNet, e o Matcher de acrônimos, que tenta achar correspondências entre acrônimos e o nome completo equivalente.
6. **Matching Estrutural:** Para ontologias de pequeno e médio porte, o AML emprega um algoritmo de correspondência estrutural, chamado *Neighbour Similarity Matcher*. Este algoritmo calcula a similaridade entre duas classes, propagando a similaridade de seus ancestrais e descendentes correspondentes usando um fator de ponderação para calcular a distância.
7. **Matching de propriedades**: Quando as ontologias de entrada têm uma alta proporção de propriedades/classes, AML emprega o *PropertyMatcher*. Esse algoritmo primeiro garante que as propriedades tenham o mesmo tipo de *domains* e *ranges*. Se sim, o matcher aplica os algoritmos descritos anteriormente para comparar os nomes das propriedades, aplicando uma correspondência de nome completo e similaridade de palavras/*strings*.
8. **Seleção de mappings:** o AML emprega um algoritmo de seleção chamado *Ranked Selector* (FARIA et al., 2013) com o intuito de reduzir a cardinalidade do alinhamento gerado. Dependendo do tamanho das ontologias de entrada, uma de três estratégias de seleção é usada: estrita (*strict*), permissiva (*permissive*) ou híbrida (*hybrid*). Na estratégia de seleção estrita não são permitidos mapeamentos simultâneos, ou seja, diferentes mapeamentos para a mesma classe/propriedade. Como resultado

é retornado um alinhamento de 1 para 1. Na estratégia de seleção permissiva os mapeamentos simultâneos são permitidos se a pontuação de similaridade for exatamente a mesma. Na seleção híbrida até dois mapeamentos por classe são permitidos acima de 75% de similaridade, sendo a seleção permissiva aplicada abaixo desse limite.

9. **Reparação:** É a última etapa no processo de *matching*. O algoritmo heurístico descrito em Santos et al. (2015) é executado com o intuito de garantir que o alinhamento final seja coerente.

Uma vez executado o processo de mapeamento o conjunto de correspondências semânticas entre termos (alinhamento) é obtido, as quais são armazenadas em um banco de dados MySQL disponível na infraestrutura de integração do ACV.

A seguir é apresentado o log do processo de mapeamento do serviço do membro 4 (Figura 38) implementado usando a tecnologia REST. Esse serviço tem a particularidade de seu modelo de dados ser implementado em espanhol; portanto, a etapa de tradução é ativada. É possível observar no log a execução das etapas descritas anteriormente desde a carga de ontologias até a reparação final do alinhamento feita antes de entregar o resultado final.

Listagem 3: Log do processo de mapeamento entre uma ontologia de um serviço e a de referência

```
1   Loading source ontology
2   http://ev.das.ufsc.br/ontologies/E4_OrderingSvc_Rest_serviceOn
    tology# loaded in 1 seconds
3   Classes: 24
4   Names: 24
5   Individuals: 0
6   Properties: 2
7   Loading target ontology
8   http://semantic.das.ufsc.br/Ontologies/UBL# loaded in 2 seconds
9   Classes: 2676
10  Names: 2685
11  Individuals: 0
12  Properties: 1352
13  Direct Relationships: 4790
14  Running transitive closure on RelationshipMap
15  Transitive closure finished in 1 seconds
16  Extended Relationships: 22216
17  Disjoints: 53
18  Finished!
19  Opening es-en dictionary
```

```
20  Using stored dictionary and complement with MS Translator.
21  Finished
22  Running Background Knowledge Matcher
23  Running Lexical Matcher
24  Finished in 0 seconds
25  Testing ecommerce.owl
26  http://www.owl-ontologies.com/ecommerce.owl loaded in 0 seconds
27  Running Cross-Reference Matcher using http://www.owl-
    ontologies.com/ecommerce.owl
28  Finished in 0 seconds
29  Testing invoice.owl
30  http://www.ontologydesignpatterns.org/cp/owl/invoice.owl
    loaded in 1 seconds
31  Running Cross-Reference Matcher using
    http://www.ontologydesignpatterns.org/cp/owl/invoice.owl
32  Finished in 0 seconds
33  Testing TaxonomyENIO.owl
34  http://www.imu.iccs.gr/fusion/Taxonomy.owl loaded in 0 seconds
35  Running Cross-Reference Matcher using
    http://www.imu.iccs.gr/fusion/Taxonomy.owl
36  Finished in 0 seconds
37  Testing mesh.lexicon
38  Running Mediating Matcher using mesh.lexicon
39  Finished in 0 seconds
40  Testing WordNet
41  Running WordNet Matcher
42  Finished in 26 seconds
43  Sorting and selecting background knowledge sources
44  Finished in 33 seconds
45  Extending Alignment with String Matcher
46  Matching Children & Parents
47  Matching Siblings
48  Finished in 0 seconds
49  Extending Alignment with Neighbor Similarity Matcher
50  Finished in 0 seconds
51  Extending Alignment with Property Matcher
52  Finished in 0 seconds
53  Building Repair Map
54  Computed check list in 0 seconds
55  Core fragments: 63 classes
56  Check list: 2 classes to check
57  Computed ancestral paths in 0 seconds
58  Paths to process: 0
59  Computed minimal conflict sets in 0 seconds
60  Sets of conflicting mappings: 0
61  Repair Map finished in 0 seconds
62  Alignment is coherent
```

Fonte: Própria

Em vista da complexidade envolvida no processo de *matching*, existe a possibilidade de imprecisões nas correspondências geradas. Portanto, considerando que a qualidade dessas correspondências é crucial para o funcionamento adequado de todo o modelo de interoperabilidade proposto (mediação de mensagens), a implementação não pode garantir que todos os mapeamentos gerados sejam corretos. Como alternativa, foi definido um limiar configurável no intervalo de 0 a 1, que representa o nível de similaridade semântica entre dois termos. Para os experimentos deste trabalho ele foi fixado em 0.95. Se a medida de similaridade é superior ao limiar, o modelo confirma o mapeamento como correto automaticamente; caso contrário, ele tem que ser aprovado pelo usuário que está executando o processo. Na Figura 44, é apresentado um trecho do resultado do processo de mapeamento apresentado na Listagem 3.

Figura 44: Trecho do XML resultado do processo de mapeamento entre ontologias

Fonte: Própria

No trecho exposto podem-se observar 4 correspondências (<*return*>) de termos identificados pelo serviço de mapeamento semântico (*Ontology Matcher* na Figura 37). Partindo da definição de

correspondência exposta na seção 3.4.2.2, ela é composta por 4 elementos principais: as duas entidades (<*entity1*> e <*entity2*>), o identificador (<*id*>) e a medida de similaridade dos termos (<*measure*>). Na correspondência de id igual a 75 a entidade *Pedido* foi mapeada com a entidade de referência *Order* calculando para elas uma similaridade de 0.9801; logo, considerando o limiar definido de 0.95, esse mapeamento é marcado como correto. Da mesma forma ocorre com a entidade *Pais*, que foi marcada como equivalente à entidade *Country*. Porém, a outra entidade *Total* foi mapeada a duas entidades presentes na ontologia de referência e sua medida de similaridade é inferior ao limiar impossibilitando marca-las como equivalentes. Essa ambiguidade terá que ser resolvida pelo usuário posteriormente.

Considerando a importância que os alinhamentos semânticos entre as ontologias de serviços e a de referência têm para obter a interoperabilidade e integração dos diferentes sistemas das empresas membro em uma EV, é necessário ter os meios para avaliar se esse processo de mapeamento de alguma forma está contribuindo e incrementando a agilidade do processo de integração. Para isso são aplicadas as medidas de avaliação de alinhamentos descritas na seção 3.4.2.2 a cada um dos alinhamentos gerados para os 12 serviços das empresas-membro avaliando-os contra seu respectivo alinhamento de referência.

Como explicado no início desse capitulo, não existem serviços publicamente disponíveis relacionados a processos de negócio interempresariais que permitissem testar o protótipo com dados reais. Para contornar esse problema, todos os serviços dos hipotéticos membros foram projetados e implementados para terem várias diferenças sintáticas e semânticas, procurando simular cenários reais. Adicionalmente, foi criado um alinhamento de referência baseado na definição de objetos de negócio da especificação UBL, o que permitiu avaliar os alinhamentos gerados durante o processo de *matching*. A seguir, na Tabela 1, são apresentados os resultados da avaliação para cada empresa

Tabela 1: Resultados da avaliação do processo de *matching* por empresa

Emp	**Svc**	**VP**	**FP**	**FN**	**P**	**Rc**	**F**	**H**	**O**
E1	3	69	24	18	0.74	0.79	0.77	0.38	0.52
E2	3	63	18	12	0.78	0.84	0.81	0.32	0.60
E3	3	48	15	12	0.76	0.80	0.78	0.36	0.55
E4	3	57	9	9	0.86	0.86	0.86	0.24	0.73
Total	12	237	66	51	0.78	0.82	0.80	0.32	0.60

Fonte: Própria

Na Tabela 1 se apresentam as diferentes medidas de avaliação aplicadas. Centralizando os resultados do processo de alinhamento por empresa, as informações são:

- **Emp:** Empresa;
- **Svc:** Número de serviços da empresa alinhados;
- **VP:** Verdadeiros Positivos;
- **FP:** Falsos Positivos;
- **FN:** Falsos Negativos;
- **P:** Precisão;
- **Rc:** *Recall*;
- **F:** *F-measure*;
- **H:** *Hamming Distance*;
- **O:** *Overall*.

Considerando que a *F-measure* pondera as medidas de precisão e *recall*, quanto mais perto de 1 melhor o alinhamento. Portanto, como exposto na tabela na linha inferior, o total da média macro nessa medida é 0.80; quer dizer que aproximadamente 80% das correspondências foram resolvidas corretamente.

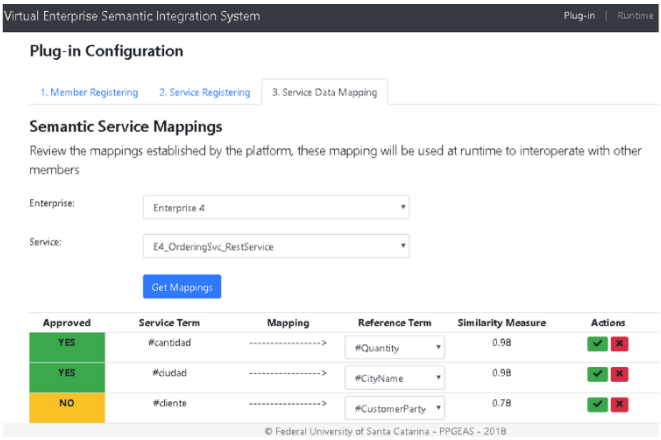
Outras medidas interessantes são a distância de *Hamming* e *Overall*. A primeira medida explica quão diferentes são o alinhamento gerado pelo *matcher* e o alinhamento usado como referência. Idealmente busca-se que o valor dessa medida tenda a 0. Para o cenário de teste o total obtido com valor de 0.32 indica que os alinhamentos são parecidos em aproximadamente 68%.

