Pós-Graduação em Ciência da Computação

Uma Abordagem Baseada em Ontologias e Raciocínio Baseado em Casos para Apoiar o Desenvolvimento Distribuído de Software

Por

Rodrigo Gusmão de Carvalho Rocha

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 2015

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rodrigo Gusmão de Carvalho Rocha

Uma Abordagem Baseada em Ontologias e Raciocínio Baseado em Casos para Apoiar o Desenvolvimento Distribuído de Software

TRABALHO APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Prof. Dr. SILVIO ROMERO DE LEMOS MEIRA

RECIFE, 2015

Catalogação na fonte

Bibliotecário Jefferson Luiz Alves Nazareno CRB4-1758

R672a Rocha, Rodrigo Gusmão de Carvalho.

Uma abordagem baseada em ontologias e raciocínio baseado em casos para apoiar o desenvolvimento distribuído de software. / Rodrigo Gusmão de Carvalho Rocha. – Recife: O Autor, 2015.

167 f.: fig., tab.

Orientador: Silvio Romero de Lemos Meira

Tese (Doutorado) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.

Inclui referências e apêndices.

1 Ciência da computação. 2. Engenharia de software. 3. Ontologias. I. Meira, Silvio Romero de Lemos (Orientador). II. Titulo.

005.1 CDD (22. ed.) UFPE-MEI 2015-083

Tese de Doutorado apresentada por **Rodrigo Gusmão de Carvalho Rocha** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Uma Abordagem Baseada em Ontologias e Raciocínio Baseado em Casos para Apoiar o Desenvolvimento Distribuído de Software**" orientada pelo **Prof. Silvio Romero de Lemos Meira** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Vinicius Cardoso Garcia
Centro de Informática/UFPE

_____

Profa. Bernadette Farias Lóscio
Centro de Informática / UFPE

_____

Profa. Juliana de Albuquerque Gonçalves Saraiva
Centro de Ciências Aplicadas e Educação / UFPB

_____

Prof. Gabriel de França Pereira e Silva
Unidade Acadêmica de Garanhuns / UFRPE

_____

Prof. Guilherme Ataíde Dias
Departamento de Ciência da Informação / UFPB

Visto e permitida a impressão.
Recife, 20 de fevereiro de 2015.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Agradecimentos

À Maria Clara, por ter sido, por ser e por que sempre será uma grande luz na minha vida.

À minha mãe por ser a base da minha vida e responsável por quem eu sou hoje.

Ao meu pai, por ser “a pessoa mais forte do mundo”, sendo idealizado por mim como um grande herói.

Ao meu orientador Silvio Meira, por me proporcionar conhecimento e orientação através de conversas, e-mails, aulas, artigos, revistas e posts em seu blog. E também, por servir como exemplo de competência, determinação e inteligência.

À minha família pela união e força em todos os momentos. Família que é o maior alicerce que possuo. Em especial à Carolina Medeiros que esteve presente no decorrer de todo o trabalho, por todo apoio, paciência e atenção, além de tudo que não poderia ser descrito aqui.

Ao meu amigo pessoal e de pesquisa, Ryan Ribeiro, que ajudou a guiar esse trabalho, me servindo desde o início como co-orientador.

Aos meus amigos pessoais que sempre foram também parte da minha formação. Em especial Viviane Nunes e suas revisões, fortalecento o trabalho e sua escrita.

Aos meus colegas de trabalho da Universidade Federal Rural de Pernambuco, que em diversos momentos me apoiaram e me incentivaram para a realização desse trabalho. Aos meus alunos desta instituição que sempre me apoiaram e foram pacientes durante todo processo de execução deste trabalho.

Aos Professores da Universidade Federal de Pernambuco, pela educação e ensinamentos transmitidos aos seus alunos, inclusive a mim.

A todos do Centro de Informática (CIn). Um lugar que eu tenho maior orgulho de fazer parte, como aluno e pesquisador. Local onde eu tenho inspiração e admiração.

A pessoa que fez com que uma doença grave não fosse apenas motivo de tristeza e redenção. Mais uma vez ela me prova que nada é fácil e nem por isso deixaremos de ser felizes e lutar pelo melhor. É para você minha mãe, te amo.

Dedico.

RESUMO

Como reflexo da globalização, empresas de software começaram a distribuir seus processos de desenvolvimento em lugares diferentes, criando o Desenvolvimento Distribuído de Software (DDS). A distribuição de equipes no desenvolvimento de software trouxe uma série de novos desafios, tais como, comunicação e compartilhamento de informação. Um outro problema é que pela falta de conhecimento dessas organizações, estas resolvem seus problemas de forma independente e de diversas maneiras diferentes, cada uma com suas práticas, algumas mais e outras menos eficientes, onde as melhores práticas são pouco difundidas na comunidade DDS. A utilização de conceitos e técnicas da Inteligência Artificial é bastante utilizado para aperfeiçoar o funcionamento de alguns sistemas e processos. Neste caso, este trabalho expõe três conceitos fundamentais: 1) o uso de Ontologias que permite a formalização do conhecimento de um domínio. Em ambientes distribuídos, a utilização de Ontologias traz alguns benefícios como compreensão uniforme das informações entre as equipes e facilidade de comunicação. Adicionalmente, conceitos como 2) Raciocínio Baseado em Casos (RBC) e 3) Processamento de Linguagem Natural (PLN) também podem ser utilizados para se tentar fazer um melhor uso de experiências já vivenciadas com intuito de mitigar possíveis problemas. Desta maneira, o objetivo desta pesquisa é propor um mecanismo baseado em ontologias que possa extrair e recomendar informações relevantes para apoiar decisões em projetos de software com times distribuídos sendo de fato uma solução viável na resolução de problemas nesse contexto. Para levantamento do estado da arte dessa pesquisa, foi realizado um mapeamento sistemático, que identificou 51 estudos primários. Estes apresentam técnicas, modelos e ferramentas que utilizam ontologias para auxiliar equipes distribuídas, bem como ontologias propostas nesse contexto. Dessa forma, os resultados principais deste trabalho são: 1) uma ontologia específica para times distribuídos de software, 2) sua ferramenta de manipulação e acesso à informação e 3) o sistema de RBC utilizando PLN. Assim, através dos resultados constatados nos testes realizados, é possível afirmar que houve uma taxa de sucesso de 91,67% na recomendação das soluções para possíveis problemas. Essa abordagem auxilia os times distribuídos recomendando técnicas ou melhores práticas para evitar ou solucionar os desafios encontrados.

Palavras-chave: Desenvolvimento Distribuído de Software. Ontologias. Raciocínio Baseado em Casos. Mapeamento Sistemático. Processamento de Linguagem Natural.

ABSTRACT

As an effect of globalization, software companies began to distribute the development process along different places, which led to the rise of Distributed Software Development (DSD). The multisite nature of DSD brings with it many new challenges, such as communication issues and sharing information efficiently. The fact that these companies have a tendency to face these challenges in an individual and isolated fashion poses another important problem, this way, good practices are thus not widespread in the DSD community. In other contexts, concepts and techniques from Artificial Intelligence are heavily used for improving how some systems and processes work. This work makes use of three of these concepts: Ontologies, Case-based Reasoning (CBR) and Natural Language Processing (NLP). Ontologies allow the formalization of a domain's knowledge, which applied to distributed development environments will bring benefits like a uniform comprehension of the information shared by the teams and ease of communication. CBR and NLP also have an important role in the attempt of making better use of past experiences, with the purpose of mitigating potential problems. An approach based on Ontologies and Case-Based Reasoning is proposed: an access tool to the ontology and a CBR tool that utilizes PLN, whose purpose is to support the entire software development process for distributed teams by recommending techniques or best practices for avoiding or solving potential challenges that may be faced. The state-of-the-art survey for this research consisted in a systematic mapping that retrieved 51 primary researches. These researches present techniques, models and tools that use ontologies for supporting distributed teams, as well as the proposed ontologies for this context. Thus, the main results of this work are: 1) a specific ontology for distributed teams of software, 2) its manipulation tool and access to information and 3) the RBC system using PLN. Thus, by the results observed in the tests, it is clear that there was a 91.67% success rate on the recommendation of solutions to possible problems. This approach helps teams distributed recommending techniques or best practices to avoid or solve the challenges encountered.

Keywords: Distributed Software Development. Ontologies. Case-Based Reasoning. Natural Language Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

Sigla	**Significado**
ABES	ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE
API	APPLICATION PROGRAMMING INTERFACE
DAML	*AGENT MARKUP LANGUAGE*
DDS	DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE
DGS	DESENVOLVIMENTO GLOBAL DE SOFTWARE
DKD	DISTRIBUTED KNOWLEDGE DEVELOPMENT
DKDOnto	DISTRIBUTED KNOWLEDGE DEVELOPMENT ONTOLOGY
DL	DESCRIPTION LOGIC
ES	ENGENHARIA DE SOFTWARE
GSD	GLOBAL SOFTWARE DEVELOPMENT
GWT	GOOGLE WEB TOOLKIT
OIL	*ONTOLOGY INFERENCE LAYER*
OWL	*WEB ONTOLOGY LANGUAGE*
PCFG	*PROBABILISTIC CONTEXT FREE GRAMMAR*
PLN	PROCESSAMENTO DE LINGUAGEM NATURAL
RBC	RACIOCÍNIO BASEADO EM CASOS
RDF	RESOURCE DESCRIPTION FRAMEWORK
SHOE	*SIMPLE HTML ONTOLOGY EXTENSIONS*
SPARQL	SPARQL PROTOCOL AND RDF QUERY LANGUAGE
TI	TECNOLOGIA DA INFORMAÇÃO
W3C	*WORLD WIDE WEB*
XML	EXTENSIBLE MARKUP LANGUAGE

SUMÁRIO

1 Introdução

Nas últimas décadas, empresas situadas entre as diversas áreas de atuação vêm tornando o uso do software algo vital para seus negócios. Geralmente elas utilizam a tecnologia em busca do gerenciamento e do controle de suas próprias atividades, obtendo com a automação de seus sistemas, maiores lucros e ganhos na competitividade junto à própria concorrência [Herblesb, 2007]. Desse modo, o software conseguiu ao longo do tempo, obter espaços maiores dentro das empresas, uma vez que torna o seu funcionamento mais eficiente de acordo com as exigências de mercado, deixando-as cada vez mais dependentes dessas novas tecnologias.

O diferencial de competitividade trazido com os novos softwares é notório, e dentro dessa perspectiva, o número de empresas que desenvolve sistemas e o que as utilizam cresceu significativamente nas últimas décadas. Isto pode ser observado a partir da pesquisa realizada pela ABES (2013), a qual apresentou o mercado mundial de software e serviços em 2012 movimentando aproximadamente US$ 1.023 bilhões, o que resulta em um crescimento de 8,7% com relação ao ano anterior.

Dessa forma, a partir do crescimento acentuado na indústria de software, juntamente com os reflexos da globalização, que contribuíram para uma aceleração na produção em todos os setores da economia, se percebeu então o Desenvolvimento Distribuído de Software (DDS). Pois, o mercado de tecnologia sentiu a necessidade em distribuir suas etapas e processos de desenvolvimento do software em lugares distintos [Herbsleb, 2007].

A concepção de DDS exige alguns requisitos fundamentais entre os participantes do processo de desenvolvimento para que se possa atingir resultados satisfatórios de forma eficiente nos projetos, traçados inicialmente. Entre esses aspectos, pode-se dizer que o DDS demanda de boa comunicação entre os responsáveis pelo *design* de partes do software que serão desenvolvidos nas diversas localidades, por diferentes especialistas [Cataldo et al., 2006].

Dentro deste contexto, o diferencial da abordagem DDS é também onde se encontram as maiores dificuldades, as quais são agravadas não somente pela distância física, mas também pela diferença cultural e temporal entre os integrantes. Esta abordagem, então, herda os mais diversos problemas existentes em qualquer processo de produção de software, inclusive o tradicional, e por diversas razões acrescenta novas dificuldades a ele [Prikladnicki, 2009].

Embora o crescimento e demanda de software seja visível através dos números apresentados inicialmente, o seu desenvolvimento não é uma tarefa simples. Segundo o relatório do Standish Group (2013) para o ano de 2010, apenas 37% dos projetos de software obtiveram sucesso, 42% tiveram problemas de atraso e quantidade inferior de funcionalidades do que o planejado e os outros 21% dos projetos ou falharam por completo em sua expectativa final, ou nunca foi utilizado pela empresa que os adquiriu.

Todos os projetos para produção de software, seja qual for o sistema a ser desenvolvido, possuem etapas e são compostos de atividades temporárias, com objetivos, custos e responsabilidades bem definidos. Tais características se tornam ainda mais complexas quando se trata do contexto distribuído. Desse modo, existe uma maior necessidade de desenvolver métodos que auxiliem o desenvolvedor na identificação prematura dos fatores que levam ao fracasso os projetos distribuídos [Rocha, 2011].

Durante o DDS, Herbsleb (2007) e Parviainen e Tihinen (2011) afirmam que alguns desafios ainda são comuns, como: dificuldade na comunicação, compreensão uniforme das informações do

projeto sobre quem desenvolve o que, como e com quem, coordenação do projeto, erros de interpretação de tarefas e sincronização dos esforços entre as equipes distribuídas.

Os principais desafios da Engenharia de Software para essa abordagem distribuída são causados pela singularidade do ambiente de trabalho e não estão relacionados somente aos problemas técnicos (por exemplo, problemas com as tecnologias envolvidas), que os gerentes de projeto utilizam para superar tais dificuldades. Os desafios que essa abordagem traz são problemas em muitos casos, já conhecidos no desenvolvimento tradicional de software, porém em ambientes distribuídos, estes são mais evidenciados. Os desafios são trazidos pela distância geográfica, temporal e cultural entre os membros da equipe global de software. Carmel (1999) afirma a existência de cinco fatores que podem levar ao fracasso de uma equipe distribuída:

(1) comunicação ineficiente;

(2) falta de coordenação;

(3) dispersão geográfica;

(4) perda do espírito de equipe;

(5) diferenças culturais chamadas de forças centrífugas.