Por outro lado, a medida *Overall* apresenta informação relevante para o processo de validação de mapeamentos que realiza o usuário da plataforma, indicando quanto esforço é necessário para corrigir o alinhamento gerado. Essa medida está entre o intervalo [-1, 1], sendo que quando o *Overall* é igual a 0 quer dizer que é necessário consertar a metade (50%) das correspondências do alinhamento. Um valor negativo indica que o resultado do *matching* é pobre e que não vale a pena o esforço para corrigir o alinhamento. No cenário de teste um *Overall* de 0.6 corresponde a 80% de acerto nas correspondências do *matching*, indicando que o esforço para corrigir o alinhamento é de apenas 20%. A partir da análise feita pode-se dizer que o processo de *matching* apresenta resultados bons agilizando o processo de integração dos sistemas dos membros na EV.

Levando em consideração que o processo de mapeamento não é correto em 100% dos casos, é necessária a validação do analista que está

a cargo da integração. Para facilitar a validação das correspondências foi criada a tela mostrada na Figura 45, onde o usuário do sistema pode observar os resultados do processo de mapeamento descrito anteriormente e pode editar/aprovar as correspondências geradas. Pode-se constatar na figura o critério de aprovação que o sistema usa ao marcar automaticamente como corretos os mapeamentos cuja medida de similaridade entre os dois termos que estão sendo avaliados ultrapassa o limar de similaridade configurável (definido no sistema no valor de 0.95). Entretanto, é importante observar que o mapeamento que não foi automaticamente aprovado (embora seja correto) não ultrapassa o limiar definido e como consequência requer a aprovação manual por parte do usuário que esteja executando o processo.

Figura 45. Validação dos mapeamentos semânticos gerados automaticamente



Fonte: Própria

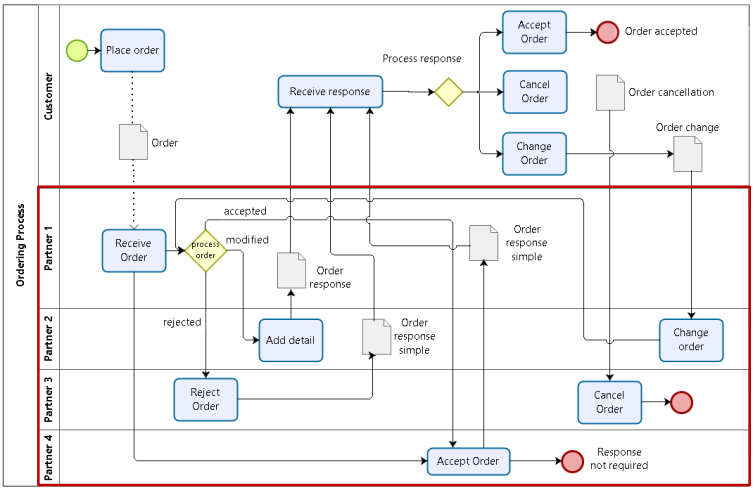
Uma vez que o processo de mapeamento semântico seja concluído em todos os serviços dos membros do ACV, é possível iniciar com a formação de EVs. No entanto, para de fato começar a execução dos seus processos de negócio, é necessário primeiro defini-los.

Como dito na introdução do capítulo, foi adotada a abordagem clássica em SOA de construção dos processos de negócio em BPMN e sua posterior transformação em BPEL. Isso permite obter uma visão geral de como as atividades do processo de negócio se dividem entre seus participantes, facilitando a compreensão geral das responsabilidades.

A seguir são apresentadas duas imagens contendo o modelo do processo de negócio tanto em BPMN (ver Figura 46) quanto em BPEL

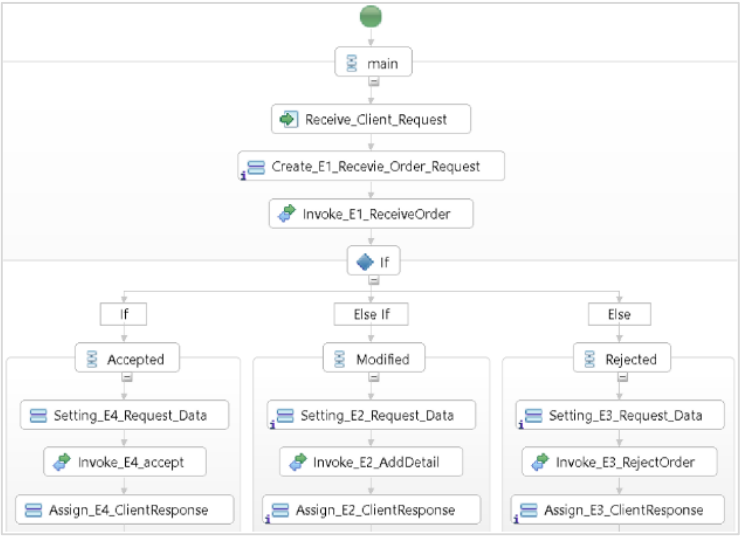
(ver Figura 47). Esse processo foi instanciado para trabalhar na EV da Figura 38 e foi usado para realizar os testes de integração dos módulos do protótipo.

Figura 46: Modelo em BPMN do processo de *Ordering*



Fonte: Própria

Figura 47. Parte do modelo do processo de Ordering representado em BPEL

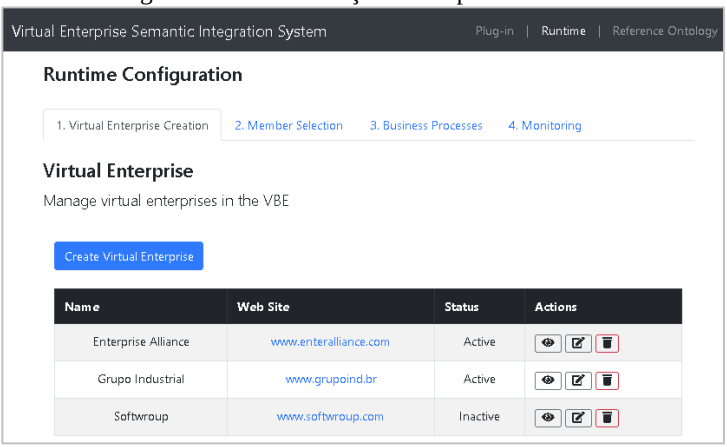


Fonte: Própria

6.2 A FASE DE PLAY

Rede de produção contínua ecossistema cluster Uma vez a fase de *plug-in* terminada, com todas as empresas membro do ACV devidamente cadastradas na plataforma de integração e todos os seus serviços mapeados contra o modelo de referência, as empresas passam a estar preparadas (do ponto de vista de TI) para participarem de EVs quando elas foram selecionadas para tal. Para esta tarefa, a plataforma provê uma interface mostrada na Figura 48 a seguir.

Figura 48: Tela de criação de Empresas Virtuais

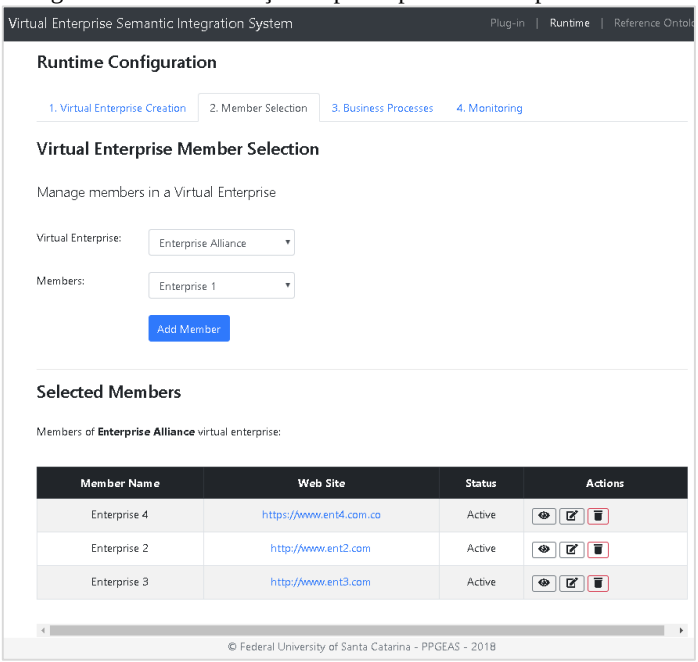


Fonte: Própria

Como já explanado anteriormente, o suporte de criação de EVs é parcial, portanto, este trabalho assume que “de alguma forma” uma empresa ou um ACV recebeu uma oportunidade de negócio e “de alguma forma” um grupo de empresas foi selecionado e contratado para executar algumas tarefas. Ainda, que uma vez definida a EV, o Coordenador da EV define quais tarefas cada empresa-membro executará. Todo esse “plano” é formalizado, no caso, em BPMN. Esses pressupostos têm a ver que o foco do trabalho é a integração dos membros e a interoperabilidade dos seus sistemas. A plataforma apenas provê uma funcionalidade básica de cadastro e gerenciamento dos membros da EV; portanto, não implementa a totalidade das funções descritas na literatura na etapa de criação de uma EV, como definição de contratos, aplicação de modelos de governança, entre outros.

Além da criação das EVs com as suas informações básicas, nessa fase o coordenador da EV seleciona os parceiros (já cadastrados no ACV na etapa de *plug-in*) que vão fazer parte da EV. A Figura 49 apresenta a tela implementada para suportar as atividades descritas.

Figura 49: Tela de seleção de participantes na Empresa Virtual



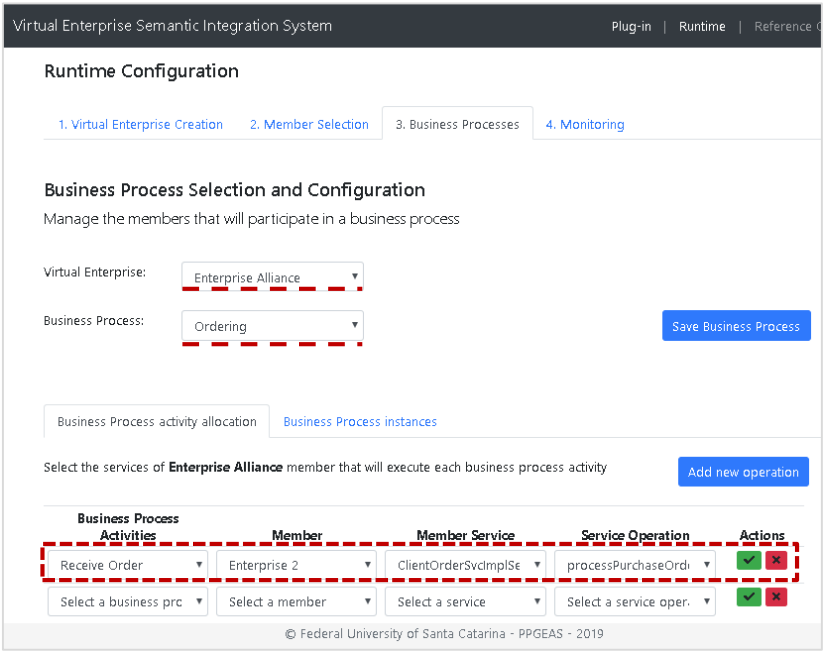
Fonte: Própria

Sob a ótica adotada BPMN/BPEL para representar os processos de negócio na plataforma, o protótipo permite selecionar os processos de negócio (esqueletos) disponíveis na EV e associar uma operação de qualquer um dos serviços membros da EV para que seja o responsável de executar essa etapa do processo. Uma vez alocados os serviços, o processo de negócio é salvo na plataforma, que cria uma instância com as informações dos serviços que participam dele. Essa funcionalidade corresponde ao processo apresentado na Figura 29.