Komi-Sirvo e Tihinen (2002) e Parviainen e Tihinen (2011), baseados em suas pesquisas também apresentam algumas áreas problemáticas e fatores associados quando o desenvolvimento é distribuído, São eles:

(1) ambientes e ferramentas de desenvolvimento;

(2) comunicação;

(3) engenharia de requisitos;

(4) gerenciamento de projetos;

(5) diferenças culturais;

(6) tempo e orçamento acima do estimado, entre outros.

Inúmeras empresas tentam adotar métodos diferentes para que possam lidar da melhor forma com as variáveis do contexto distribuído. Contudo, é necessário que existam mais estudos e que novas técnicas, metodologias e ferramentas sejam concebidas e eficientemente aplicáveis a ambientes distribuídos [Lopes et al, 2003] [Prikladnicki, 2009]. As empresas que utilizam esse modelo de desenvolvimento, sentem extrema necessidade em definir quais metodologias e estratégias irão usar para o projeto em andamento e quais suas respectivas ferramentas e práticas, mas não possuem informações que sejam capazes de determinar por exemplo, qual a metodologia é mais adequada para determinado tipo de projeto, quais são os membros mais interessantes para se alocar, quais melhores ferramentas ou quais as boas práticas recomendadas para tal projeto.

Dessa forma, uma possível alternativa visando minimizar os problemas mencionados acima, seria a utilização do conceito de ontologias. De acordo com Freitas (2005), a utilização de ontologias permite estruturar de forma mais clara o conhecimento de um domínio específico. Além disso, a utilização de um domínio pré-definido, através de ontologias, permite uma definição mais precisa e provê consistência semântica à informação. Assim, ontologias representam um excelente mecanismo para formalizar um padrão de compartilhamento de informações entre pessoas ou agentes de software. Isso possibilita a afirmação que o uso de ontologias pode ser uma alternativa para auxiliar projetos distribuídos no sentido da disponibilização de informações a cerca de um projeto de software, o que

permite que membros daquele projeto possam utilizar informações com objetivo de obter melhores resultados em suas atividades.

Dessa maneira, é possível afirmar que a inexistência de um modelo público, formal e normalizado (ontologia) das informações do projeto dificulta não somente a comunicação no decorrer do projeto, mas também o entendimento dos membros e dos artefatos. O que pode ser agravado quando a cultura e demais características dos membros das equipes, como o idioma, tradições e fusos horários, são distintas entre si. Da mesma forma, a ausência de ferramentas que possam explorar esse modelo formal, a fim de extrair informações importantes sobre o projeto e sobre as formas de evitar ou mitigar os desafios e mais ainda, as formas de se melhorar o trabalho dentro daquele contexto, não possibilitam a evolução do trabalho de produção de maneira adequada.

1.1 Problematização e Justificativa

Com a descentralização das empresas e a produção de software acontecendo de forma distribuída em diversas filiais espalhadas pelo globo, o desenvolvimento de software torna-se mais complexo, exigindo que as organizações busquem maneiras eficazes para guiar seus projetos utilizando ferramentas, técnicas e modelos, que consigam satisfazer suas características de mercado e suas necessidades [Prikladnicki e Audy, 2007].

Para reduzir as complexidades trazidas com a aplicação do DDS, o uso de teorias e práticas advindas da Inteligência Artificial pode ser útil, como é o caso do conceito de Web Semântica através de Ontologias ou o conceito de sistemas de Raciocínio baseado em Casos. Wongthongtham *et al.* (2006) asseguram que a utilização de ontologias é um tipo de paradigma para Engenharia de Software e que pode ser aplicado principalmente para conceder semântica para ferramentas auxiliares, favorecendo uma forte comunicação baseado nas informações, centralização e disponibilidade de informação e também, fornecer uma base de informações para que agentes de software capazes de formar um ambiente automatizado possam apoiar o DDS.

Smite e Borzovs (2009) afirmam que os responsáveis pelo projeto devem estar cientes de que maneira a organização do time e do projeto pode ser feita, no sentido de que as formas de trabalho das equipes possam levá-los a obter um melhor desempenho. Desse modo, é necessário que haja maior conhecimento dos responsáveis pelo projeto a respeito de como e de quais recursos (atividades, membros, ferramentas, entre outros) podem ser melhor utilizados, para tentar amenizar os problemas que foram encontrados ou que serão encontrados no decorrer do projeto. Como já citado, esse conhecimento pode ser contemplado com o emprego das Ontologias.

Dessa maneira, essa abordagem com ontologias implica também em utilizar conhecimentos passados permitindo que os projetos de software tentem evitar problemas e situações indevidas durante sua execução. Através da aplicação de ontologias, responsáveis pelo projeto podem consultar a base em busca de informações sobre soluções para possíveis problemas, bem como identificar melhores práticas para o desenvolvimento daquela configuração de projeto. Ou seja, projetos antigos com a mesma configuração ou similares, são úteis para consultas, visando solucionar problemas existentes ou problemas que podem vir a acontecer no projeto atual.

Em outro contexto, projetos de software passam por muitas experiências, estas experiências possuem características próprias que trazem conhecimento influente em um ambiente de DDS. Como já citado, muitos desafios existem nessa abordagem de desenvolvimento, assim como boas práticas. Muitas dessas características se repetem em experiências passadas, podendo ser aproveitadas e reutilizadas como possíveis soluções em casos e desafios semelhantes em outros projetos de DDS. Dessa maneira, se assemelhando aos casos utilizados nos sistemas de RBC.

Dessa maneira, Urnau, Kipper e Frozza (2014) argumentam que o RBC permite sua escalabilidade, através da harmonização/integração com outras técnicas e normas já conceituadas. Ampliando assim se necessário a gama recursos para apuração das melhores informações a serem utilizadas na resolução de problema. Isto permite que, um sistema RBC possa ser utilizado também com ontologias.

A utilização de instâncias de uma ontologia como forma de representar casos é um recurso interessante para sistemas baseados em conhecimento. Além disso, para a utilização de ontologias para a representação é importante que exista um meio que proporcione executar consultas que permitam a identificação e recuperação de instâncias similares [Mustapha et al. 2010].

Dentre as vantagens da representação de casos com ontologias, destaca-se a capacidade de representação semântica do caso, além da reutilização da ontologia para outros sistemas baseados em conhecimento. Através da construção de ontologias para representação de casos é possível representar conhecimento semântico associado aos casos, que, além de viabilizar a recuperação de casos com uma melhor precisão, possibilita a reutilização da ontologia em outros sistemas baseados em conhecimento [Mendes, 2013].

Mendes (2013) garante que o uso de ontologias para representação de casos ajuda na construção de um método inteligente para a identificação e recuperação de casos. O que é necessário para a resolução de problemas através do RBC, visto que ao utilizar semântica, é possível obter valores de similaridade mais precisos na comparação de casos, proporcionando uma melhor efetividade na recuperação das soluções.

Este estudo situa-se exatamente nas lacunas supracitadas, sendo relevante no sentido de estudar a utilização de Ontologias para o DDS. Onde a proposta é uma ontologia de domínio e ferramentas que utilizem técnicas específicas de Inteligência Artificial já conhecidas para que possam explorar as informações que estão semanticamente inseridas que dizem respeito aos projetos atuais bem como a projetos antigos. Dessa forma, buscar auxílio para a tomada de decisões ou para mitigar riscos e dirimir desafios, bem como fazer uso das melhores práticas para o projeto.

Portanto, este trabalho define como o Problema de Pesquisa, a seguinte questão/problema:

- De que maneira uma abordagem baseada em Ontologias e RBC pode extrair conhecimento a partir de projetos distribuídos já executados de forma que novas equipes e novos projetos sejam auxiliados?

Consequentemente, para tentar atender devidamente esta questão, são formadas as seguintes hipóteses que reportam ao objetivo inicial apresentado na seção seguinte.

- Hipótese Nula - $\mathbf{H_0}$: “a abordagem não é capaz de recomendar soluções de contextos similares que possam auxiliar os times distribuídos
- Hipótese Alternativa ou Hipótese de Pesquisa - $\mathbf{H_1}$: “a abordagem é capaz de recomendar soluções de contextos similares que possam auxiliar os times distribuídos”.

Para responder à questão colocada e verificar as hipóteses definidas, essa abordagem utilizando Ontologias e RBC foi contruída, conforme é discutido no capítulo 4. O resultado desta pesquisa tem foco em todo processo de desenvolvimento de software com equipes distribuídas, nas perspectivas técnica e organizacional, permitindo que o projeto receba auxílio através de informações que são retiradas a partir das experiências de outros projetos, o que envolve práticas e decisões tomadas por diretores, gerentes e integrantes das equipes.

1.2 Objetivos

O objetivo geral deste trabalho é propor um mecanismo que possa extrair e recomendar informações relevantes para apoiar decisões em projetos de software com times distribuídos, demonstrando que é uma solução viável na resolução de problemas nesse contexto. Permitindo o apoio a resolução dos desafios e limitações desse ambiente, mitigando riscos, minimizando os custos inerentes aos projetos, melhorando a comunicação entre os participantes, disponibilizando informações sobre recursos, técnicas e metodologias, e também quais os membros mais capazes para determinadas atividades.

Para a consecução do objetivo geral proposto, foram definidos os seguintes objetivos específicos:

- Mapear sistematicamente o uso de ontologias no DDS, para discussão e avaliação das ontologias existentes desse contexto;

- Formalizar e desenvolver uma ontologia de domínio para o contexto de DDS, povoada inicialmente com base na descrição de projetos em artigos científicos;

- Demonstrar a utilização de uma ferramenta para manipulação e extração de informações da ontologia proposta para apoiar decisões e definições no decorrer do projeto;

- Prover ao usuário experiências passadas armazenadas nas bases de dados com maior similaridade possível ao novo desafio, através de uma ferramenta de raciocínio baseado em casos;

- Povoar base de experiências passadas (casos) provenientes de artigos científicos e experiência de profissionais da indústria para o contexto de DDS;

- Auxiliar equipes de DDS no processo de aquisição de soluções a problemas reduzindo as dificuldades e tempo empregados no desenvolvimento;

1.3 Contribuições Acadêmicas

Durante a execução deste trabalho, 13 artigos referentes aos conceitos envolvidos e artefatos desenvolvidos foram publicados em eventos e revistas nas áreas espefícicas e outros ainda estão em processo de avaliação para publicação. As publicações já confirmadas são apresentadas a seguir.

- Rocha, R. G. C ; Azevedo, R., Rodrigues, Cleyton Mário de Oliveira., Nascimento Filho, D. C., Meira, S. R. L. A System based on Ontology and Case-Based Reasoning to Support Distributed Teams. In: 12th International Conference on Information Technology : New Generations (ITNG), 2015, Las Vegas, Nevada, USA. Proceedings 12th International Conference on Information Technology : New Generations, 2015.
- Rocha, R. G. C., Azevedo, R. R., Sousa, C. Y., Tavares, E., Meira, Silvio. A Case-Based Reasoning System to support the Global Software Development. In: 18th International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES), 2014, Gdynia, Poland. Proceedings of Conference in Knowledge Based and Intelligent Information and Engineering Systems.
- Rocha, R. G. C., Azevedo, R. R., Meira, Silvio. A Proposal of Ontology-based System for Global Software Development. In: Software Engineering and Advanced Applications (SEAA), 2014, Verona, Italy. Proceedings of Software Engineering and Advanced Applications, 2014.

- Rocha, R. G. C., Azevedo, R. R. Cassimiro, Dimas. , Meira, Silvio. DKDs: An Ontology-based System for Distributed Teams. In: 26th International Conference on Software Engineering and Knowledge Engineering, 2014, Vancouver, Canada. Proceedings of 26th International Conference on Software Engineering and Knowledge Engineering, 2014.
- Rocha, Rodrigo G. C., Azevêdo, Ryan., Costa, Catarina; Duarte, Marcos; Fechine, João Paulo; Freitas, Fred; Meira, Silvio. (2014). An Ontology-based System to Support Distributed Software Development. Journal of Communication and Computer (JCC).
- Rocha, Rodrigo G. C., Azevedo, Ryan., Costa, Catarina; Duarte, Marcos; Fechine, João Paulo; Freitas, Fred; Meira, Silvio. (2013). An Ontology-based System to Support Distributed Software Development. In The Eighth International Conference on Software Engineering Advances (ICSEA) 2013. October 27 - November 1, 2013 - Venice, Italy.
- Rocha, Rodrigo G. C., Azevedo, Ryan Ribeiro, Meira, Silvio. Extração de Conhecimento e Experiências em Equipes de Desenvolvimento Distribuído de Software. Workshop de Teses e Dissertações (WTDSoft) do Congresso Brasileiro de Software (CBSoft). Brasília – DF, 2013.
- Rocha, Rodrigo G. C., Azevedo, Ryan Ribeiro. Mendonca, Sergio, Borges, Alex, Costa, Catarina, Soares, Sergio, Meira, Silvio. Ontologies in Global Software Development. The 19th International Conference on Distributed Multimedia Systems. Seafront, United Kingdom, 2013.
- Alves, Ana Raquel, Silva, Israel Felipe, Fernandes, Renan L., Rocha, Rodrigo G. C., Burity, Thaís, Azevedo, Ryan. Meira, Silvio. Avaliação de Ontologias de Domínio para o Desenvolvimento Distribuído de Software. Escola Regional de Informática (ERIPE-SBC), Garanhuns – PE. 2013.
- Borges, A. N. ; Rocha, R. G. C. ; Costa, C. ; Soares, S. ; Oliveira, H. T. A. ; Meira, S. R. L. . Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study. In: International Conference on Evaluation and Assessment in Software Engineering (EASE), 2013, Recife. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2013.
- Rocha, R. G. C. ; Meira, S. R. L. . DsdK: an Ontology-based System to Explore Distributed Software Development Experiments. In: International Conference on Global Software Engineering, 2012, Porto Alegre – RS. Proceedings of International Conference on Global Software Engineering, 2012.
- Borges, A. N. ; Rocha, R. G. C. ; Costa, C. ; Azevedo, R. R. ; Silva, F. Q. B. . Ontologies supporting the Distributed Software Development: a Systematic Literature Review. In: International Conference on Global Software Engineering (ICGSE), 2012, Porto Alegre. Procedings of International Conference on Global Software Engineering (ICGSE), 2012.
- Rocha, R. G. C. ; Costa, C. ; Rodrigues, C. M. O. ; Azevedo, R. R. ; Farias Junior, I. H. ; PRIKLADNICKI, R. ; MEIRA, S. R. L. Collaboration Models in Distributed Software Development: a Systematic Review. CLEI Electronic Journal, v. 1, p. 1-12, 2011.