A tela que implementa essa funcionalidade é apresentada na Figura 50. Pode-se observar a configuração de uma instância do processo de negócio "*Ordering*" na EV "*Enterprise Alliance*". Na parte inferior no bloco tracejado se expõe a alocação da operação

"*ProcessPurchaseOrder*" ao serviço "*ClientOrderSvcImplService*" do membro "*Enterprise 2*".

Figura 50: Tela de associação de serviços aos processos de negócio



Fonte: Própria

Uma vez salvo o processo de negócio, a plataforma permite que ele seja executado. A ferramenta escolhida para esse processo é o motor de execução (ou orquestrador) BPEL, Apache ODE[6]. Seguindo o modelo proposto na Figura 33, este invoca o end-point *proxy* exposto como um serviço web SOAP, que serve como ponto de entrada para a infraestrutura de mensagens que é representada pelo barramento de serviços empresariais ESB (MULE[7]). Este último é responsável pelo gerenciamento da troca de mensagens entre o orquestrador e os serviços destinatários. Assim que o *proxy* receber a mensagem, ele a repassa para o serviço de mediação semântica, que resolve as correspondências semânticas entre os termos da informação que está trafegando no

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

barramento e realiza as transformações necessárias: *Lowering* para entregá-la ao destinatário e *Lifting* para transformar a resposta do destinatário no formato de dados de referência UBL.

Um exemplo do resultado da operação *Lowering* no processo de mediação semântica é mostrado na Figura 51, onde se representa a interação entre um serviço REST de um membro de uma EV que usa JSON para representar seu modelo de dados **(b)** e a requisição recebida do orquestrador BPEL usando o modelo de dados de referência UBL em XML **(a)**.

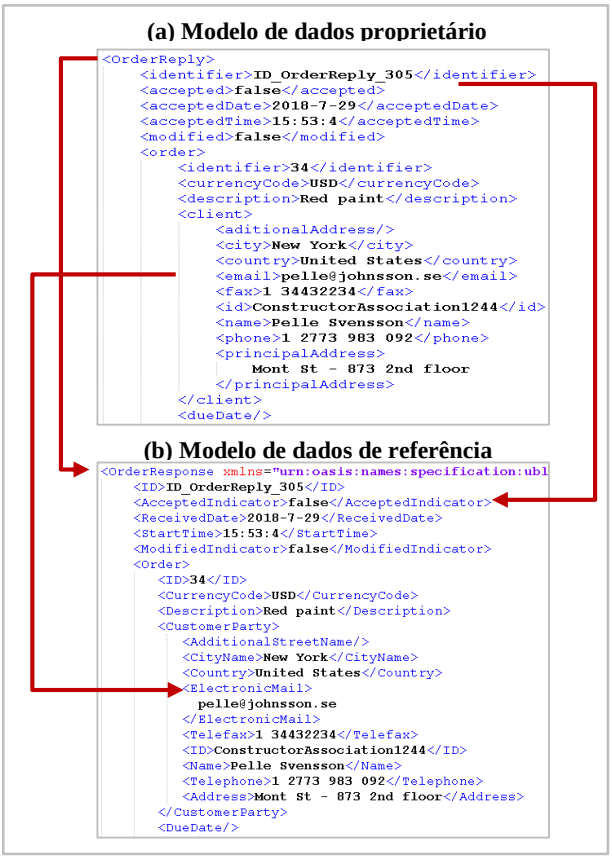
Figura 51: Resultado do processo de *lowering* de um serviço REST/JSON

Fonte: Própria

Usando as correspondências semânticas realizadas na fase de *plugin*, o processo de mediação efetua a transformação entre modelos de dados. Pode-se verificar que as duas mensagens são semanticamente equivalentes mesmo tendo modelos de dados em línguas diferentes (inglês e espanhol) e formatos de dados diferentes (XML e JSON). Como exemplo, são apresentadas três correspondências onde se constata que a transformação da mensagem funciona como era esperado.

Depois da mensagem ser entregue ao serviço destinatário, ele irá retornar a informação resultado do seu processamento no seu modelo de dados; portanto, é necessário transformar de volta para o modelo de referência, processo correspondente ao *Lifting* (Figura 52).

Figura 52: Resultado do processo de *lifting* de um serviço SOAP/XML

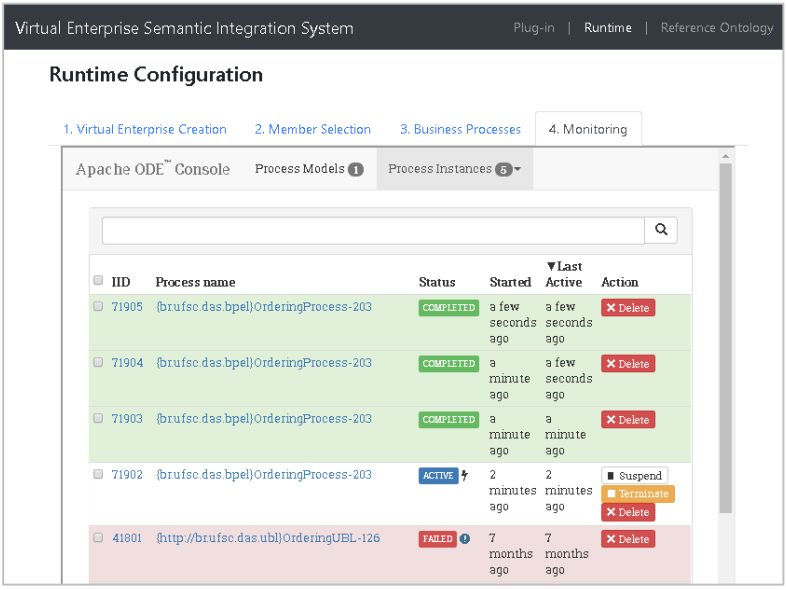


Fonte: Própria.

Essa transformação (*Lifting*) se evidencia na Figura 52, com trechos da resposta de um serviço SOAP/XML **(a)** e a transformação equivalente para o modelo de referência UBL **(b)**. Além disso são destacadas três correspondências. Nesse caso do modelo de dados proprietário os termos *OrderReply*, *accepted* e *email* são semanticamente equivalentes no modelo de referência a *OrderResponse*, *AcceptedIndicator* e *ElectronicMail* respectivamente, dessa mesma forma com todos os atributos apresentados.

Uma vez que todos os membros foram cadastrados e seus modelos de dados alinhados com o modelo de referência, a instanciação dos processos de negócio é concluída e assim os processos de negócio podem ser efetivamente executados. É possível acompanhar os estados da execução dos diferentes processos de negócio no orquestrador por meio da tela de monitoramento da plataforma proposta. A Figura 53 apresenta a tela que implementa o monitoramento descrito.

Figura 53: Monitoramento de execuções dos processos de negócio



Fonte: Própria

Com a implementação da fase *play* finalizada e tendo já avaliado o processo de interoperabilidade semântica na seção 6.1, a seguir são realizados alguns testes de integração com o intuito de avaliar o processo

geral de integração. Esses testes, consistem em implantar todos os componentes do modelo implementados em um ambiente controlado e realizar requisições controladas para medir alguns indicadores de desempenho (tempos de reposta, número de requisições com erro, entre outras) dos processos de negócio implementados.

Todos os componentes foram implantados em um computador ASUS X456UF, com 8 Gb de memória RAM e processador Intel Core i7 de 4 núcleos a 2.6GHz, organizados como apresentado na Figura 54. Nessa figura podem ser observados 5 contêineres principais e os componentes com suas interligações e protocolos de comunicação. Entre os contêineres, há 2 servidores Apache *Tomcat 9*; um deles executa o orquestrador de serviços *Apache ODE* e o outro disponibiliza os serviços web e o portal de suporte necessários para o funcionamento do modelo proposto. O terceiro servidor (parte superior direita) disponibiliza o registro de serviços *Apache jUDDI*. Como apresentado anteriormente o ESB é suportado pela ferramenta *MULE* que roda em um contêiner *Glassfish.* Por último está o contêiner que executa o banco de dados, implementado em *MySQL*.

Os testes foram projetados e executados usando a ferramenta *JMeter*[8], que permite projetar planos de teste definindo o corpo das requisições que serão feitas, o número de requisições, os intervalos de tempo usados para executá-las, entre outras opções, e colhe alguns indicadores do resultado da execução desse plano de teste, entre eles o Apdex (*Application Performance Index*).

A EV usada para testar é a definida na Figura 38 e o processo de negócio é o *Ordering* (que dispara uma série de ações e, conforme projeto da EV, desencadeia a execução de vários outros processos UBL). O primeiro teste consiste em realizar 100 requisições sequenciais entre tempos de 500ms no motor de execução BPEL simulando a chegada de 100 ordens. O compilado dos tempos de execução e erros é apresentado na tabela a seguir.

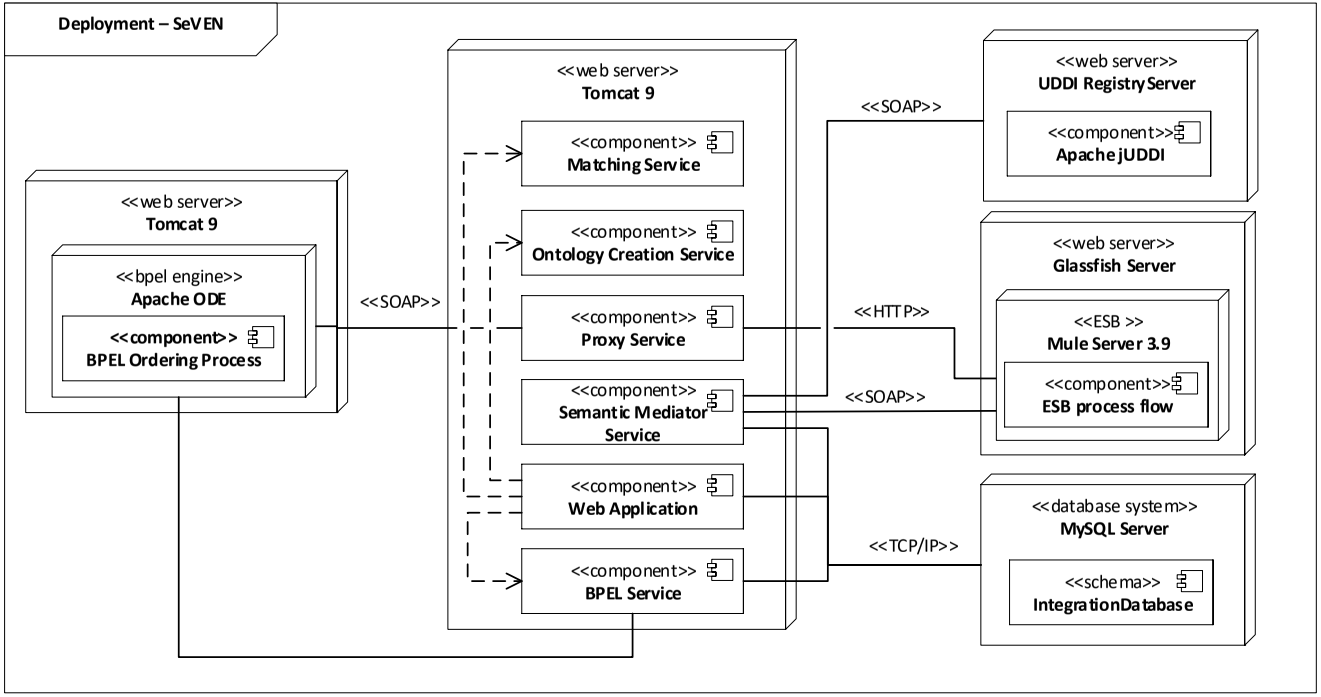
Tabela 2. Resultados de 100 requisições no processo de negócio *Ordering*