1.4 Organização do trabalho

Este trabalho possui seu desenvolvimento definido em 6 capítulos e está organizado da seguinte maneira:

No Capítulo 2 é apresentado o embasamento teórico necessário para entender este trabalho, fornecendo uma visão geral do Desenvolvimento Distribuído de Software, bem como o conceito de Ontologias, de mapeamento sistemático, que é uma especificidade da engenharia de software empírica, e ainda, o conceito de Sistema de Raciocínio Baseado em Casos e de PLN, todos tradados como conceitos necessários e inerentes ao entendimento dessa abordagem.

No Capítulo 3 são apresentados os métodos para a realização desse trabalho, como o mapeamento sistemático sobre os recursos existentes baseados em ontologias no ambiente distribuído, bem como seus resultados, suas conclusões e avaliações feitas, é apresentado também o *survey* realizado com alguns profissionais da indústria para povoamento inicial da ontologia e criação dos casos utilizados pelo sistema RBC. E também, a realização de leitura e extração de informações em artigos científicos para a formação dos casos e também povoamento inicial da Ontologia.

No Capítulo 4, por sua vez, apresenta o mecanismo de extração de informação, primeiramente com a formalização e modelagem do conhecimento do Desenvolvimento Distribuído de Software através da Ontologia proposta DKDOnto, em seguida pela ferramenta DKDs que permite acesso a Ontologia e as instâncias nelas contidas, bem como o CBRs-DKD, ferramenta baseada em raciocínio baseado em casos. É apresentado também nesse capítulo, na última seção os testes realizados para avaliação do mecanismo proposto.

No Capítulo 5 são apresentados os trabalhos relacionados, a descrição de cada um deles, bem como a comparação entre eles e esta tese. E, por fim, no Capítulo 6 são trazidas as considerações finais da presente pesquisa, incluindo as principais contribuições, limitações e as perspectivas de trabalhos futuros.

2. Fundamentação Teórica

Este capítulo disserta acerca dos principais conceitos envolvidos na pesquisa e, possui como principal objetivo fornecer uma percepção ampliada sobre o DDS, o conceito de Ontologias, e de uma método que é uma especificidade da Engenharia de Software Empírica, que é o Mapeamento Sistemático da Literatura. Nestes termos, serão evidenciadas apreciações a respeito dos sistemas de RBC, e, nesses ditames, teorias sobre o PLN.

2.1 Desenvolvimento Distribuído de Software

É notável que a evolução do software tenha ocorrido de forma acelerada, de maneira que diversas áreas do conhecimento têm reconhecido a sua importância como posição estratégica perante o mercado. Nas últimas décadas a globalização dos negócios impactou também a indústria de TI [Carmel e Tjia, 2005] [Carmel e Prikladnicki, 2013].

Alguns autores [Ramasubbu, Cataldo, Balan e Herbsleb, 2011], [Herbsleb e Moitra, 2001], afirmam que as forças econômicas transformaram mercados nacionais em globais. Estas transformações não alteraram apenas o marketing de vendas, mas também, a forma como os produtos são concebidos, construídos, testados e entregues aos clientes finais.

Como consequência disso, o custo dos profissionais cresceu, enquanto as empresas competiam almejando a constratação destes [Karolak, 1998]. Nos Estados Unidos, país responsável por grande parte da demanda por software, as cotas de vistos de trabalho temporário para área de tecnologia se esgotavam antes do final do ano fiscal, gerando uma maior escassez de profissionais [Carmel, 1999]. Nesse contexto, com um número de funcionários insuficiente e altos custos para contratação, a disponibilidade de recursos equivalentes em outras localidades a um custo mais acessível, tornou-se um grande atrativo para as empresas que desenvolvem software, possibilitando um conceito relativamente novo para o mundo da ES, o Desenvolvimento Distribuído de Software (DDS) [Estler HC *et al*., 2014] . A Figura 2.1 ilustra equipes globais distribuídas em vários países e continentes diferentes.

Figura 2.1 – Desenvolvimento Global de Software e Equipes Globais.

Fonte: Prikladnicki, 2003.

Em algumas de suas obras Herblesb (2007) e Herblesb et. al. (2001) afirmaram que já não é mais incomum projetos de software possuírem equipes de desenvolvimento distribuídas em mais de um local, às vezes até em mais de um continente. A crescente busca por maior competitividade tem levado

às empresas a adotarem o DDS, onde diferentes partes do software são desenvolvidas em localidades distintas. Tentando realizar desenvolvimento a baixo custo, empresas têm atravessado fronteiras, formando um mercado global.

O autor supracitado relata ainda que o software se tornou um componente estratégico para diversas áreas de negócio, mais especificamente na área de ES, criando novas formas de cooperação e competição que vão além das fronteiras dos países. Desenvolver software no mesmo espaço físico, na mesma organização ou ainda, no mesmo país, tem se tornado cada vez mais oneroso e menos competitivo [Audy e Pkladnicki, 2007]. O avanço da economia, a sofisticação dos meios de comunicação e a pressão por custos têm incentivado o investimento maciço no DDS.

Segundo Brooks (1978), o software em si possui muitas características que o fazem unicamente apropriado para uma abordagem distribuída e o diferencia de outros setores, pois pode ser replicado, transmitido, corrigido e usado sobre distâncias ilimitadas com custo virtual zero. Contudo, ele também é sujeito a mudanças é intangível e pode tornar-se potencialmente complexo. Visto isto, o DDS tem se apresentado nos últimos anos como uma alternativa para o desenvolvimento de software. É um fenômeno que vem crescendo desde a última década e que tem sido caracterizado pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, localizados em cidades ou países diferentes [Meyer, 2006] [Ramasubbu, Cataldo, Balan e Herbsleb, 2011].

Karolac (1998) alega que, quando a distância física entre os atores em um ambiente de DDS envolve mais de um país, o desenvolvimento é considerado global, chamado de Global Software Development (GSD), também conhecido Desenvolvimento Global de Software (DGS). O DGS passa a existir quando existem equipes globais de desenvolvimento de software (Global Software Teams). Prikladnicki e Carmel (2013) define como sendo um grupo de pessoas distribuídas em dois ou mais países, colaborando ativamente em um projeto comum.

De acordo com Prikladnicki e Audy (2004), a distância física entre os membros de um determinado grupo de *stakeholders*[1] (equipe de projeto, por exemplo) é chamada de nível de dispersão. Assim, nível de dispersão diz respeito à distância física entre os atores envolvidos em um determinado projeto ou fase deste. Em sua pesquisa, Prikladnicki (2003) definiu quatro situações para o tipo de distância física e suas características principais, são elas:

- Mesma localização física: situação onde a empresa possui todos os stakeholders em um mesmo local.
- Distância nacional: situação caracterizada por ter os stakeholders localizados dentro de um mesmo país, em cidades, estados e em regiões diferentes.
- Distância continental: situação caracterizada por ter os stakeholders localizados em países diferentes, necessariamente dentro do mesmo continente.
- Distância global: situação caracterizada por ter os envolvidos localizados em países diferentes e em continentes diferentes, formando muitas vezes uma distribuição global.

Conforme Evaristo et al. (2004), quanto maior o nível de dispersão, maior é a dificuldade de monitorar o comportamento de diferentes grupos. As Figuras 2.2, 2.3, 2.4 e 2.5 exibem esses quatro níveis de dispersão. A Figura 2.2 representa a mesma localização física, a Figura 2.3 é a distância

[1] Todos os envolvidos no processo de desenvolvimento de software

nacional. Por fim, a Figura 2.4 representa a distância continental, bem como a Figura 2.5 a distância global.

Figura 2.2 – Mesma localização Física

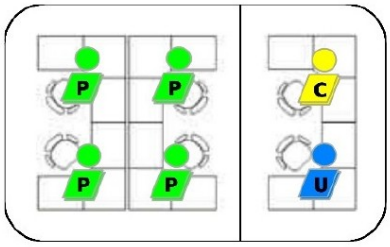


Figura 2.3 – Distância nacional

Figura 2.4 – Distância continental

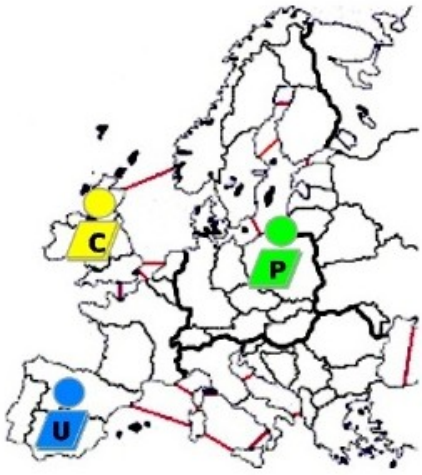
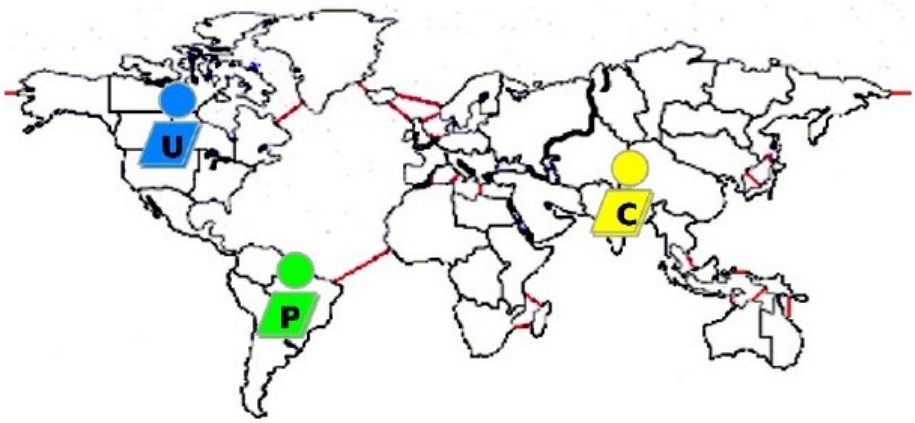


Figura 2.5 – Distância global



Fonte: Prikladnicki, 2003.

Nestas imagens, podem ser vistos três tipos de *stakeholders* representados pelos números: U, C e P. É possível interpretar esses *stakeholders*, da presente forma: U são os usuários do sistema, podem fazer parte ou não do cliente, são responsáveis por utilizar na prática o sistema; C são os clientes, responsáveis por ditar as regras, fechar contrato e entender da necessidade; e por fim o P, consiste nos programadores, membros da equipe contratada que seriam os responsáveis por implementarem o que foi passado pelos analistas.

2.1.1. Motivações / Incentivos para o Desenvolvimento Distribuído

O DDS não é uma abordagem nova. Desde o final da última década que esse modelo é estudado por empresas e pesquisadores. A literatura menciona diversas razões que levam essas empresas a distribuírem seus processos de desenvolvimento, assim como, os desafios que esse cenário distribuído pode apresentar. Apesar dos desafios inerentes ao contexto distribuído, o DDS é uma estratégia de

desenvolvimento que está em ascensão na indústria [Herbsleb et al., 2001, Battin et al., 2001, Ebert e De Neve, 2001] [Orsoletta e Prikladinicki, 2013] [Estler HC *et al*., 2014].

Consoante a Carmel (2001) e Dantas (2003), existem mais de 50 (cinquenta) nações envolvidas em projetos de desenvolvimento colaborativo de software internacionalmente, o que demonstra uma crescente tendência de mercado. Grandes empresas como Motorola Fujitsu [Ayoma, 1998; Gao et al, 1999], [Battin et al, 2001], Alcatel [Ebert e De Neve, 2001], Bell Labs [Mockus e Meiss, 2001] descrevem suas experiências com o Desenvolvimento Global de Software em seus projetos.

Os benefícios esperados em ambientes distribuídos, como a possibilidade de desenvolvimento de alta velocidade através da utilização de equipes em diferentes fuso horários, a contratação de pessoas mais capacitadas, o menor desenvolvimento dos custos e da capacidade de responder às necessidades locais dos clientes, superam os riscos envolvidos [Komi-Sirvo e Tihinen, 2005] [Estler HC *et al*., 2014]. Estes, porém, não são os únicos fatores que contribuem para que os processos de desenvolvimento das empresas sejam distribuídos, conforme alega Prikladnicki (2009), existem também:

- A necessidade de aumentar os recursos quando a disponibilidade de mão-de-obra local estiver escassa;
- Vantagem de estar perto do mercado local, incluindo o conhecimento dos clientes e as condições do ambiente;
- A rápida formação de corporações e times virtuais visando explorar as oportunidades de mercado;
- Pressões relativas ao *time-to-market* (tempo de entrada do produto no mercado), com o uso de diferentes fuso-horários permitindo inclusive o desenvolvimento 24x7 (vinte e quatro horas, sete dias por semana).