Requisições	**Execuções**			**Tempos de resposta (ms)**			
Label	**#**	**KO**	**% KO**	**Média**	**Min**	**Max**	**90th pct**
Ordering Process BPEL	100	0	0%	234.88	129	951	381.20

Fonte: Própria

[8] https://jmeter.apache.org/usermanual/index.html

Figura 54: Diagrama de implantação do protótipo

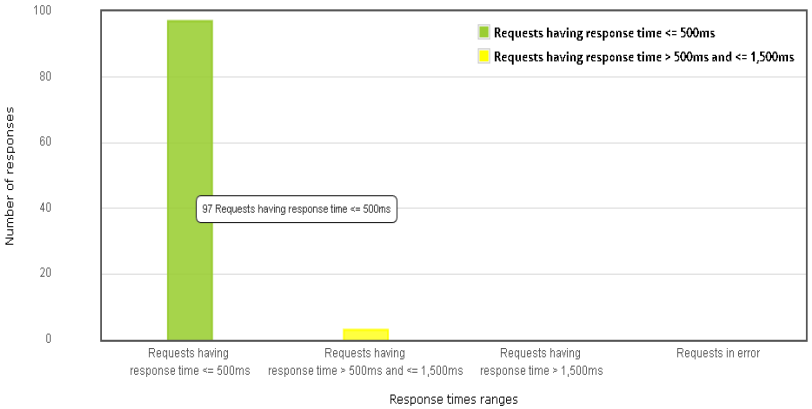


Fonte: Própria

Como se evidencia na Tabela 2, no teste de execuções sequenciais o protótipo consegue atender as 100 requisições com sucesso gastando em média 235ms para retornar um resultado.

Na Figura 55 se apresenta uma visão global do tempo para retornar as requisições. Pode ser observado na figura que 97 requisições das 100 feitas retornaram em um tempo menor ou igual a 500ms e 3 com tempos entre 500ms e 1500ms.

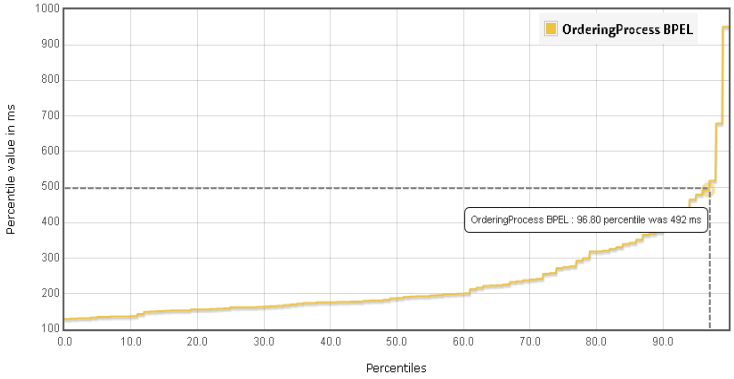
Figura 55: Visão global do tempo de resposta para 100 requisições sequenciais



Fonte: Própria

Na Figura 56 observa-se que 97% das requisições realizadas retornaram em um tempo menor a 500ms, assim como ilustra-se que o máximo tempo gasto por uma requisição foi aproximadamente 950ms.

Figura 56. Percentis de tempo de resposta de 100 requisições sequenciais



Fonte: Própria

Considerando que o cenário de negócios eletrônicos normalmente não tem associados requisitos temporais fortes, sendo que as interações são realizadas geralmente através de um portal web que permite ao cliente submeter as ordens de compra, os tempos obtidos no cenário anterior são considerados bons.

O seguinte teste consiste em aumentar o número de requisições ao triplo (300), criando uma a cada 0.6 segundos, com o intuito de observar o comportamento do protótipo em um cenário de maior estresse. Os resultados obtidos são sumarizados na Tabela 3.

Tabela 3. Resultados de 300 requisições no processo de negócio *Ordering*

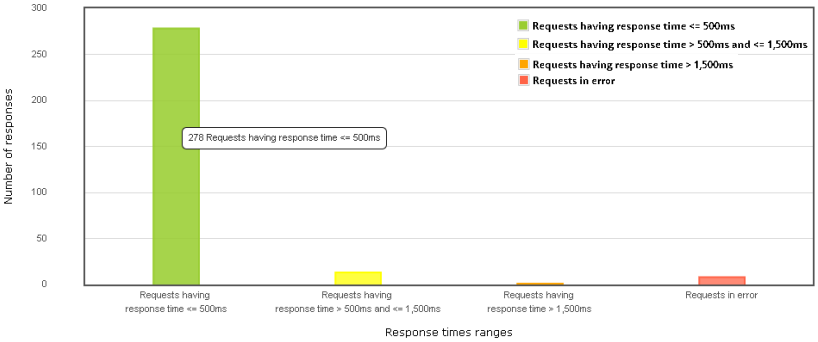
Requisições	**Execuções**			**Tempos de resposta (ms)**			
Label	**#**	**KO**	**% KO**	**Média**	**Min**	**Max**	**90 pct**
Ordering Process BPEL	300	8	2.67%	3492.12	110	120320	428.90

Fonte: Própria

Analisando resultados da tabela nota-se que das 300 requisições 8 apresentaram erro (KO). Nos tempos reportados na tabela pode-se observar que a média de resposta é aproximadamente de 3.5 segundos, o que é alto em comparação com os tempos retornados no teste anterior (0.2 segundos). Em um cenário real esses 8 erros representariam pedidos submetidos pelos clientes que não foram atendidos pelo sistema da EV, o que de fato não deveria acontecer em um cenário real considerando as implicações monetárias dos negócios e de imagem das empresas envolvidas. Por outro lado, é importante ressaltar que o cenário de testes executa múltiplas requisições em paralelo em um tempo muito curto, o que dadas as caraterísticas de negócios das EVs é pouco provável que aconteça com grande frequência. De qualquer forma, considerando que o protótipo é apenas uma prova de conceito, esses problemas podem ser corrigidos a nível de implementação.

Na Tabela 3 pode-se observar que o tempo máximo gasto por uma requisição é ao redor de 2 minutos, o que coincide com o tempo de *timeout* configurado no orquestrador, o que naturalmente aumenta a média de resposta. Da mesma forma que no teste anterior, na Figura 57 se detalha a visão geral do tempo gasto pelas requisições para retornar e os erros reportados. Pode-se observar que 278 das 300 requisições retornaram antes de 500ms, 13 com tempos entre 500ms e 1500ms, 1 com tempo maior a 1500ms e 8 erros.

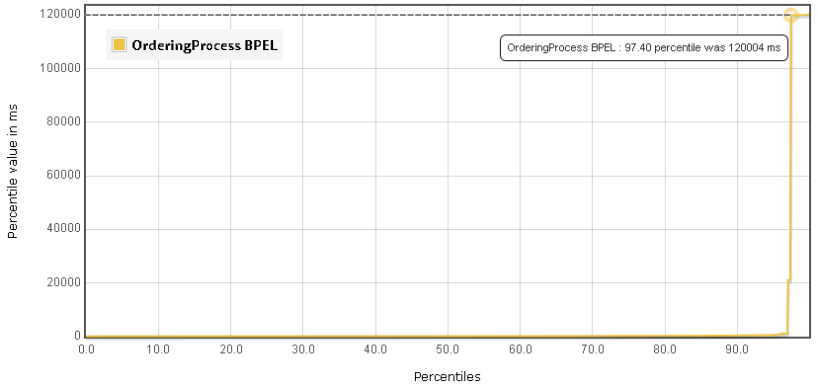
Figura 57. Visão global do tempo de resposta para 300 requisições incrementais



Fonte: Própria

Para validar se o tempo médio reportado na Tabela 3 é realmente representativo no conjunto de requisições realizadas, é apresentada a Figura 58, estabelecendo a relação do tempo de resposta com os percentis das requisições. É possível observar que 2.6% das requisições retornaram aos 120 segundos, o que coincide com as 8 reportadas como erro. Conforme dito anteriormente esse tempo de 120s corresponde ao *timeout* configurado no orquestrador BPEL.

Figura 58. Percentis de tempo de resposta de 300 requisições incrementais



Fonte: Própria

Para validar se esses erros estão sendo produzidos realmente por *timeout* no orquestrador, foi criado um plano de teste que simula as

requisições da instância do processo de negócio de *Ordering* apresentado na Figura 46. Isto permite obter resultados específicos do comportamento de cada um dos serviços participantes no processo de negócio, facilitando assim a verificação dos tempos de resposta individuais. O teste consistiu em realizar 200 requisições de forma incremental, divididas em 4 grupos de 50 requisições que ativam uma delas a cada 600ms. Os resultados da execução desse processo são sumarizados na Tabela 4, onde pode-se observar que o serviço da empresa 1 (*E1Ordering*) executa a operação *receiveOrder* 200 vezes. Isto ocorre porque ele é a atividade de entrada no processo de negócio; os outros serviços são executados segundo o fluxo de informação adotado depois de receber a ordem. Isto é, se a ordem é aceita, se é aceita com mudanças ou se é rejeitada (que são as opções em UBL sobre o que pode ocorrer quando há um processo *Ordering*). No total essas três operações foram executadas 62, 73 e 65 vezes, respectivamente, acumulando um total de 200 como era esperado.

Em relação aos erros (KO), ocorreram no total 7, distribuídos entre os 4 serviços, e os tempos de resposta estão no intervalo de 36ms até 1019ms. Pode-se evidenciar que o serviço que demorou mais para responder é o encarregado de receber a ordem. Isso é uma consequência da estrutura do processo de negócio, já que ele sozinho executa 200 requisições, o que o expõe a uma maior carga e gerando assim esses atrasos em comparação com os outros.

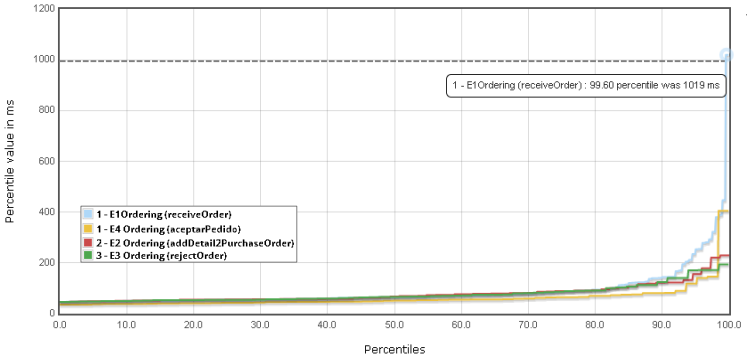
Tabela 4. Resultados de 200 requisições incrementais ao serviço Proxy

Requisições	**Execuções**			**Tempos de resposta (ms)**			
Label	**#**	**KO**	**% KO**	**Média**	**Min**	**Max**	**90 pct**
E1Ordering (*receiveOrder*)	200	2	1	90.58	46	1019	143.7
E4 Ordering (*aceptarPedido*)	62	1	1.61	62.34	36	405	81.7
E2 Ordering (*addDetail2PurchaseOrder*)	73	2	2.74	78.78	46	230	122.6
E3 Ordering (*rejectOrder*)	65	2	3.08	77.55	46	194	131.4

Fonte: Própria

A distribuição dos tempos de resposta ao longo dos percentis é apresentada na Figura 59.