Prikladnicki e Carmel (2013) confirmam as mesmas razões que Evaristo *et al.* (2005) apontaram, como também, outras que motivam a adoção do DDS, tais como:

- Necessidade de profissionais qualificados em áreas especializadas;
- Incentivos fiscais para o investimento de pesquisas em informática;
- Disponibilidade de mão-de-obra especializada e de custos reduzidos em países em desenvolvimento;
- Realização de etapas do desenvolvimento de software perto dos clientes;
- Redução dos prazos de entrega proporcionada pelo desenvolvimento *round-the-clock* (desenvolvimento contínuo ou duradouro durante às vinte e quatro horas do dia);
- Formação de organizações e de equipes virtuais para aproveitar oportunidades de mercado;

- Necessidade de integrar recursos resultantes de aquisições e fusões organizacionais.

Diante desse quadro significativo, Carmel (1999) refere a existência de seis fatores que podem levar uma equipe ao sucesso: (1) infra-estrutura de comunicação; (2) arquitetura do produto; (3) construção de uma equipe; (4) metodologia de desenvolvimento; (5) tecnologia de colaboração; e, (6) técnicas de gerência, chamados de forças centrípetas, já que as mesmas podem inferir diretamente para que o projeto obtenha sucesso.

2.1.2. Fatores que Afetam o Desenvolvimento Distribuído de Software

Audy e Prikladnicki (2007) explicam que o desenvolvimento tradicional de software sempre se apresentou de forma complexa e que o Desenvolvimento Distribuído acrescentou outros desafios ao processo, ao adicionar fatores como dispersão física, distância temporal e diferenças culturais. Como descrito anteriormente, com o surgimento do DDS, novos desafios foram acrescentados ao desenvolvimento de software [da Silva et al, 2010] [Khan, Niazi e Ahmad, 2011] [Jimenez, Piattini e Vizcaino, 2009].

A Figura 2.6 apresenta o resultado da pesquisa realizada por Komi-Sirvo e Tihinen (2005), onde os participantes classificaram dentre as oito áreas sugeridas as que possuem mais problemas no contexto distribuído. Os maiores índices foram atribuídos ao ambiente e ferramentas de comunicação e comunicação em geral, sendo seguidas pelas áreas de conhecimento do projeto, gerenciamento de projeto, diferenças culturais. Das áreas identificadas a que possuiu menos desafios segundo a pesquisa apresentada é a de ferramentas de comunicação, conforme segue ilustração.

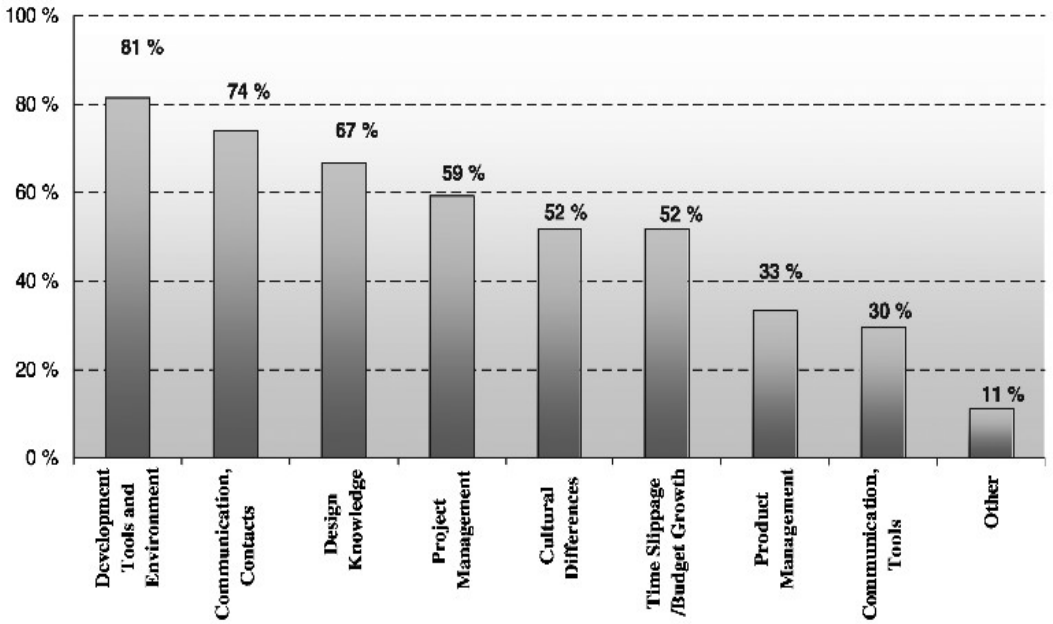


Figura 2.6 – Desafios no DDS.

Fonte: Komi-Sirvo e Tihinen, 2005.

Komi-Sirvo e Tihinen (2005), apontam com base nos resultados de suas respectivas pesquisas, algumas áreas problemáticas e fatores associados quando o desenvolvimento é distribuído: (1) ambientes e ferramentas de desenvolvimento; (2) comunicação; (3) engenharia de requisitos; (4) gerenciamento de projetos; (5) diferenças culturais e, (6) tempo e orçamento acima do estimado, entre outros. Os fatores aludidos por Carmel, Komi-Sirvo e Tihinen, entre outros fatores, podem influenciar

direta e indiretamente o desenvolvimento de um projeto distribuído. Estes são descritos a seguir, conforme segue subtópicos.

2.1.2.1. Gerência de Projetos

Cleland e Ireland (2002) citam que as principais funções da gerência de projetos são: planejamento, organização, motivação, direção e controle. É importante ressaltar que os problemas de gerência de projetos em ambientes distribuídos não diferem significativamente dos apresentados no desenvolvimento tradicional (co-localizado). Entretanto, no contexto distribuído é reforçada a importância das técnicas formais de gerenciamento de projetos; por exemplo, as apresentadas no PMBOK – Project Management Body of Knowledge[2] [Audy e Prikladnicki, 2007] [da Silva et al, 2010].

A gerência de projeto, no que diz respeito à coordenação e ao controle, se torna extremamente difícil no DDS. A integração entre as diversas partes e módulos do projeto deve acontecer de modo eficiente para que o fato de estarem longe fisicamente não interfira [Souza, 2007] [da Silva et al, 2010]. Prikladnicki (2009) afirma que o gerenciamento de projetos de DDS exige uma adaptação de algumas técnicas utilizadas em projetos co-localizados, de forma a suportar e reduzir as dificuldades impostas pela dispersão da equipe.

Em sua obra Audy e Prikladinicki (2007) asseguram que para se obter uma melhor coordenação e gerência, é necessário controlar bem as métricas envolvidas. Os autores apresentam exemplos de métricas comuns no ambiente distribuído: (1) tempo utilizando tecnologias de colaboração, como áudio e videoconferências; (2) quantidade de sessões/mensagens usando tecnologias de colaboração; (3) construção de equipe; (4) dias de viagem; (5) taxa de resolução de problemas.

2.1.2.2. Processos de Software

Evaristo et al. (2005) e Bannerman, Hossain e Jeffery (2012) sugerem que um processo de desenvolvimento é fundamental, já que, a sincronização de atividades é um dos objetivos principais e que quando as equipes distribuem o processo de desenvolvimento em diversas localidades, a falta de sincronização pode se tornar crítica.

Angioni, Sanna e Soro (2005) relatam a proximidade entre Metodologias Ágeis e DDS mostrando algumas características comuns para ambos, como *feedback* contínuo, *releases* frequentes, valorização da comunicação e padrões de codificação. Além disso, Angionini apresenta e descreve o MAAD (Methodology for Agile Distributed Development), que é um processo baseado em metodologias ágeis, criado inicialmente com foco na comunidade *open source* (código aberto). Essa metodologia foi idealizada a partir da avaliação de outros processos de desenvolvimento de software voltado para o ambiente distribuído, como é o caso do DXP (Distributed eXtreme Programming) [Kircher, 2001].

Algumas organizações utilizam metodologias convencionais para desenvolver projetos distribuídos, enquanto outras utilizam adaptações experimentais [Jimenez, Piattini e Vizcaino, 2009] [Bannerman, Hossain e Jeffery, 2012] [dos Santos et al., 2014]. Desta maneira, ainda estão testando se tais adaptações suportarão as devidas necessidades enquanto as demais organizações já utilizam adaptações sólidas, isto é, criaram seu próprio processo para lidar com o contexto distribuído [Audy e Prikladnicki, 2007] [Nordio *et al*, 2011] [dos Santos et al., 2014].

Diversos autores como Carmel (1999), da Silva *et al* (2011) e Audy e Prikladnicki (2007), mencionam a importância de se utilizar uma referência como modelo de desenvolvimento no ambiente

[2] Conjunto de práticas para Gerência de Projetos fornecido pelo Instituto de Gerenciamento de Projetos

distribuído, mas, também ressaltam a escassez de metodologias adaptadas que atendam a esse contexto e suas necessidades.

2.1.2.3. Distância sócio-cultural

Cultura é a aquisição de um conjunto de conhecimento, experiência, crenças, valores, atitudes, sentidos, hierarquias, religião, noções de tempo, funções, relações espaciais, conceitos do universo por um grupo de pessoas através das gerações [Samovar e Porter, 2004].

Quando o contexto trata de desenvolvimento global de software existem diferenças culturais, por se tratar de pessoas espalhadas ao redor do mundo que possuem diferentes hábitos, crenças, atitudes com relação à hierarquia, senso de tempo e estilos de comunicação [Nordio *et al*, 2011]. Souza (2007) fala que as diferenças culturais podem influenciar nas decisões dentro do projeto devido a determinadas tradições e costumes de cada país e que há necessidade de que estas sejam minimizadas ao máximo para que não interfiram no resultado final do produto.

Souza (2007) relaciona o fator sócio-cultural principalmente no que se refere à confiança entre as equipes envolvidas. Para que um projeto tenha um desenvolvimento eficiente, é necessário que os integrantes localizados em ambientes diferentes tenham confiança entre eles e também no projeto. Audy e Prikladnicki (2007) asseguram que muitas empresas trabalham com o conceito de *liaison*, ou seja, uma pessoa que desempenha o papel de ponte entre duas ou mais culturas, justamente por entender ou já as ter vivido.

2.1.2.4. Motivação

Equipes são formadas por pessoas, logo, são unidades sociais frágeis que por diversas razões (fraca comunicação, distância temporal e geográfica, infra-estrutura utilizada, falta de comunicação informal, diferença cultural e o tamanho do grupo, entre outros) se tornam vulneráveis. Evaristo e Scudder (2000) asseguram que a confiança é fundamental para o time cultivar o espírito de equipe, que é difícil, porém é essencial manter a confiança no desenvolvimento de um projeto distribuído. Pois times que tem poucos encontros presenciais, que os membros não possuem certo hábitos de convivência, como informalidade e estreiteza de relacionamento pode ser prejudicial no quesito de confiar no projeto e na equipe.

De acordo com Olson e Olson (2004), a motivação dos indivíduos também difere de país para país dependendo das suas culturas. Em países onde existe a valorização do individualismo, as pessoas buscam ganho material e reconhecimento pessoal. Já em outros, onde a ênfase está voltada ao coletivo, busca-se relacionamentos pessoais. Os sistemas de recompensa devem levar em conta esses valores, recompensando cada grupo com montantes ou dias de folga de acordo com seu merecimento [da Silva et al, 2011].

2.1.2.5. Distância Geográfica e Temporal

Quando a distância entre envolvidos em projetos distribuídos ultrapassa 30 metros, a frequência de comunicação diminui para nível idêntico aos que estão distribuídos à centenas/milhares de metros [Herblesb et al, 2001 apud Allen, 1977]. Evaristo e Scudder (2000) asseguram que o alto nível de dispersão proporciona maior dificuldade para monitorar o comportamento entre diferentes grupos com relação aos demais.

Audy e Prikladnicki (2007) em seu trabalho declaram que é necessário entender o nível de distância existente, pois, com isso é possível ajudar na identificação de prováveis fontes de dificuldades ou apenas caracterizar melhor a distribuição física das equipes envolvidas em projetos distribuídos. Além da dispersão física, a dispersão temporal exerce efeitos sobre o DDS. Participantes de uma

equipe estão dispersos no tempo quando há diferença nos horários de trabalho, fusos horários e/ou ritmos de trabalho que diminuam o tempo disponível para interação síncrona [Audy e Prikladnicki, 2007].

Souza (2007) declara que a distância geográfica e temporal podem ser um problema se não forem administradas da melhor forma possível. Devido à distância e à diferença de fusos horários que possam existir, a comunicação, interação e troca de informações entre as equipes de desenvolvimento devem acontecer através de ferramentas (por exemplo: baseadas em web, framework e controle de versões) e, juntamente com um suporte de procedimentos/boas práticas para que o projeto consiga atingir os objetivos previstos.

2.1.2.6. Complexidade e Tamanho do Projeto

Herbsleb *et al.* (2001) declaram que a complexidade e o tamanho do projeto também podem afetar a performance de projetos distribuídos e que a dimensão do mesmo também pode ser fator de complexidade.

Souza (2007) aponta que normalmente projetos com desenvolvimento distribuído de software são complexos e de grande porte, pois o investimento inicial neste tipo de projeto é elevado, podendo por isso, envolver diversos países. Não obstante, a utilização de metodologias tradicionais e mais completas é necessária nesse aspecto.

2.1.2.7. Comunicação

Carmel (1999), Audy e Prikladnicki (2009), Trindade et al (2008), Souza (2007), entre outros, garantem que a comunicação é fator fundamental para o desenvolvimento de projetos distribuídos. Teixeira (2007) declara que a comunicação é um dos fatores mais prejudicados no DDS, por isso é importante um canal de comunicação estruturado entre as equipes através do ambiente de desenvolvimento, mantendo todos os participantes informados sobre o andamento do projeto.

Souza (2007) assegura que a comunicação, em todos os seus sentidos, é um dos grandes problemas do desenvolvimento distribuído. É preciso que haja meios eficientes que ultrapassem as barreiras impostas pelo desenvolvimento em locais diferentes. A definição de interfaces de comunicação formal pode ser obtida por meio de modelos de processo bem definidos, com marcos (*milestones*) e métricas bem estabelecidas. Os canais de comunicação informal, por sua vez, podem abranger videoconferência, espaços de trabalho compartilhado, programas de troca de mensagens (*instant messengers*), entre outros [Teixeira, 2007] [da Silva et al, 2011].

Junior et al. (2012) retratam em seu trabalho que algumas ferramentas são fundamentais para a comunicação entre os membros, tais como: comunicadores instantâneos ou *messengers* e um *website* e, que ferramentas comuns de comunicação (e-mail, messengers, etc) precisariam evoluir para poder suportar melhor o trabalho distribuído. Apesar disso, os problemas com comunicação não são frequentemente ligados às ferramentas utilizadas. De acordo com sua pesquisa Komi-Sirvo e Tihinen (2005) constataram que apenas 30% dos projetos envolvidos relataram ter problemas com suas ferramentas de comunicação.

2.1.2.8. Ambientes e Ferramentas de Desenvolvimento

Komi-Sirvo e Tihinen (2005) declaram que embora a infra-estrutura técnica disponível atualmente pareça proporcionar o apoio adequado para o desenvolvimento distribuído, pesquisas realizadas com empresas que possuem projetos distribuídos indicaram que ferramentas e ambientes de

desenvolvimento ainda não apoiam de maneira eficiente o desenvolvimento no contexto distribuído. Isso é explicado pela falta de recursos mesmo para o contexto geral de desenvolvimento, não somente pela fase de codificação em si [da Silva et al, 2010].

Komi-Sirvo e Tihinen (2005) descrevem que a incompatibilidade de ferramentas e versões usadas por sites de desenvolvimento diferentes é muito comum e o estabelecimento de um ambiente uniforme é uma tarefa desafiadora. Além disso, as ferramentas de desenvolvimento são baseadas no pressuposto que as redes de comunicação são extremamente rápidas, o que nem sempre ocorre [Parviainen e Tihinen, 2011].

A Engenharia de Requisitos é reconhecida como a fase mais crítica na Engenharia de Software, e em ambientes distribuídos, esta apresenta uma acentuação das dificuldades existentes [Damian e Zowghi, 2002]. O processo é altamente dependente de comunicação e uma comunicação clara, que por sua vez, torna-se fundamental para evitar maus entendimentos e conflitos [Parviainen e Tihinen, 2011] [Lopes, Prikladnicki e Audy, 2004] [Bannerman, Hossain e Jeffery, 2012]. Segundo a pesquisa de Komi-Sirvo e Tihinen (2005), essa área é considerada a principal fonte de erros num projeto distribuído. Os resultados do estudo mostram que as três principais fontes de erros consistem em:

- Interpretação errada de requisitos: foi considerada a maior fonte de erro, com 74% das respostas.
- Mudança de requisitos: foi a segunda na classificação, com 70% das respostas.
- Falta de requisitos: foi a terceira maior causa de erro na pesquisa, com 59% das respostas.

As porcentagens dentro das colunas diz respeito as fontes de pesquisa. Outras fontes de erros foram citadas pelos participantes, como *Interfaces*, erros de projeto, erros de codificação, problemas de ambiente, e vulnerabilidades de segurança. A Figura 2.7 apresenta as fontes de erros em projetos distribuídos.

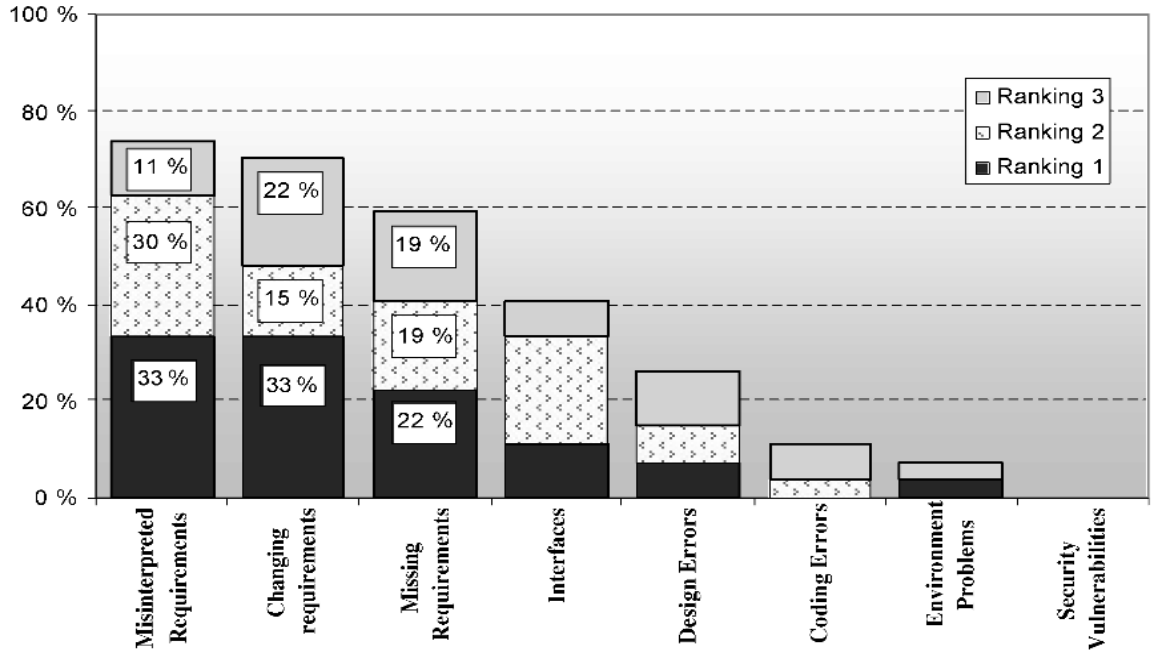


Figura 2.7 – Fontes de erros em projetos distribuídos.

Fonte: Komi-Sirvo, S; Tihinen M. (2005).

Nessa primeira seção foram abordados os conceitos sobre o modelo de desenvolvimento distribuído, bem como características e fundamentos em suas principais referências. Conforme anteriormente citado, no decorrer do capítulo todos os conceitos serão abordados, assim, a próxima seção se refere ao conceito de ontologia.

2.2 Ontologias

Diversas definições são dadas a fim de descrever o significado computacional para ontologias. A mais popular e mais conhecida é a definição dada por Gruber (1995) que a considera: “uma especificação formal e explícita de uma conceitualização compartilhada”, Nesses termos, ser formal, implica em declarativamente definida e compreensível para agentes inteligentes de software; explícita, significa que os elementos e suas restrições estão claramente definidos; conceitualização trata de um modelo abstrato, uma área de conhecimento ou de um universo limitado de discurso; compartilhada, indica um conhecimento consensual, seja uma terminologia comum da área modelada.

Assim, ontologias, em um nível de abstração mais alto, estabelecem uma terminologia comum e não-ambígua para os diversos domínios de conhecimento. Uma ontologia, segundo Guizzard (2000), é composta de conceito, relação, função, axiomas e instâncias. Em resumo, conceito pode ser “qualquer coisa” a respeito de “algo” que será explicado. O tipo de interação entre conceitos de um domínio e seus atributos é chamado de relação, cujo tipo denomina-se função.

O uso de ontologias tem se popularizado através de diversas outras subáreas da Ciência da Computação, tais como: Engenharia de Software, Inteligência Artificial, Banco de Dados e Sistemas de Informação [Freitas, 2003]. Um dos principais responsáveis por esse fenômeno é o criador da Web Semântica [Berners-Lee, 2001], este conceito foi fundamental para a disseminação das ontologias, inclusive para outras áreas conforme mencionado acima.

A principal aplicação de ontologias é a representação da Web Semântica. Entretanto, para possibilitar a comunicação entre agentes de domínios diferentes que utilizam ontologias diferentes é necessário, primeiramente, encontrar as correspondências entre conceitos destas ontologias. Este é um processo específico, chamado de mapeamento de ontologias, e apresenta muitas dificuldades tanto em determinar a correspondências como encontrá-las automaticamente [Noy, 2009]. Portanto, esse mapeamento entre formalismos de representação de conhecimento liga dois formalismos criando uma interface interoperável de acesso comum para eles, permitindo a um agente acessar o conhecimento de outro agente.

Diversas razões motivam o desenvolvimento de uma ontologia. De acordo com Noy e McGuinness (2003) e Freitas (2003) algumas dessas motivações são:

- Compartilhar entendimento comum da estrutura de informação entre pessoas ou entre agentes inteligentes de software;
- Reuso de conhecimento de um domínio. Caso exista uma ontologia que modele adequadamente certo conhecimento de um domínio, ela pode ser compartilhada e usada por engenheiros e desenvolvedores de ontologias, bem como equipes que desenvolvem aplicações semânticas e cognitivas;
- Tornar explícitas pressuposições de um domínio. As ontologias fornecem um vocabulário para representação de conhecimento e seu uso evita interpretações ambíguas.

Através da utilização desse conceito é possível realizar a tradução entre diversas linguagens e formalismos de representação do conhecimento. Essa tradução concretiza um ideal perseguido por

gerações de pesquisadores de IA, com objetivo de facilitar o reuso de conhecimento, podendo vir a permitir comunicação entre agentes em formalismos diferentes, uma vez que este serviço encontra-se disponível em um número cada vez maior de formalismos de representação do conhecimento.

2.2.1 Principais componentes de uma Ontologia

Uma ontologia pode tomar diferentes formas, mas necessariamente inclui um vocabulário de termos e alguma especificação de seus significados [Uschold e Gruninger, 1996]. O nível de formalidade de uma ontologia pode variar, porém, alguns dos componentes básicos desta não devem sofrer tal variação sendo eles: classes, propriedades, instâncias, axiomas, entre outros.

Horridge (2009) sugere que os componentes básicos independentes de uma formalização tão completa e/ou complexa de uma ontologia, se baseiam nos seguintes componentes:

- Instâncias: que representam os objetos do domínio que está sendo representado;
- Axiomas: que modelam sentenças que são sempre verdadeiras;
- Propriedades: são relações binárias entre instâncias;
- Classes: são conjuntos que contêm instâncias. Podem ser consideradas como representações concretas de conceitos do domínio.

Scatalon, Garcia e Correia (2010) argumentam que as classes devem ser organizadas em uma hierarquia. Considerando as classes, como exemplo abaixo: Programador e Gerente devem ser subclasses de Pessoa, ou seja, as instâncias do domínio que são programadores ou gerentes, também devem ser pessoas. Subclasses especializam suas superclasses [Scatalon, Garcia e Correia, 2010]. Na Figura 2.8 é possível visualizar uma representação de classes contendo instâncias.

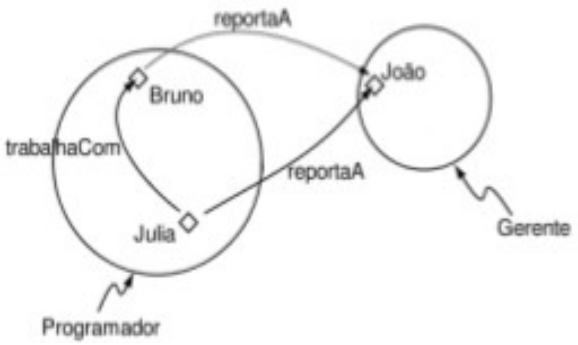


Figura 2.8 Representação de Classes

Na Figura 2.8, foi demonstrado um simples exemplo de uma ontologia com seus componentes. A classe Programador contém todos os programadores que trabalham na empresa de software do exemplo. Assim como a classe Gerente contém todos os gerentes da empresa. João, Bruno e Julia, são instâncias das classes Programador e Gerente. Entre as instâncias existem sempre os relacionamentos que foram definidos na construção das classes e de suas propriedades.

Uma classe é definida por meio de descrições formais (matemáticas) que declaram os requisitos de pertinência da classe (para uma instância). Tais requisitos delimitam a interpretação do conceito valendo-se de restrições de propriedades. Todos os tipos de restrições descrevem um conjunto que pode conter instâncias, o qual pode ser visto como uma classe anônima. Qualquer instância que for membro

dessa classe anônima satisfaz a restrição que a descreve [Horridge, 2009]. Para melhor entendimento, segue no próximo subtópico como se dá o desenvolvimento de ontologias.

2.2.2 Desenvolvimento de Ontologias

Existem algumas maneiras para se construir ontologias, desde a utilização de diretizes ou não utilização. Para que a concepção da ontologia tenha maior chance de sucesso é interessante adotar alguma metodologia. Segundo Mattos, Simões e Farias (2007), o uso de metodologias tem como objetivo combater alguns problemas que são prováveis de acontecer quando não se é aplicado a utilização de uma metodologia específica, como por exemplo: quando a conceitualização da ontologia não fica muito clara no código da implementação, outro caso, a falta de padronização acaba por dificultar seu reuso, como também, quando se existe dificuldade de compreensão acaba gerando dificuldade para a implementação de ontologias mais complexas, resultando a transferência do conhecimento para a implementação mais difícil.