Figura 59. Percentis de tempo de resposta de 200 requisições ao Proxy



Fonte: Própria

Como se constata na Figura 59, não são evidenciados tempos de resposta maiores a 1200ms, o que é muito menor em comparação com os testes realizados com o orquestrador BPEL incluso, onde se reportaram tempos de resposta de até 2 minutos. Na figura se confirma a informação da Tabela 4, onde o serviço que mais tempo precisou para responder foi o *E1Ordering* (*receiveOrder*).

Com os dados obtidos na totalidade dos testes realizados é possível calcular alguns indicadores para avaliar o desempenho do sistema em relação à interoperabilidade. Como exposto anteriormente, a ferramenta utilizada para executar e colher os dados dos testes (*jMeter*) gera informações sobre o desempenho do software que está sendo testado, que permitem a obtenção de alguns dos indicadores apresentados na seção 3.3. A Tabela 5 apresentada a seguir condensa os indicadores de desempenho para os resultados dos casos expostos anteriormente nas Tabela 2, Tabela 3 e Tabela 4.

Tabela 5. Indicadores de desempenho de interoperação

Métrica	**Número de requisições**			
	100 BPEL	**300 BPEL**	**200 Proxy**	**Total**
Apdex (500ms de tolerância)	0.985	0.948	0.981	0.973
Qualidade da troca de informação	0	8	7	15
Qualidade da operação	0	0	0	0
Duração da interoperação (ms)	234.88	3492.12	81.93	1269.64

Fonte: Própria

Segundo a informação exposta na tabela pode-se concluir que o processo de integração para os testes feitos é bom. Baseado no índice de desempenho da aplicação (*Apdex*) com uma tolerância de atraso na mensagem de 500ms se obteve na média 0.973%, resultados muito próximos de 1, indicando um bom desempenho.

Por outro lado, a *qualidade da troca de informação* representa a diferença entre as requisições feitas e as que retornaram satisfatoriamente (número de erros). Portanto, busca-se que esse número tenda a zero. Considerando a quantidade de requisições realizadas e como evidenciado na Figura 58 os 8 erros reportados foram por mensagens que não retornaram (*timeout*) e representam uma taxa de 2.67% para o cenário de 300 requisições ao processo BPEL, o que é considerado baixo.

Em relação a *qualidade de operação*, a diferença com a *qualidade de troca de informação* é basicamente que a primeira representa a diferença das mensagens enviadas e as mensagens recebidas pelo destinatário (sem importar se retornaram algum valor). Nesse sentido, todas as mensagens enviadas foram recebidas (aquelas que não retornaram apresentaram erros de processamento e não de recepção), portanto, a porcentagem de erro nesse indicador é 0%.

Finalmente, encontra-se a medida de *tempo de interoperação*, que basicamente mede o tempo que uma mensagem leva para chegar até o destinatário e disponibilizar sua resposta de volta ao sistema solicitante. Os resultados gerais são bons. No entanto, é interessante observar que a diferença dos tempos entre o teste de 300 requisições e os outros é considerável (aproximadamente 3400ms). Isto pode ser explicado dado que a medida usada para calcular esse índice é a média, portanto, valoriza demais o tempo reportado nos *timeouts* (120000) mesmo tendo uma representação minoritária no número total de mensagens.

Considerando os resultados apresentados é possível dizer que o processo de interoperabilidade funciona segundo o esperado, viabilizando assim que as EVs interoperem e colaborem na execução de processos de negócio distribuídos.

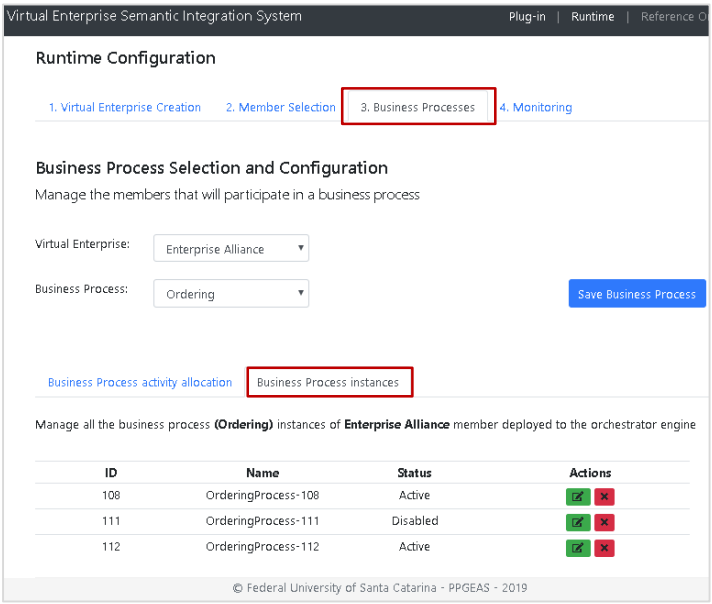
Em concordância com o ciclo de vida de uma EV, essas atividades de execução e monitoramento pertencem à etapa de operação. Entretanto, quando uma empresa quer ou deve deixar de participar em uma EV ou finaliza as suas tarefas, a etapa do ciclo de vida correspondente é a evolução ou a dissolução. A seguir, é apresentada a implementação da fase de *unplug*, que fornece estas funcionalidades.

6.3 A FASE DE UNPLUG

Conforme foi apresentado anteriormente existe a possibilidade de um parceiro sair da EV em um determinado momento da colaboração. Considerando essa possibilidade, nesta última etapa a plataforma fornece um método simplificado de substituir um membro em um processo de negócio, lembrando que este processo tem algumas restrições adotadas dada a natureza distribuída e as complexidades legais e tecnológicas envolvidas.

Selecionando a aba de "*processos de negócio"* (3. *Business Processes*) e selecionando na parte inferior a aba "*instâncias de processo de negócio*" (*Business Process Instances*) é possível gerenciar todas as instâncias de processo criadas, tendo a opção de editar seus detalhes, como a alocação de serviços dos membros às atividades do processo. Na Figura 60 a seguir é apresentada a aba de gerenciamento de instâncias.

Figura 60: Gerenciamento de instâncias de processos de negócio

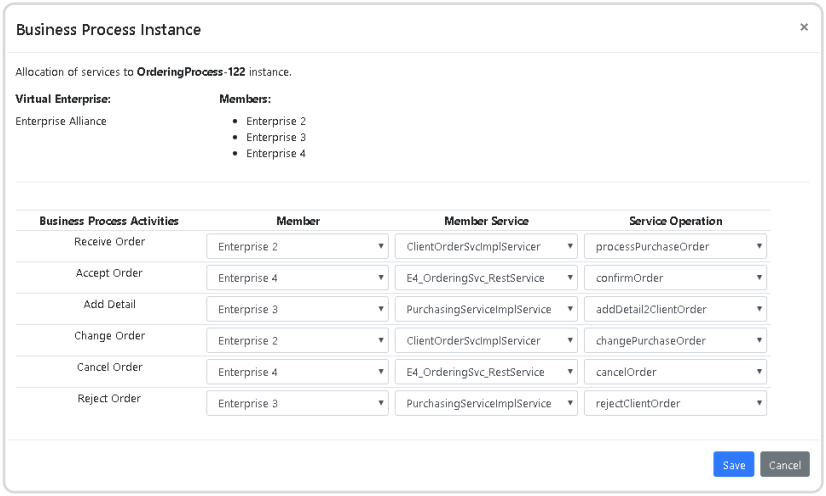


Fonte: Própria

Ao selecionar uma das instâncias dos processos de negócio disponíveis na tabela (ver Figura 60 parte inferior) e acessar seus detalhes

é possível observar e modificar sua distribuição de serviços. Uma vez salvas as modificações de substituição de parceiros e serviços, a plataforma usa o esqueleto do processo de negócio para gerar uma nova instância e realiza a implantação no orquestrador de serviços. A instância antiga é removida do orquestrador, permitindo assim substituir um membro. A tela em questão pode-se observar na Figura 61.

Figura 61: Tela de edição da alocação de serviços à instância do processo de negócio



Fonte: Própria

É importante destacar que a instância da qual o parceiro sai é recriada; portanto, os dados associados ao seu funcionamento são perdidos. É essa a razão que uma das restrições desta funcionalidade é que o parceiro a ser substituído não esteja executando nenhuma tarefa e que o processo de negócio tenha finalizado.

A título de exemplo, é apresentado na Figura 62 um trecho do processo BPEL *Ordering* antes e depois de substituir o membro 1 pelo membro 4 na tarefa de recepção de pedidos. É possível observar a mudança feita na informação que vai ser enviada pelo *orquestrador de serviços* ao *Proxy* da integração quando executar o processo BPEL, especificamente as tags ***<to>***, que corresponde ao endereço do serviço destinatário, e a ***<serviceOperation>***, que corresponde à operação que será executada no serviço invocado.

Figura 62. Mudança no XML do processo BPEL quando é trocado um serviço

(a) Trecho do processo BPEL antes de trocar o membro

```
<bpel:assign validate="no" name="Create_Recevie_Order_Request">
    <bpel:copy>
        <bpel:from>
            <bpel:literal xml:space="preserve">
                <tns:handleRequest
                    xmlns:tns="http://proxy.integration.das.ufsc.br/"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <request>
                        <from>http://localhost:8080/E1_OrderingService/E1OrderingService?wsdl</from>
                        <payload></payload>
                        <serviceOperation>receiveOrder</serviceOperation>
                        <to>http://localhost:8080/E1_OrderingService/E1OrderingService?wsdl</to>
                    </request>
                </tns:handleRequest>
            </bpel:literal>
```

(b) Trecho do processo BPEL depois de trocar o membro

```
<bpel:assign validate="no" name="Create_Recevie_Order_Request">
    <bpel:copy>
        <bpel:from>
            <bpel:literal xml:space="preserve">
                <tns:handleRequest
                    xmlns:tns="http://proxy.integration.das.ufsc.br/"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <request>
                        <from>http://localhost:8080/E4_OrderingSvc_Rest/E4/application.wadl</from>
                        <payload></payload>
                        <serviceOperation>recibirPedido</serviceOperation>
                        <to>http://localhost:8080/E4_OrderingSvc_Rest/E4/application.wadl</to>
                    </request>
                </tns:handleRequest>
            </bpel:literal>
```

Fonte: Própria

Com esta funcionalidade é finalizada a implementação do protótipo usado como prova de conceito para avaliar o modelo proposto. Foram apresentadas as funcionalidades fornecidas pela plataforma, ilustrando seu funcionamento aplicado no caso de uma empresa virtual que executa o processo de recebimento de pedidos (*Ordering*) e seu posterior desdobramento em vários outros processos de negócio a serem executados por outras empresas da EV. Foram abordadas cada uma das fases propostas e apresentados resultados de cada uma delas, avaliando de forma quantitativa os processos de alinhamento semântico e de interoperabilidade.