Segundo Fernandez-Lópes e Gómes-Pérez (2002), cada metodologia se preocupa mais com determinadas atividades no processo de ontologias em detrimento de outras. Desta maneira, é possível escolher a mais adequada para a resolução do problema e/ou combinar as diversas metodologias, objetivando possuir os melhores resultados. Segundo Noy e McGuinness (2001), desenvolver uma ontologia inclui:

- Definir classes na ontologia;
- Organizar as classes em uma hierarquia (subclasse-superclasse);
- Definir propriedades e descrever valores permitidos a elas;
- Preencher os valores de propriedades para instâncias.

Entretanto, elaborar ontologias envolve mais do que apenas definir os conceitos da ontologia de maneira formal. É necessário que se determine o escopo do domínio, que ele seja analisado para capturar a sua conceituação, considerar o reuso de ontologias já existentes, entre outras atividades que trouxeram o surgimento da Engenharia de Ontologias [Smith e Katherine, 2008], que lida com o seu desenvolvimento prático. Há diversas propostas de processos de desenvolvimento de ontologias, como em Uschold e Gruninger (1996), Noy e McGuinness (2001) e Rautenberg, Todesco e Gauthier (2009). O último se compõe das melhores práticas de outras metodologias e está dividido em cinco grandes atividades. São elas:

1. Especificação: avaliar os custos do desenvolvimento da ontologia;
2. Conceituação: descrever um modelo conceitual do domínio de discurso;
3. Formalização: transformar o modelo conceitual em um modelo formal, passível de ser implementado;
4. Implementação: implementar a ontologia formalizada em uma linguagem de representação adequada;
5. Avaliação: a ontologia é validada quanto ao entendimento aceito sobre o domínio em fontes de conhecimento. Verifica-se a coerência do conhecimento representado na ontologia e se certifica sobre sua utilidade.

Na literatura, existem diversas linguagens com diferentes propósitos, relacionadas à *Web* Semântica, com intuito de se criar ontologias. A seguir, são descritas as principais linguagens utilizadas para representação, busca e inferência no contexto de ontologias e *Web* Semântica.

Na Figura 2.9 observam-se as linguagens de representação de ontologias para a *Web* Semântica.

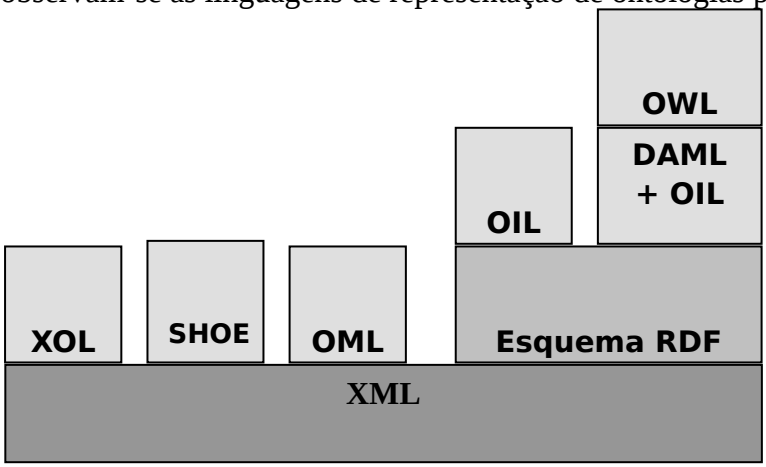


Figura 2.9: Linguagens de representação de ontologias.
Fonte: Koivunen e Miller, 2001.

XML (*eXtensible Markup Language*, Linguagem Extensível de Marcação), que é uma *meta-linguagem* de editoração, por permitir a representação de outras linguagens de forma padronizada. A linguagem RDF (*Resource Description Framework*, Arcabouço de Descrição de Recurso), por exemplo, é definida sobre ela.

XML é utilizado para prover estruturas que possam ser processadas por aplicações computacionais. Oferecendo aos usuários uma descrição de dados estruturados. Portanto, XML *Schema* é utilizado para descrever a estrutura de um documento XML. Apesar de aplicações computacionais processarem esses documentos, esta linguagem descreve a estrutura sintática apenas. Por essa razão, uma nova camada é indispensável para definir a estrutura semântica como a RDF

A RDF (*Resource Description Framework*) (MANOLA, 2004) é uma camada conceitual que junto com a XML (*Extensible Markup Language*) (BRAY et al., 2000) tornam-se cruciais para a criação de linguagens para ontologias baseadas em extensões ao RDF tais como: *SHOE* (*Simple HTML Ontology Extensions*), *OIL* (*Ontology Inference Layer*), DAML (*DARPA Agent Markup Language*), *DAML+OIL* e *OWL* (*Web Ontology Language*). Foi desenvolvida pelo W3C – *World Wide Web Consortium* e é o padrão do W3C para metadados na *Web*. Essa linguagem permite aos computadores representar e compartilhar dados semânticos na *Web* (MANOLA, 2004).

A RDF pode ser utilizada em várias áreas de aplicação da *Web*: na busca de recursos para melhorar os mecanismos de *sites* de busca já existente, em bibliotecas virtuais descrevendo o conteúdo disponível, no comércio eletrônico, principalmente na segurança e em *websites* particulares. Também é útil em outras aplicações que estão fora do escopo da Web, como recursos multimídias em geral, bibliotecas digitais e outras. A RDF em si é uma linguagem simples que é capaz de fazer relacionamentos entre informações, mas, além disso, é necessário um meio para definição de dados. A RDF *Schema* (BRICKLEY, 2000) foi criada pelo W3C com essa finalidade.

A RDF *Schema* (BRICKLEY, 2000) é responsável por prover mecanismos para declaração de relacionamentos entre recursos. Um esquema não define somente as propriedades dos recursos, mas também os tipos de recursos que estão sendo descritos. Pode ser entendido como uma espécie de dicionário onde são definidos os termos que serão utilizados em declarações RDF. A especificação da RDF *Schema* do W3C fornece os mecanismos necessários à definição de elementos, de classes de

recursos, de possíveis restrições de classes e relacionamentos e detecção de violação de restrições (LIMA, 2001).

A linguagem *OWL* (*Web Ontology Language*, ou linguagem de ontologias para a *Web*) (SMITH et al., 2003), que, na verdade, deriva de um consenso entre duas propostas, a européia OIL (*Ontology Inference Layer*, camada de inferência para ontologias, ou ainda, *Ontology Interchange Language*, linguagem de intercâmbio em ontologias) e a *DAML* (*DARPA Agent Markup Language*, linguagem de anotação para agentes do Departamento de Defesa dos Estados Unidos). Na qual *OIL* foi a primeira destas linguagens, e teve como principal requisito a facilidade de adoção por parte dos desenvolvedores, servindo principalmente à comunidade ligada à *Web* semântica (HORROCKS et al., 2000). A vantagem da linguagem *OIL* é fornecer uma linguagem de representação do conhecimento que combina lógica de descrição, sistemas baseados em *frames* e linguagens da *Web* como linguagens dais quais são baseadas sua sintaxe: *XML* e *RDF*.

A linguagem OWL (OWL, 2008) definida pelo W3C permite expressividade de alto nível e inferência explícita, sendo uma revisão da linguagem *DAML+OIL*. Entre as funcionalidades desta linguagem estão: construção de ontologias, explicitação de fatos a respeito de um determinado domínio e racionio sobre ontologias e fatos.

OWL representa conceitos e relacionamentos na forma de uma ontologia. Além disso, possui três dialetos diferentes, em ordem crescente de expressividade (SMITH et al 2003):

- OWL Lite: representa uma linguagem muito simples, portanto recomendada para a definição de hierarquias simples e/ou restrições. Ela abrange a expressividade de *frames* e lógica de descrições, com algumas restrições. Além de ser dotada de riqueza semântica, sendo, por isto, ideal para usuários iniciantes e desenvolvedores que preferem *frames* a lógica de descrições. Os axiomas e construtores para esse dialeto da OWL Lite são: *inverseOf*, *transitiveProperty*, *functionalProperty*, *inverseFunctionalProperty*, *allValuesFrom*, *someValuesFrom*, *minCardinality*, *maxCardinality* e *intersectionOf*.
- OWL-DL: é mais expressiva que a OWL Lite e é baseada na lógica de descrição, classes, portanto podem ser construídas por união, interseção e complemento, pela enumeração de instâncias e podem ter disjunções. Tipos são mantidos cuidadosamente separados (por exemplo, uma classe não pode ser instância e propriedade ao mesmo tempo). OWL-DL permite a verificação de satisfatibilidade de conceitos e classificação de hierarquias. Garante completude, decidibilidade e toda a expressividade da lógica de descrições, almejando satisfazer engenheiros de conhecimento familiarizados com esta tecnologia. Novos axiomas e construtores para esse dialeto foram adicionados em relação ao OWL Lite como: *oneOf*, *disjointWith*, *unionOf*, e *complementOf*.
- OWL FULL: representa a linguagem OWL mais expressiva. Essa linguagem é utilizada quando a expressividade do conhecimento for mais importante que garantias de computabilidade. Fornece a expressividade de OWL e a liberdade de usar RDF, inclusive permitindo novas metaclasses, já que elas são subclasses definidas em arquivos RDF.

2.2.3 Aplicação de Ontologias no DDS

Como já supracitado, o DDS é um modelo de desenvolvimento de software onde os envolvidos em um determinado projeto estão dispersos. Dessa forma, Prikladnicki (2009), Carmel e Agarwal (2001) avaliam como a distância física contribui para a complexidade neste tipo de ambiente e definem três desafios principais para o DDS: a coordenação, o controle das informações e a comunicação.

Nas últimas décadas, as ontologias vêm sendo estudadas e aplicadas na área de Ciências da Computação, como uma forma de representar e compartilhar conhecimento. A utilização de ontologias permite estruturar de forma mais clara e precisa os conceitos de um domínio e provê consistência semântica às informações [Freitas, 2003].

Ontologia se tornou um mecanismo para facilitar a produção de software em ambientes distribuídos, permitindo uma evolução na comunicação, através da agregação de conhecimento e um compartilhamento de informações de forma clara e coesa entre os envolvidos [Dillon et al., 2008].

Dillon et al. (2008) e Wongthongtham (2007) discutem a utilização de ontologias como um novo paradigma para a Engenharia de Software, através do uso de semântica como um mecanismo central de compartilhamento de informações. Em ambientes distribuídos, ontologias podem ser utilizadas, principalmente, para:

- Fornecer um mecanismo de comunicação forte e consistente através da agregação de conhecimento;
- Fornecer integração de informações e acesso a dados de fontes heterogêneas;
- Permitir desenvolver ferramentas *web* de apoio ao DDS, fornecendo semânticas definidas para Web Services;
- Criação de um repositório de conhecimento centralizado, onde todos os envolvidos podem ter acesso a mesma informação. Promovendo assim o entendimento comum entre os desenvolvedores de software, além de ser usado como modelos de domínio;
- Fornecer uma base comum de conhecimento para agentes de software que podem compor um ambiente automatizado de apoio ao processo de DDS;
- Provisão de semântica para as informações sobre times atuais e projetos passados.

Neste contexto, a utilização de ontologias pode auxiliar na resolução de alguns problemas do DDS, tais como: melhorar a comunicação, permitir uma compreensão uniforme das informações do projeto, facilitar a coordenação do projeto, evitar erros de interpretação de tarefas e sincronizar os esforços entre as equipes distribuídas. Em seu trabalho, Borges et al. (2013) garantem que existem diversas vantagens em se utilizar ontologias para esse contexto do desenvolvimento de software.

Para fazer o embasamento teórico deste trabalho sobre o conceito de ontologias, além de todas as buscas aleatórias e recebimento de material por meio de especialistas na área, foi realizado também um mapeamento sistemático, conforme apresentado na seção a seguir.

2.3 Mapeamento Sistemático da Literatura

Mapeamentos Sistemáticos [Arksey and O’Malley, 2005; Petersen and Wohlin, 2009] tentam reunir todas as pesquisas relacionadas a um conteúdo específico. Um conceito bem semelhante que existe é a revisão sistemática da literatura (RSL) [Kitchenham, 2004], que é um estudo que atua como norteador para o desenvolvimento de projetos, de forma a direcionar determinada pesquisa, sintetizando todos os estudos relevantes que respondem à dada questão. Dessa maneira, Kitchenham (2004) argumenta que entre eles existe apenas uma pequena diferença; a de que as questões de pesquisa do mapeamento sistemático são mais gerais com objetivo principal de serem mais abrangentes, no sentido de ser mais exploratório.

Linde e Willich (2003) declaram que esses métodos de pesquisa auxiliam na aglutinação de informações presentes em inúmeros estudos executados isoladamente e sobre determinada temática,

podendo exibir resultados coincidentes ou não. Travassos e Biolchini (2007) afirmaram que o conceito consiste numa forma de realizar um mapeamento ou revisão sistemática de forma não tendenciosa e abrangente, fazendo com que seus resultados tenham cunho científico. Por sua vez, Kitchenham (2004) assegura que existem alguns critérios que devem ser estabelecidos antes do início das pesquisas, para que a análise seja realmente executada de forma imparcial, onde se enquadram os seguintes pontos: critérios de busca, inclusão e exclusão de trabalhos.

Com base nisso, existem alguns aspectos que precedem o início desses estudos, os quais devem ser levados em conta, por exemplo: delimitar o objetivo da pesquisa, reconhecer a literatura e avaliar os possíveis estudos a serem incluídos. Estes três itens são essenciais ao pesquisador, pois os auxilia a aperfeiçoar a questão norteadora da revisão, tornando-a bem formulada e clara. [Domholdt, 2005]

O início do mapeamento se dá, primeiramente, pela determinação do protocolo que enumera as questões a serem pesquisadas e os métodos utilizados para guiar a revisão. Sendo assim, Travassos e Biolchini (2007) assinalam três fases essenciais para a construção do mapeamento sistemático, são elas: Planejamento, Execução e Análise dos Resultados, os quais são melhor detalhados na Figura 2.10.

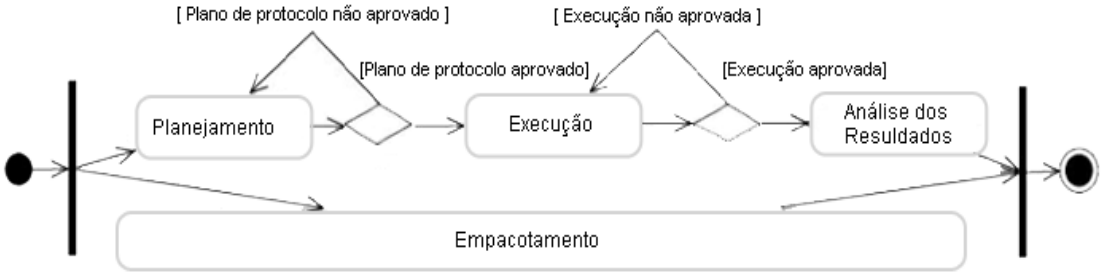


Figura 2.10 - Processo de Construção do Mapeamento Sistemático e-ou da Revisão Sitemática da Literatura.

No Planejamento estão incluídos os objetivos principais da pesquisa, as questões e os critérios que influenciaram na inclusão ou exclusão dos achados, as estratégias de investigação, a escolha prévia dos artigos obtidos, a seleção final dos mesmos e o direcionamento da revisão. É nele que se define o protocolo de revisão, onde é sugerida sua revisão por especialistas ou até mesmo através de um teste de execução.

É na fase da execução que se realiza a investigação nas fontes pré-estabelecidas prosseguindo, então, para o estudo dos trabalhos selecionados, bem como sua classificação, segundo os critérios de inclusão e exclusão determinados no protocolo. Caso haja restrições no decorrer das buscas, podem ser feitas adaptações para satisfazer quaisquer limitações.

Na última fase, a Análise de Resultados, é realizada a coleta dos dados extraídos dos artigos encontrados e selecionados segundo os critérios estabelecidos no protocolo. A partir disto, tais dados são analisados e sintetizados de acordo com o método escolhido. Ao final da execução dessa pesquisa será obtido um mapeamento da conjuntura real do conhecimento, facultando um melhor planejamento.

2.4 Raciocínio Baseado em Casos (RBC)

O conceito de RBC é um paradigma de resolução de problemas emergentes nos sistemas de tomada de decisão médica [Bichindaritz, Montani, 2011]. Pantazi, Arocha e Moerh (2004) afirmam que, ao invés de depender exclusivamente de conhecimento geral de um domínio de problema, esta abordagem utiliza o conhecimento específico de situações problemáticas concretas já vividas (chamados de casos) para enfrentar os novos.

Figura 2.11. Interação entre modelos mentais e conhecimento experimental no RBC.

Fonte: Wangenheim, 2013.

Neste contexto, o RBC é uma das tecnologias mais populares para o desenvolvimento de sistemas baseados em conhecimento, utilizando uma abordagem para solução de problemas e aprendizado por meio da reutilização de casos anteriores. Trata-se de uma tecnologia para construir sistemas computacionais inteligentes e resolver problemas reais em áreas como as do comércio eletrônico, centrais de atendimento de clientes, interpretação de dados, monitoramento e diagnóstico médico [Watson, Marir, 1994]. O RBC apresenta alguns aspectos do pensamento e comportamento humano, o que pode o fazer ser considerado um modelo cognitivo para ajudar o entendimento do funcionamento desses aspectos [Wangenheim, 2003].

Segundo Kolodner (1993), RBC é um modelo cognitivo de raciocínio e um método de construção de sistemas inteligentes, caracterizando-se por utilizar antigas experiências para resolver novos problemas procurando sempre reproduzir a capacidade de fazer analogias entre novas situações e experiências passadas.

O uso de RBC é ideal para quando não se tem um modelo, para o caso em questão, ou quando possui modelos mais simples que não resolvem os problemas enfrentados. Esse recurso faz a interação entre modelos mentais e o conhecimento experimental, conforme demonstrado na Figura 2.11, acima [Wangenheim, 2013].

Wangenheim (2003) afirma que o RBC é comumente utilizado no cotidiano pelas pessoas, onde elas resolvem seus problemas relembrando experiências passadas, comparando-as e as adaptando ao contexto do problema atual. A descrição de um problema, a solução adotada e opcionalmente a efetividade da aplicação dessa solução juntos formam um caso. O conceito de casos é muito bem descrito por Watson (1994) e Aamodt (1994), onde o colocam como sendo a representação do conhecimento de uma experiência, contendo seu conteúdo e o contexto no qual a lição pode ser aplicada.

Os casos são armazenados em uma base de dados conhecida no RBC como base de casos [Wangenheim, 2003]. Para resolver um novo problema é efetuado um cálculo de similaridade entre o novo problema e os casos contidos na base para recuperar semelhantes ao novo problema. As soluções desses casos recuperados são utilizadas para ajudar na resolução do novo problema na situação atual [Thé, 2001; Leake, 1996].

Em estudos como os de Watson (1994) e Kolodner (1993) o RBC é visto como uma metodologia para modelar o pensamento e o raciocínio humano. De maneira simplificada, um sistema que utiliza de RBC:

- Armazena experiências anteriores (casos) na memória;
- No intuito de resolver problemas, recupera experiências similares a partir de situações na memória (fazendo analogias);
- Posteriormente ao tópico acima reutiliza a experiência no contexto da situação nova (reutilização completa, parcial ou adaptada);
- Armazena a experiência nova na memória (aprendizagem).

Nesse sentido, Thé (2001) e Leake (1996) afirmam que os sistemas de RBC podem ser interpretativos ou de resolução de problemas:

- Nos interpretativos os casos anteriores são utilizados como referência para a classificação de novas situações, formando um julgamento ou a classificação de uma nova situação, comparando com casos que já foram classificados. A participação de um engenheiro do conhecimento é necessária para ajustar a situação e determinar quais características são relevantes. Assim, ele recupera os casos similares, compara estes com a nova situação, determina qual interpretação deve-se aplicar, a situação atual e a interpretação que formam um novo caso e são armazenados na base de casos, para uso futuro [Leake, 1996].
- Nos sistemas de RBC de resolução de problemas são utilizados casos anteriores para sugerir soluções para aplicar em novas situações. Seu enfoque está na geração de uma solução adaptada de um caso antigo para um novo problema [Leake, 1996; Menegazzo, 2001].

Uma situação específica de experiência é um caso do RBC, que pode ser um acontecimento ou evento, uma história ou alguma situação tipicamente com problemas, podendo ser compreendida, resolvida e recordada onde a solução foi bem sucedida [Thé, 2001]. Um caso guarda vários atributos e valores, não é uma regra, é uma experiência passada. Conforme dito por Wangenheim (2003), todos os casos são independentes e seus componentes básicos são:

- Problema/Descrição da Situação: Aponta as características e restrições da situação onde se encontra o problema;
- Solução: Representa o modo como o problema foi solucionado. As técnicas, ferramentas ou métodos utilizados estão implícitos na descrição da solução;
- Resultado/Efetividade da Solução adotada (Opcional): Identifica o resultado da aplicação da solução na situação do problema descrita, seja ela de boa efetividade ou não. Não é um componente obrigatório.

Dessa forma, para se desenvolver um sistema de RBC, inicialmente, como uma das atividades mais complexas e importantes, é necessário realizar a modelagem dos casos que, nada mais é do que identificar como representar o conhecimento. Em seguida, desenvolver um mecanismo de recuperação e identificar como buscar os casos medir sua similaridade e, opcionalmente, dependendo da solução proposta, desenvolver um mecanismo de adaptação e um mecanismo de aprendizagem.

Logo, os elementos básicos que compõem o ciclo de funcionamento de um sistema RBC são recuperação, adaptação/reutilização, revisão e retenção, conforme pode ser visto na Figura 2.12 [Wangenheim, 2003].

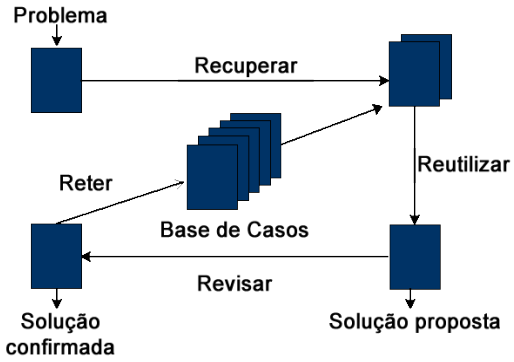
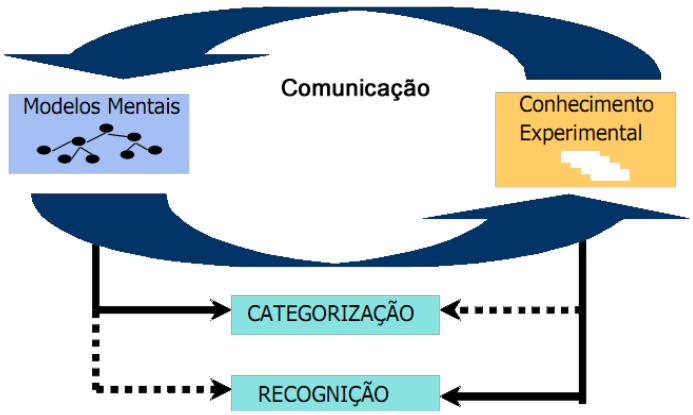


Figura 2.12 Ciclo de Desenvolvimento de um Sistema RBC. Fonte: Wangenheim, 2013.

As principais vantagens em se utilizar RBC são aquisição e manutenção de conhecimento, resolução de problemas com eficiência, qualidade nas soluções dos problemas e aprendizagem [Leake, 1996]. Em Thé (2001) elas e algumas outras são melhores explicadas:

- Aquisição do conhecimento: a aquisição pode ser feita por uma descrição direta de como a experiência da situação passada se procedeu;
- Aprendizagem: Pode aprender automaticamente na medida em que o sistema é utilizado, crescendo em robustez e incrementando sua eficiência com o tempo;
- Soluções de fácil acesso: Possibilidade de adaptação de soluções recuperadas para novos problemas, o que traz bastante eficiência pelo fato da solução não precisar ser construída totalmente;
- Ótimo poder na resolução de problemas em domínios considerados pobres. Também captura o domínio do especialista onde as regras são indefinidas, incompletas e inconsistentes;
- Criar uma base de casos é geralmente mais rápido do que criar uma base de regras, o que traz rapidez no desenvolvimento de aplicação e rápido processo de raciocínio.

Porém, ainda em Thé (2001) são listadas algumas desvantagens, nelas podemos destacar: custo de manutenção da base de dados, possibilidade de informações obsoletas na base de casos, em algumas situações requer severa indexação e avaliação de similaridade.



Vários trabalhos envolvendo a utilização de RBC em diversos tipos de aplicações podem ser encontrados. Bugmann (1999) utiliza a tecnologia no desenvolvimento de um sistema de informação voltado para o plantio de árvores frutíferas, Thé (2001) realiza a utilização de RBC unido à Lógica

Fuzzy [Weber, Klein, 2003] aplicado a diagnóstico médico e Melchiors (1999) aplica um sistema de informação construído com RBC ao gerenciamento de falhas em redes de computadores.

2.4.1 Representação dos Casos

A forma que os casos são representados possibilita melhores resultados, é uma tarefa complexa e com importância elevada [Aamondt; Plaza, 1994]. O conhecimento é representado na descrição de experiências passadas do mundo real, o caso pode ter sua experiência representada por textos, imagens e muitas outras formas de representação. Em geral, um item do mundo real é complexo, logo qualquer representação só poderá modelar estes itens incompletamente, assim, quais aspectos devem ser representados depende do problema a ser resolvido e da meta do sistema. O conhecimento pode ser interpretado de várias formas, depende muito da sua complexidade, a forma mais utilizada é a representação atributo-valor, outra forma importante é a orientação a objetos [Wangenheim, 2003].

Pode-se dizer que muitas vezes quando se representa casos em um sistema de RBC, está sendo representado o conhecimento de certa especialidade, que deve ser aproveitado em outros momento por um certo tipo de especialista, a base de casos formando assim uma base de conhecimento daquela especialidade [Kolodner, 1993].

O mais importante na forma de representação é que ela seja feita de maneira apropriada para o raciocínio que deve ser realizado sobre estes dados, não, especificamente, o modo como ela será feita. Este ponto depende muitíssimo de como a aquisição dos dados é feita, como obtê-los e que informações são relevantes para caracterizar um novo caso, evitando sempre redundâncias [Menegazzo, 2001].

2.4.2 Recuperação

A recuperação do RBC parte da hipótese de que problemas similares possuem soluções semelhantes. O sistema tem de ser capaz de encontrar um caso relevante para o problema atual na base de conhecimento, base de casos no RBC, para que esse caso relembrado se for similar, possa ajudar na resposta que traz a resolução desse novo problema [Wangenheim, 2003].

Menegazzo (2001) assegura que o objetivo principal da busca de casos similares é selecionar casos que possam ser facilmente adaptados para a situação ou problema atual, logo sua solução também será. O que ressalta a teoria de que a similaridade é a essência do RBC.

De forma resumida, o processo de recuperação funciona da seguinte maneira: é dada a descrição do problema/situação atual, depois é feito a recuperação através de uma busca na base de casos, posteriormente é realizado o cálculo de similaridade entre o caso da base e o problema/situação atual, selecionando aqueles que poderão ser aproveitados. Importa destacar que, não existe uma similaridade absoluta, sempre depende de recuperação da aplicação específica [Menegazzo, 2001].