6.4 AVALIAÇÃO DOS ATRIBUTOS DE QUALIDADE DA ARQUITETURA DERIVADA DO MODELO PROPOSTO

Dado que o modelo proposto neste trabalho é uma representação abstrata de uma colaboração interempresarial no contexto de RCOs, e mais especificamente dos tipos ACV e EVs, para efeitos de avaliação o modelo foi instanciado em uma arquitetura que permitiu implementá-lo

como prova de conceito. Além das avaliações quantitativas realizadas quanto ao desempenho em termos gerais da implementação, é importante também avaliar a arquitetura proposta em si, que é o artefato conceitual mais concreto e passível de alguma avaliação visto que o modelo propriamente dito é uma abstração.

Em se tratando de uma arquitetura, se faz uma avaliação qualitativa, com base em propriedades. Para tal foram considerados os atributos de qualidade definidos em (BASS; CLEMENTS; KAZMAN, 2003; O'BRIEN; MERSON; BASS, 2007), sendo o primeiro para arquiteturas "clássicas" (não orientadas a serviços) e o segundo para orientadas a serviços:

1. **Interoperabilidade:** refere-se ao grau em que dois ou mais sistemas podem trocar informações por meio de interfaces em um dado contexto. A definição desse atributo inclui tanto a habilidade de trocar dados (interoperabilidade sintática) quanto a habilidade de interpretar esses dados trocados (interoperabilidade semântica). Esse é um dos atributos principais abordados no modelo. Esta propriedade é garantida principalmente pelo uso de uma arquitetura híbrida de ontologias, um módulo de mapeamentos semânticos semiautomático, um ESB como infraestrutura de comunicação e o uso de padrões abertos como UBL, SOAP, WSDL, UDDI, BPEL, REST, XML, JSON entre outros. O detalhamento desses módulos foi definido no capítulo 5.

2. **Desempenho:** de forma geral esse atributo está relacionado ao tempo e a capacidade do sistema de software para atender aos requisitos temporais, ou seja, retornar com a resposta esperada a eventos (como interrupções, mensagens, etc.) dentro do prazo temporal estabelecido como aceitável.
   Segundo os testes apresentados ao longo desse capítulo foi evidenciado que, em média, considerando cenários de baixo e alto estresse, o tempo de resposta foi tido como baixo para o ambiente de transações eletrônicas interempresariais (foco do trabalho), mesmo considerando que os testes foram efetuados em um ambiente controlado. É importante salientar que em processos distribuídos e em ambientes dinâmicos, que permitem a interação de diferentes fontes de dados heterogêneas e ainda com tecnologias diferentes, é necessário o processo de mediação. Portanto, devido às transformações necessárias, é gerado um

overhead que afeta o tempo de resposta e impacta o desempenho geral da solução, porém dentro de limites aceitáveis.

3. **Segurança:** refere-se à capacidade do sistema de proteger dados e informações contra acessos não autorizados e de, ao mesmo tempo, garantir acesso a pessoas e sistemas autorizados. De forma geral, a propriedade de segurança está associada a quatro princípios: confidencialidade (que garante que o acesso à informação e serviços seja concedido apenas a atores autorizados); autenticidade (relacionado à confiança, garante que o autor ou remetente indicado é o responsável pela informação); integridade (que garante que a informação não seja corrompida); e disponibilidade (que garante que o serviço esteja disponível em tempo hábil). Existem alguns padrões, como *WS-Security*, *OAuth*, que propõem mecanismos de segurança em sistemas baseados em SOA; porém, esses aspectos não foram considerados a nível de projeto da arquitetura, senão apenas aspectos básicos de segurança intrinsecamente suportados pelas ferramentas usadas na implementação.

4. **Confiabilidade:** é a capacidade de um sistema continuar operando ao longo do tempo sem falhas. Diversos aspectos de confiabilidade são importantes dentro de uma solução SOA, particularmente a confiabilidade das mensagens trocadas entre clientes e provedores de serviços e a confiabilidade dos próprios serviços.
   Do ponto de vista da confiabilidade das mensagens, o modelo proposto considera o uso do ESB como uma infraestrutura de integração e suporte de comunicações utilizando-o como uma ponte entre o orquestrador de serviços (que executa os processos de negócio) e o serviço destinatário (que está sendo invocado, como apresentado na Figura 36).
   Do ponto de vista dos serviços, existem alguns desafios que têm que ser abordados para garantir que, caso ocorra uma falha, ela não será reportada ao usuário e que o sistema não fique em um estado inconsistente ou mesmo que pare de operar. Isso tem a ver com resiliência computacional em sistemas largamente distribuídos e assíncronos, um problema dos mais complexos. Apesar de haver middlewares e propostas de implementações para tratar resiliência em SOA, esta perspectiva não foi prevista nesta prova de conceito e é, a propósito, um tema de sugestão de

trabalhos futuros desta dissertação. No modelo proposto, esses mecanismos de compensação podem ser aplicados no projeto dos processos de negócio (na linguagem BPEL) controlando possíveis erros e executando atividades de compensação que mantenham a informação do sistema em um estado estável.

5. **Disponibilidade:** Faz referência a uma propriedade do software que indica que ele está pronto para realizar sua tarefa quando for requerido. O modelo foi projetado seguindo padrões de integração e considerando tecnologias amplamente usadas na indústria, permitindo assim escalar horizontalmente por meio de réplicas e consequentemente aumentando sua disponibilidade conforme o cenário desejado.

6. **Modificabilidade:** essa propriedade se refere a determinar o custo e risco de fazer mudanças em um sistema software da forma mais rápida e econômica possível. Dado que o modelo proposto se baseia na filosofia da arquitetura orientada a serviços (SOA), cada componente (serviço) é independente e tem um contrato que o descreve, beneficiando assim a possibilidade de adicionar novos serviços ou modificar a lógica de implementação dos já existentes sem afetar outros módulos. A atividade de maior impacto no sistema é uma mudança que implique alteração dos contratos dos serviços dado que os clientes que estão utilizando essa interface vão ser afetados, tendo que atualizar seu código para incluir essa mudança.

7. **Testabilidade:** propriedade que indica o grau em que um sistema ou serviço facilita o estabelecimento de critérios de teste e a realização de testes para determinar se ditos critérios são atendidos. Pelo fato do modelo ser baseado em SOA, e dada a modularização inerente dessa arquitetura, o processo de testes isolados (unitários) com um escopo bem definido e independente dos outros módulos envolvidos no modelo é facilitado. Já os testes de integração, implicam alguns desafios que devem ser considerados pela natureza do ambiente distribuído. Por exemplo, em um processo de negócio que envolve múltiplos serviços é mais difícil ter uma rastreabilidade do fluxo das informações dado que os serviços estão distribuídos na rede. Outro ponto importante é que se é utilizada descoberta de serviços em tempo de execução não é possível ter a

previsibilidade de qual serviço será invocado, o que dificulta o processo de teste automatizado.

8. **Usabilidade:** é uma medida da qualidade da experiência de um usuário na interação com informações ou serviços fornecidos pelo sistema. Normalmente mede a facilidade com que o usuário realiza uma tarefa desejada e o tipo de suporte que o sistema oferece. Em relação a esse aspecto, o modelo considera um módulo de interface web para o usuário (ver Figura 54) com o intuito de facilitar o uso da ferramenta. O módulo inclui a configuração dos membros do ACV, verificação e ajuste dos mapeamentos semânticos gerados entre os diferentes serviços, criação de EVs, gerenciamento e monitoramento da execução dos processos de negócio.

9.Escalabilidade: Escalabilidade é a capacidade que têm uma aplicação SOA de funcionar bem (sem degradação de outros atributos de qualidade) quando o sistema é alterado em tamanho ou em volume para atender às necessidades dos usuários. O principal problema na escalabilidade de uma aplicação SOA é a capacidade do servidor (em que os serviços estão localizados) para atender um número crescente de clientes do serviço sem degradação do desempenho. Considerando que o protótipo que implementa o modelo de interoperabilidade proposto é uma prova de conceito, decisões arquiteturais não consideraram aspectos de escalabilidade, como balanceadores de carga, embora a modularidade inerente da arquitetura SOA permita que os módulos de suporte do modelo proposto possam ser escalados individualmente conforme seja necessário.

7 CONCLUSÕES

Este trabalho apresentou um modelo de interoperabilidade voltado para a integração de Empresas Virtuais (EV) no contexto de redes colaborativas de organizações.

Tirando proveito das potencialidades do paradigma de arquitetura orientada a serviços (SOA), o modelo desenvolvido oferece os meios para integrar de uma forma mais ágil os mais diversos tipos de sistemas das empresas que fazem parte de uma EV, conservando a autonomia delas para escolher seus modelos de dados e sendo minimamente invasivo nas tecnologias de integração. Esta agilidade se reflete na forma praticamente toda dinâmica e transparente com a qual empresas entram e saem de uma EV ao longo do seu ciclo de vida, suportando uma interoperabilidade tanto sintática como semântica. Para tal, adota fortemente padrões abertos de TI e de modelos de processos de negócios bem como padrões (*patterns*) consolidados de integração de sistemas.

O modelo foi concebido com base no estado-da-arte em termos de arquiteturas de integração de sistemas heterogêneos e largamente distribuídos, considerando ainda a natural flexibilidade e escalabilidade necessária em um cenário de EV.

O modelo proposto é constituído de três fases principais diretamente relacionadas ao ciclo de vida de EV. A primeira fase, de "conexão" (*plug*), consiste na entrada das empresas ao ACV (Ambiente de Criação de empresas/organizações Virtuais) e na consequente habilitação dos sistemas computacionais delas para poderem trabalhar em EVs. A segunda, fase de "operação" (*play*), é onde as EVs são formadas e seus membros interagem executando seus processos de negócio. A terceira e última fase, de "desconexão" (*unplug*), se refere a quando os membros de uma EV saem da colaboração ao terminarem suas obrigações legais ou são substituídos por outros membros quando de problemas.

Ao longo dessas três fases, um amplo e variado conjunto de atividades é executado para que as empresas de uma EV, uma vez formada, consigam transacionar adequadamente entre si sem a necessidade de intervenção ou mapeamentos manuais durante a sua operação e desconexão.

O modelo assume que o ACV – e assim as empresas-membro – tem como diretriz básica de TI que todos os sistemas das empresas devem ser encapsulados como serviços de software sob uma ótica SOA, mesmo que implementadas em diferentes tecnologias. Desta forma, todas as funcionalidades dos sistemas podem flexivelmente e de forma desacoplada serem invocados por um orquestrador, que por sua vez está

atrelado aos inúmeros processos de negócios que uma EV / seus membros devem executar. Em função disto, foi adotada uma abordagem de integração geral de *messaging*, baseada em barramento de mensagens (*message bus*) e implementada com uma ferramenta de ESB (*Enterprise Service Bus*).

Uma estratégia fundamental dentro do modelo para garantir a interoperabilidade semântica foi a adoção de um modelo interno de referência para processos de negócios. Para tal, adotou-se o padrão aberto UBL (*Universal Business Language*), que é aplicado em cenários de cadeias de suprimento (*supply chain*), um tipo de rede colaborativa muito semelhante a uma EV só que estática em termos de alteração da composição dos membros. Assim sendo, uma ontologia foi implementada de acordo com a especificação UBL, servindo de estrutura de referência para mediação semântica entre os diferentes sistemas das diversas empresas da EV. Na verdade, o modelo é aberto para que outros padrões de processos de negócio sejam adotados, sejam abertos, sejam proprietários (que é a forma mais usual adotada em redes de empresas, e com isso cada uma tem que fazer mapeamentos nos seus sistemas de acordo com as convenções e acordos definidos pelos participantes ou, por vezes, por uma empresa grande, líder).

Em termos de proposição de valor e contribuição científica do trabalho, foi evidenciado através da revisão da literatura que nos projetos de integração em EVs uma parte significativa do esforço é destinado para atividades de suporte à interoperabilidade entre sistemas, na mediação dos diferentes modelos de dados, normalmente feita efetuado de forma basicamente manual. Entre outros efeitos, isso acarreta em maior custo, usualmente muitos erros e necessidades de retrabalho, falta de padronização e sistematização do processo de integração, e ainda dificuldades de expertise dos integradores devido a múltiplas tecnologias diferentes e de diversas gerações. Esses problemas ganham ainda maior importância em termos de se ter soluções que os minimizem na medida que os ACVs e, por conseguinte, as EVs que dele emergem, são preponderantemente formadas por micro, pequenas e médias empresas, com limitações de toda sorte.

Além disso, verificou-se que muito poucos trabalhos abordam todas as etapas do ciclo de vida das EVs, especialmente a parte de "desconexão" e a completa dinamicidade de entrada e saída de empresas da EV; em outras palavras, um "*play & unplug*" fluído e transparente.

No que toca à interoperabilidade semântica, foram utilizadas ontologias e algoritmos de alinhamento semântico de conceitos visando agilizar o processo de mediação. A partir da modelagem de processos em

um ambiente tipo BPM (*Business Process Management*), os processos de negócio de uma dada EV são modelados e expressos em BPMN (*Business Process Modeling and Notation*). O resultado é posteriormente convertido em BPEL (*Business Process Execution Language*), que atua como o orquestrador geral da execução dos processos, ou seja, da invocação de cada um dos serviços (sistemas) de cada membro da EV em questão, incluindo todos os documentos (e terminologias envolvidas) trocados entre os membros conforme especificação UBL. Quando uma EV muda sua composição (entrada de novas empresas e/ou saída da outras), toda orquestração é dinamicamente adaptada e toda interoperabilidade sintática e semântica é mantida. Internamente na implementação, apesar do BPEL suportar apenas a tecnologia de web services SOAP, adaptações foram realizadas de forma que outras tecnologias pudessem ser usadas, REST em particular. Portanto, todo esse processo ocorre de forma totalmente transparente às empresas e usuários.

Importante ainda ressaltar que uma dada empresa geralmente estará envolvida em várias EVs simultaneamente (e assim por várias empresas diferentes, com diferentes sistemas, TIs, modelos de processos e terminologias), o que é igualmente suportado pelo modelo desenvolvido.

Assume-se que toda a infraestrutura computacional necessária para executar a arquitetura é mantida e/ou hospedada no ambiente do ACV ou do grupo de empresas de cada EV. Isso pode incluir a possibilidade de se utilizar um provedor externo de serviços de Nuvem. Caberá a cada ACV ou EV definir o seu modelo de "implantação" (*deployment*), seja em servidores locais, seja tudo completamente na Nuvem, seja algo híbrido. No modelo e arquitetura propostos, a rigor nada impede que a infraestrutura e os serviços dos sistemas das empresas-membro de EVs estejam hospedados na Nuvem e acessados como SaaS (*Software-as-a-Service*), bastando que o provedor dos serviços disponibilize o acesso a eles. Considerando que normalmente os membros de um ACV (e assim de uma EV) são pequenas e médias empresas, uma arquitetura baseada em Nuvem pode ser atrativa em função de algumas vantagens já bem conhecidas. Por outro lado, há também alguns aspectos potencialmente problemáticos nisso e que devem ser analisados pelos responsáveis do ACV, tais como, o nível de controle e flexibilidade na infraestrutura, de segurança de dados, e de aprisionamento tecnológico (*vendor lock-in*), que podem não atender aos requisitos desejados do dado ACV e/ou de suas EVs.

O protótipo implementado e os experimentos executados mostraram que o alinhamento semântico automático consegue diminuir o esforço necessário no processo de alinhamento de modelos de dados heterogêneos, e por consequência o tempo gasto na fase de configuração e entrada de membros em uma colaboração. Além disto, conseguiu-se agilizar o processo de formação de EVs com o modelo proposto. Isto é importante pois pode servir como um estímulo para as empresas participarem de redes de empresas – uma grande tendência e proeminente modelo de sustentabilidade empresarial – na medida que parte dos problemas de TI e integração ficariam bastante minimizados.

Todavia, considerando a complexidade e a abrangência do tema de integração e interoperação de sistemas, foi necessário delimitar o escopo em alguns aspectos. Do ponto de vista da mediação de modelos de dados só foram abordadas as interoperabilidades sintática e parcialmente semântica, não se explorando a estrutural. Da mesma forma, nenhum aspecto de segurança foi considerado, o que pode gerar vários problemas de interoperabilidade entre diferentes domínios de segurança que precisam ser atravessados, entre eles problemas de acesso a informação (em função de diferentes modelos de governança de TI de cada empresa diferente), e de desempenho.

Além disso, visando manter a estratégia de ser uma solução pouco invasiva, é necessário que as empresas que queiram participar da plataforma exponham seus sistemas como serviços (REST ou SOAP) com suas respectivas descrições WSDL e WADL. No caso de uma empresa ter sistemas legados (o que é o mais usual, a propósito), ela tem que encapsulá-los como serviços, o que é um pré-requisito obrigatório imposto pelo modelo proposto. No mundo das pequenas empresas, isso tipicamente irá implicar em elas terem que contratar alguma empresa de software para fazer esse trabalho e depois mantê-lo.

Outra limitação da proposta foi identificada na fase de *unplug*. Apesar de um certo ineditismo no modelo proposto, a natureza dinâmica da colaboração e a complexidade de se tratar certas questões fez com que alguns pressupostos tivessem que ser assumidos e simplificações realizadas. As mais importantes são que: A troca de serviços na etapa de evolução da EV só é suportada de 1 a 1 em termos de granularidade, o que restringe a possibilidade de uma atividade do processo de negócio ser executada por mais de um serviço; e unicamente é possível realizar a troca do serviço quando este finaliza suas atividades atuais nos processos de negócio nos quais está envolvido. O modelo proposto não considera a transferência parcial de informações (memória) de execução de um serviço para outro.

7.1 TRABALHOS FUTUROS

Considerando a abrangência do tema e também as limitações expostas anteriormente, surgem ideias de trabalhos futuros para melhorar o modelo proposto, entre eles:

- Estudar as técnicas de mapeamento automático de esquemas para assim integrar o suporte de interoperabilidade estrutural.

- Explorar as possibilidades de incluir múltiplos serviços associados a uma atividade de um processo de negócio; isto é, considerar uma granularidade maior das operações nos serviços dos membros da EV e permitir uma alocação de uma composição de serviços a uma atividade do processo de negócio.

- Aprofundar a pesquisa na fase de *unplug* tentando retirar as simplificações feitas na proposta, aumentando ainda mais a flexibilidade do modelo.

- Integrar o suporte à segurança, pesquisando de que forma requisitos de segurança definidos no nível BPM podem ser "traduzidos" no nível tecnológico de integração das EVs para dentro do ESB.

- Implantar o protótipo em um cenário menos controlado, tentando se aproximar mais da realidade industrial, testando processos de negócio mais complexos e avaliar o desempenho em cenários com alto estresse.

REFERÊNCIAS

ABADI, A. et al. An ontology-based framework for virtual enterprise integration and interoperability. **Proceedings of 2016 International Conference on Electrical and Information Technologies, ICEIT 2016**, p. 36–41, 2016.

ACHICHI, M. et al. **Results of the Ontology Alignment Evaluation Initiative 2017**. Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017). **Anais**...: Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz,Michelle Cheatham,Oktie Hassanzadeh.out. 2017

AFSARMANESH, H.; CAMARINHA-MATOS, L. M.; ERMILOVA, E. VBE Reference Framework. In: CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; OLLUS, M. (Eds.). . **Methods and Tools for Collaborative Networked Organizations**. Boston, MA: Springer US, 2008. p. 35–68.

ALLAMARAJU, S. **RESTful Web Services Cookbook**. 1st Ed. ed. [s.l.] O'Reilly, 2010.

BALDO, F.; RABELO, R. J. **Guidelines to Transform Industry Clusters in Virtual Organization Breeding Environments -- A Case Study**. (W. Cellary, E. Estevez, Eds.)Software Services for e-World. **Anais**...Berlin, Heidelberg: Springer Berlin Heidelberg, 2010

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. [s.l.] Addison-Wesley Professional, 2003.

BECKER, I. B. **Electronic data interchange by UN/EDIFACT standards**. Proceedings of the 6th International Conference on the Application of Standards for Open Systems. **Anais**...out. 1990

BEDINI, I. et al. **Transforming XML Schema to OWL Using Patterns**. 2011 IEEE Fifth International Conference on Semantic Computing. **Anais**...2011

BEDINI, I.; NGUYEN, B. **Automatic Ontology Generation: State of the Art**. [s.l: s.n.].

BERNSTEIN, P. A.; HAAS, L. M. Information Integration in the Enterprise. **Commun. ACM**, v. 51, n. 9, p. 72–79, 2008.

BHADORIA, R. S.; CHAUDHARI, N. S.; TOMAR, G. S. The Performance Metric for Enterprise Service Bus (ESB) in SOA system: Theoretical underpinnings and empirical illustrations for information processing. **Information Systems**, v. 65, p. 158–171, 2017.

BLAIR, G. S. et al. Interoperability in Complex Distributed Systems. In: BERNARDO, M.; ISSARNY, V. (Eds.). . **Formal Methods for Eternal Networked Software Systems: 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 1–26.

BOURAS, A.; GOUVAS, P.; MENTZAS, G. **ENIO: An Enterprise Application Integration Ontology**. 18th International Workshop on Database and Expert Systems Applications (DEXA 2007). **Anais**...2007

CAMARINHA-MATOS, L. M.; AFSARMANESH, H. Collaborative networks: a new scientific discipline. **Journal of Intelligent Manufacturing**, v. 16, n. 4, p. 439–452, out. 2005.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H. A comprehensive modeling framework for collaborative networked organizations. **Journal of Intelligent Manufacturing**, v. 18, n. 5, p. 529–542, out. 2007.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H. (EDS.). Collaboration forms. In: **Collaborative Networks: Reference Modeling**. Boston, MA: Springer US, 2008a. p. 51–66.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H. (EDS.). The ARCON modeling framework. In: **Collaborative Networks: Reference Modeling**. Boston, MA: Springer US, 2008b. p. 67–82.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; OLLUS, M. **Ecolead: A Holistic Approach to Creation and Management of Dynamic Virtual Organizations**. (L. M. Camarinha-Matos, H. Afsarmanesh, A. Ortiz, Eds.)Collaborative Networks and Their Breeding Environments. **Anais**...Boston, MA: Springer US, 2005

CHEN, L. Integrating cloud computing services using Enterprise Service Bus (ESB). **Business and management research**, v. 1, n. 1, p. 26, 2012.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. **Distributed systems: concepts and design**. 5th. ed. [s.l.] pearson education, 2012.

DA SILVA SERAPIÃO LEAL, G.; GUÉDRIA, W.; PANETTO, H. Interoperability assessment: A systematic literature review. **Computers in Industry**, v. 106, p. 111–132, 2019.

DACLIN, N.; CHEN, D.; VALLESPIR, B. Developing enterprise collaboration: a methodology to implement and improve interoperability. **Enterprise Information Systems**, v. 10, n. 5, p. 467–504, 2016.

DE SOUZA, A. P.; RABELO, R. J. A Dynamic Services Discovery Model for Better Leveraging BPM and SOA Integration. **International Journal of Information Systems in the Service Sector**, v. 7, n. 1, p. 1–21, 2015.

DEN HAMER, P.; SKRAMSTAD, T. **Autonomic Service-Oriented Architecture for Resilient Complex Systems**. 2011 IEEE 30th Symposium on Reliable Distributed Systems Workshops. **Anais**...out. 2011

DER VEER, H.; WILES, A. **ETSI White Paper No. 3 Achieving Technical Interoperability - the ETSI Approach**. [s.l: s.n.].

DO, H.-H.; RAHM, E. Chapter 53 - COMA — A system for flexible combination of schema matching approaches. In: BERNSTEIN, P. A. et al. (Eds.). . **VLDB '02: Proceedings of the 28th International Conference on Very Large Databases**. San Francisco: Morgan Kaufmann, 2002. p. 610–621.

DUFRESNE, T.; MARTIN, J. Process modeling for e-business. **Information Systems Department, George Mason University**, 2003.

ENGEL, R. et al. **From Encoded EDIFACT Messages to Business Concepts Using Semantic Annotations**. 2012 IEEE 14th International Conference on Commerce and Enterprise Computing. **Anais**...2012

EUZENAT, J.; SHVAIKO, P. **Ontology Matching**. 2nd. ed. [s.l.] Springer Publishing Company, Incorporated, 2013.

FARIA, D. et al. **The AgreementMakerLight Ontology Matching System**. (R. Meersman et al., Eds.)On the Move to Meaningful Internet Systems: OTM 2013 Conferences. **Anais**...Berlin, Heidelberg: Springer Berlin Heidelberg, 2013

FETTKE, P.; LOOS, P. **Reference Modeling for Business Systems Analysis**. Hershey, PA, USA: IGI Global, 2006.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. [s.l.] University of California, Irvine, 2000.

FURDÍK, K. et al. Support of Semantic Interoperability in a Service-based Business Collaboration Platform. **Scalable Computing: Practice and Experience**, v. 12, 2011.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: [s.n.].

GUARINO, N.; OBERLE, D.; STAAB, S. What Is an Ontology? In: STAAB, S.; STUDER, R. (Eds.). . **Handbook on Ontologies**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–17.

HACHEROUF, M.; BAHLOUL, S. N.; CRUZ, C. Transforming XML documents to OWL ontologies: A survey. **Journal of Information Science**, v. 41, n. 2, p. 242–259, 2015.

JARDIM-GONCALVES, R.; GRILO, A.; STEIGER-GARCAO, A. Challenging the Interoperability Between Computers in Industry with MDA and SOA. **Comput. Ind.**, v. 57, n. 8, p. 679–689, 2006.

JÄRVINEN, P. Action Research is Similar to Design Science. **Quality & Quantity**, v. 41, n. 1, p. 37–54, fev. 2007.

JOSUTTIS, N. **Soa in Practice: The Art of Distributed System Design**. [s.l.] O'Reilly Media, Inc., 2007.

KHALFALLAH, M. et al. A cloud-based platform to ensure interoperability in aerospace industry. **Journal of Intelligent Manufacturing**, v. 27, n. 1, p. 119–129, fev. 2016.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [s.l: s.n.].

KOTINURMI, P.; HALLER, A.; OREN, E. Ontologically Enhanced RosettaNet B2B Integration. In: **Global Business: Concepts, Methodologies, Tools and Applications**. [s.l.] IGI Global, 2011. p. 782–808.

LARMAN, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

LUKÁČ, G. et al. A process-oriented service infrastructure for networked enterprises. **Electronic Commerce Research and Applications**, v. 21, p. 1–16, 2017.

MENGE, F. **Enterprise Service Bus**. 2007

MYERSON, J. **Enterprise Systems Integration**. 2th. ed. [s.l.] Auerbach Publications, 2001.

NAUJOK, K.; HUEMER, C. Case Study: Designing ebXML — The Work of UN/CEFACT. In: **Ontologies-Based Business Integration**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 79–93.

NOY, N.; MCGUINNESS, D. **Ontology development 101: A guide to creating your first ontology**Stanford University, , 2001.

O'BRIEN, L.; MERSON, P.; BASS, L. **Quality attributes for service-oriented architectures**. Proceedings of the international Workshop on Systems Development in SOA Environments. **Anais**...2007

OASIS. **Electronic Business using eXtensible Markup Language**, nov. 2001. Disponível em: <http://www.ebxml.org>. Acesso em: 4 nov. 2018

OASIS. **Universal Description Discovery & Integration**, 2004. Disponível em: <http://www.uddi.org/pubs/uddi_v3.htm>. Acesso em: 20 out. 2018

OASIS. **Web Service Business Process Execution Language**, 2007. Disponível em: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>. Acesso em: 18 dez. 2018

OASIS. **Universal Business Language Version 2.2**, mar. 2018. Disponível em: <http://docs.oasis-open.org/ubl/UBL-2.2.html>

OLIVEIRA, R. **Proposta de Catálogo Eletrônico de Processos de Negócio Baseados em UBL para Composição de Aplicações SOA**. Florianópolis: Universidade Federal de Santa Catarina, 2011.

OTERO-CERDEIRA, L.; RODRÍGUEZ-MARTÍNEZ, F. J.; GÓMEZ-RODRÍGUEZ, A. Ontology matching: A literature review. **Expert Systems with Applications**, v. 42, n. 2, p. 949–971, 2015.

PANETTO, H. et al. New perspectives for the future interoperable enterprise systems. **Computers in Industry**, v. 79, p. 47–63, 2016.

PAPAZOGLOU, M. **Web Services & SOA Principles and Technology**. 2nd Ed. ed. [s.l.] Pearson Prentice Hall, 2012.

PICARD, W. et al. Breeding virtual organizations in a service-oriented architecture environment. **SOA infrastructure tools: Concepts and methods**, p. 375–396, 2010.

RABELO, R. J. et al. Smart Configuration of Dynamic Virtual Enterprises. In: CAMARINHA-MATOS, L. M. (Ed.). . **Virtual Enterprises and Collaborative Networks: IFIP 18th World Computer Congress TC5 / WG5.5 - 5th Working Conference on Virtual Enterprises 22--27 August 2004 Toulouse, France**. Boston, MA: Springer US, 2004. p. 193–204.

RABELO, R. J. et al. The Ecolead Plug&Play Collaborative Business Infrastructure. In: CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; OLLUS, M. (Eds.). . **Methods and Tools for Collaborative Networked Organizations**. Boston, MA: Springer US, 2008. p. 371–394.

RABELO, R. J. et al. Virtual Enterprises: Strengthening SMES Competitiveness via Flexible Businesses Alliances. In: NORTH, K.; VARVAKIS, G. (Eds.). . **Competitive Strategies for Small and Medium Enterprises: Increasing Crisis Resilience, Agility and Innovation in Turbulent Times**. Cham: Springer International Publishing, 2016. p. 255–272.

RABELO, R. J.; GUSMEROLI, S. **The Ecolead Collaborative Business Infrastructure for Networked Organizations**. (L. M. Camarinha-Matos, W. Picard, Eds.)Pervasive Collaborative Networks. **Anais**...Boston, MA: Springer US, 2008

REZAEI, R. et al. Interoperability evaluation models: A systematic review. **Computers in Industry**, v. 65, n. 1, p. 1–23, 2014.
RICHARDSON, L.; RUBY, S. **RESTful Web Services**. [s.l.] O'Reilly, 2007.

ROMERO, D.; GALEANO, N.; MOLINA, A. **A Virtual Breeding Environment Reference Model and Its Instantiation Methodology**. (L. M. Camarinha-Matos, W. Picard, Eds.)Pervasive Collaborative Networks. **Anais**...Boston, MA: Springer US, 2008

ROMERO, D.; RABELO, R.; MOLINA, A. Special Issue on Collaborative Networks as Modern Industrial Organizations: Real Case Studies. **International Journal of Computer Integrated Manufacturing**, v. 26, p.

1–2, 2013.

ROMERO, D.; VERNADAT, F. Enterprise information systems state of the art: Past, present and future trends. **Computers in Industry**, v. 79, p. 3–13, 2016.

ROTEM-GAL-OZ, A. **SOA Patterns**. Shelter Island, NY: Manning Pubications Co., 2012.

SANTOS, E. et al. Ontology alignment repair through modularization and confidence-based heuristics. **PloS one**, v. 10, n. 12, p. e0144807, 2015.

SCHRATZENSTALLER, W. M. K.; BALDO, F.; RABELO, R. J. Semantic integration via enterprise service bus in virtual organization breeding environments. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9622, p. 544–553, 2016.

SHI, K. et al. **Integration framework with semantic aspect of heterogeneous system based on ontology and ESB**. The 26th Chinese Control and Decision Conference (2014 CCDC). **Anais**...maio 2014

SHVAIKO, P.; EUZENAT, J. Ontology Matching: State of the Art and Future Challenges. **IEEE Transactions on Knowledge and Data Engineering**, v. 25, n. 1, p. 158–176, 2013.

ULLAH, A.; LAI, R. A Systematic Review of Business and Information Technology Alignment. **ACM Trans. Manage. Inf. Syst.**, v. 4, n. 1, p. 4:1--4:30, 2013.

VAISHNAVI, V.; KUECHLER, B.; PETTER, S. **Design Science Research in Information Systems**. [s.l: s.n.].

VERNADAT, F. B. Technical, semantic and organizational issues of enterprise interoperability and networking. **Annual Reviews in Control**, v. 34, n. 1, p. 139–144, 2010.

W3C. **Web Services Architecture**, fev. 2004. Disponível em: <https://www.w3.org/TR/ws-arch/>. Acesso em: 12 out. 2018

W3C. **Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language**, jun. 2007. Disponível em: <https://www.w3.org/TR/wsdl20/>. Acesso em: 17 out. 2018

WACHE, H. et al. **Ontology-Based Integration of Information - A Survey of Existing Approaches**. 2001

YÜKSEL, M. **A SEMANTIC INTEROPERABILITY FRAMEWORK FOR REINFORCING POST MARKET SAFETY STUDIES**. [s.l.] MIDDLE EAST TECHNICAL UNIVERSITY, 2013.

ZHONG, R. Y. et al. Intelligent Manufacturing in the Context of Industry 4.0: A Review. **Engineering**, v. 3, n. 5, p. 616–630, 2017.

ZHU, W. **Semantic Mediation Bus: An Ontology-based Runtime Infrastructure for Service Interoperability**. 2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops. **Anais**...set. 2012