Wangenheim (2003) apresenta que os valores das similaridades geralmente são normalizados em uma faixa de 0 a 1, onde 0 representa a dissimilaridade total (0% de similaridade) entre o caso da base e o novo problema e 1 representa a similaridade total (100% de similaridade), representando a coincidência absoluta entre o caso da base e o novo problema.

A medida de similaridade entre casos pode ser dividida entre; Global e Local. A igualdade local é no nível de métricas e atributos, sendo que, essas métricas são as que definem a semelhança entre atributos de um caso e do novo problema de acordo com determinada característica. Podendo ser calculada entre atributos de tipo numérico, textual, binário, imagens e muitos outros [Wangenheim, 2003].

A similaridade Global consiste no nível de casos, pois, nela é feita a combinação das medidas de similaridade local de acordo com o peso que cada atributo representa para o cálculo da global. A medida de similaridade é o valor numérico resultante do uso das métricas. A métrica é uma função numérica que calcula sinteticamente os valores de similaridade locais, combinadas as suas importâncias, resultando numa medida. Esta medida serve de referência para ordenar os casos mais similares [Thé, 2001]. A escolha pelo algoritmo a ser utilizado no cálculo de similaridade global depende da estrutura da solução a ser desenvolvida, segundo estudo feito por Watson (1994) os principais métodos são: Algoritmo da Vizinhança (*Nearest Neighbor* Ponderada [Cover; Hart, 1967]), Indução Guiada por Conhecimento, Algoritmo de Indução e Recuperação de Padrões.

2.4.3 Adaptação/Reutilização

Muito dificilmente casos passados recuperados são idênticos ao novo problema encontrado, logo, sistemas de RBC avançados têm mecanismos e conhecimento para adaptar os casos recuperados completamente, com o fim de satisfazer às características da nova situação [Wangenheim, 2003]. Watson e Marir (1994) justificam que à adaptação procura por diferenças entre o caso recuperado e a situação atual e aplica regras específicas para compensar essas diferenças.

Para a realização da Adaptação/Reutilização devem ser considerados primordialmente quais aspectos da situação devem ser adaptados, quais modificações devem ser realizadas, que método aplicar para realizar a adaptação e, por fim, como controlar o processo de adaptação. Para tornar possível a adaptação automática, as técnicas enfocam dois aspectos: As diferenças entre o caso atual e o caso passado e qual parte do caso recuperado pode ser transferida para o novo caso [Thé, 2001; Watson, 1997; Rafter, 2001].

Wangenheim (2003) e Watson e Marir (1994) afirmam as principais técnicas de adaptação utilizadas em sistemas RBC, algumas delas são: Adaptação Nula, Adaptação Transformacional, Adaptação Gerativa-Derivacional e Adaptação Composicional. Assim, descrevemos a seguir:

- Adaptação Nula: O sistema fornece a solução sem realizar nenhum tipo de adaptação;
- Adaptação Transformacional: Com aplicação de conhecimento específico ao domínio do problema a solução do novo caso recuperado é transformada em uma nova solução satisfazendo o problema da situação atual. A transformação pode ser com base em substituição, onde é feita uma reorganização dos elementos da solução, com base em reinstanciação, substituição baseada em regras ou em casos. Ou a transformação pode ser estrutural onde é feita adição e deleção de elementos da solução. Tipicamente emprega regras transformacionais que modificam a solução, dependendo das diferenças entre atributos do problema encontrado e do problema atual;
- Adaptação Derivacional: Neste processo não somente a solução de fato do caso recuperado é armazenada, mas também o registro do processo que levou a esta solução. Executam-se novamente todos os passos utilizados no caso anterior para calcular a resposta para o problema da situação atual;
- Adaptação Composicional: Novos componentes de solução adaptados de vários casos anteriores são combinados para produzir uma nova solução composta. A solução encontrada consiste de diferentes partes que possam ser adaptadas de forma mais ou menos independente.

2.4.4 Revisão e Retenção

A fase de retenção de casos também é conhecida como fase de aprendizagem do sistema. Dessa forma, Menegazzo (2001) assegura que desde nascimento as pessoas acumulam conhecimento que lhes

permitem agir de modo que são seres inteligentes, este fato recebe o nome de aprendizagem. Esta fase tem o objetivo similar ao acumulo de conhecimento humano. Assim, para que um sistema de RBC possa se manter atualizado e evoluindo continuamente, sempre que um novo problema for resolvido ele deve reter esse novo caso como um novo conhecimento para a base de casos [Wangenheim, 2003].

Avalia-se a solução gerada pela reutilização/adaptação, se for considerada correta, a inserção do novo caso é feita na base de casos. Caso contrário, repara-se a solução para o caso, utilizando conhecimento específico sobre o domínio da aplicação ou informações fornecidas pelo usuário. A revisão é feita com a finalidade de corrigir a solução caso necessário, para melhorar a qualidade da solução ou por algum motivo específico que o usuário deseje [Aamondt; Plaza, 1994].

A retenção nada mais é do que o processo de incorporação do novo caso relevante ao conhecimento existente na base de casos. O objetivo é melhorar continuamente a performance do sistema RBC, o tornando com o passar do tempo cada vez melhor no que tange a solucionar problemas, melhorando sua eficiência e a qualidade da solução sugerida [Watson; Marir, 1994].

Em Wangenheim (2003) são descritos três tipos de retenção de casos em sistemas RBC, estes são: Sem retenção; Retenção de Soluções de Problemas, considerada como a forma típica, nele a experiência do novo caso é incorporada a base de casos assim que o novo problema é resolvido; Retenção de Documentos, nela a aquisição do novo conhecimento é feita de forma assíncrona ao processo de solução de problemas, é feita sempre que se encontra disponível.

2.5 Processamento de Linguagem Natural (PLN)

PLN é uma sub-área da inteligência artificial e da linguística que consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em uma língua natural [Pereira, 2013]. Na computação, o objetivo de utilizar este conceito é a evolução na forma como o computador é utilizado. A maioria do conhecimento humano é registrada na forma de linguagem natural; computadores que sejam capazes de entender a linguagem natural poderão utilizar essa informação de forma rápida e para as mais variadas aplicações [Allen, 1995].

Jensen, Heidorn e Richardson (2013) citam que a linguagem natural é fácil para pessoas e difícil para máquinas e que por décadas o objetivo tem sido tentador para que computadores possam lidar com linguagens humanas para benefício das pessoas. Segundo Nadkarni, Machado e Chapman (2011) o conceito de processamento de linguagem natural ainda é promissor para as mais diversas áreas do conhecimento, principalmente para a medicina. Novas discussões são realizadas a cada ano com intenção de aprimorar esse conceito dentro do conhecimento médico.

Liddy (1998) afirma que PLN é um conjunto de técnicas que têm como objetivo analisar e representar naturalmente textos que ocorrem em um ou mais níveis de análise linguística com o propósito de alcançar um processamento de linguagem semelhante ao humano em diferentes tipos de atividades e aplicações.

Hoje em dia, a maior parte das abordagens existentes se baseiam ainda na representação sintática de texto, que é um método baseado principalmente nas frequências de ocorrência de palavras.

Tais algoritmos são limitados pelo fato de que eles podem processar apenas uma informação que pode ver. No caso, os processadores de texto humanos, essas limitações não existem, onde cada palavra vista pelo humano ativa uma cascata de conceitos relacionados semanticamente, episódios relevantes e experiências sensoriais, todos com intuito de concluir de as atividades de PNL complexas de uma forma rápida e sem esforço [Cambria e White, 2014].

As técnicas de PLN são classificadas de acordo com o nível de unidade linguística processada [Liddy, 1998], são eles:

- Nível Fonológico: Nível em que são interpretados os sons da fala através de fonemas;
- Nível Morfológico: Analisa as formas variantes de palavras por meio de seus prefixos, radicais e sufixos;
- Nível Lexical: Faz a análise da parte do discurso e significado léxico das palavras;
- Nível Sintático: Analisa as palavras em uma sentença com a finalidade de descobrir a estrutura gramatical das sentenças;
- Nível Semântico: Busca identificar o possível significado das sentenças, incluindo a retirada de palavras ambíguas em contexto;
- Nível de Discurso: Interpreta a estrutura e o significado transmitido por textos maiores que uma sentença;
- Nível Pragmático: Compreende o propósito do uso de certos tipos de linguagem em situações específicas, em particular aqueles aspectos da linguagem os quais requerem conhecimento de mundo.

A representação do significado de uma sentença é obtida através da sua forma lógica [Allen, 1995; Franconi, 2001]. A forma lógica codifica os possíveis sentidos de cada palavra e identifica os relacionamentos semânticos entre palavras e frases [Gonzales, 2002]. Na Figura 2.13 são demonstradas as transformações que compõem o processamento da sentença até sua forma lógica.

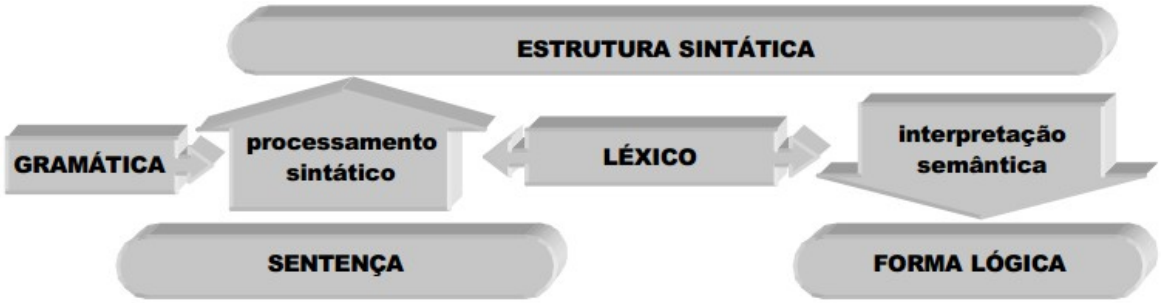


Figura 2.13 - Transformações de Sentença até a Forma Lógica.

Fonte: Gonzalez (2002).

As representações da estrutura sintática de uma sentença e de sua forma lógica apresentados na Figura 2.13, são definidas por leis gramaticais e se concretizam graças ao conhecimento do significado dos itens léxicos, viabilizando o processamento sintático e a interpretação semântica (Gonzales, 2002).

A estrutura sintática é obtida após o processamento sintático (também conhecido como análise sintática e/ou *parsing*), que tem suas regras definidas pela gramática adotada para a linguagem da sentença.Para esta etapa sintática são necessárias as categorias gramaticais das palavras que são encontradas em uma etapa anterior chamada de processamento léxico (também conhecido como analise léxica), onde é feito um processo de *scanning* (palavra em que o processo da analise léxica é conhecido) e as categorias das palavras são encontradas. Estas categorias podem ser definidas através de muitas formas nas linguagem, onde o uso de expressões regulares como formalismo é uma boa

opção (Gonzalez, 2002). Estas categorias das palavras encontradas na análise léxica são utilizadas durante todo o processo de transformações.

A estrutura sintática tem suas características analisadas pela interpretação semântica (também conhecido como análise semântica) e assim é encontrada sua forma lógica, em alguns casos, a realização da interpretação semântica em conjunto com o processamento sintático é uma boa opção. Com a forma lógica das sentenças reconhecidas, a próxima etapa é a interpretação contextual, onde a forma lógica é traduzida para a representação de conhecimento final escolhida.

Em estudo feito por Vieira (2001) são destacadas diversas aplicações decorrentes do estudo e desenvolvimento da área de linguística computacional. Algumas delas podem ser atreladas a atividade de PLN, alguns exemplos: reconhecedores e sintetizadores de fala, corretores ortográficos e gramaticais, tradutores automáticos, geradores de texto e resumo, extração de informação e interfaces de linguagem natural para domínios específicos.

Gonzalez e Lima (2003), afirmam que a gramática e o léxico são recursos indispensáveis a transformação da sentença em sua forma lógica. Dessa forma, a seguir os dois temas são abordados de forma que suas seus principais conceitos e características sejam apresentados.

2.5.1 Gramática

Uma gramática é uma maneira de especificar de forma finita linguagens eventualmente infinitas. A gramática define um conjunto finito de regras que quando aplicadas sucessivamente geram sentenças (também conhecido como palavra na área das linguagens formais), onde o conjunto de todas as sentenças geradas por uma gramática define a linguagem [Menezes, 2000].

Muitos tipos de linguagens podem ser formalizadas por gramáticas. Para isso diversos tipos de gramáticas com diferentes poderes de geração existem, as formas básicas das mais famosas são: Gramática Regular (gera linguagens regulares), Gramática Livre de Contexto (gera linguagens livres de contexto), Gramática Sensível ao Contexto (gera linguagens sensíveis ao contexto) e Gramática Irrestrita (gera linguagens enumeráveis recursivamente) [Menezes, 2000].

Nenhum tipo de gramática é tido como o melhor, depende muito da finalidade de sua utilização. Porém os modelos que se situam entre as gramáticas livres de contexto e aquelas sensíveis ao contexto têm sido propostos pelos pesquisadores como os mais indicados para PLN [Gonzalez, 2002; Vieira, 2001].

Na Figura 2.14 é demonstrado um exemplo de gramática de um fragmento da língua portuguesa, onde “o”, “gato”, “rato” e “caçou” são os símbolos terminais e “frase” é o não-terminal inicial. A regra de produção “frase -> sujeito predicado” estabelece que uma “frase“ é composta de um sujeito seguido de um predicado, a regra de produção “substantivo -> gato | rato” estabelece que um substantivo pode ser “gato” ou “rato”.

$$
\begin{aligned}
&frase \Rightarrow sujeito\ predicado\\
&sujeito \Rightarrow artigo\ substantivo\\
&predicado \Rightarrow verbo\ artigo\ substantivo\\
&artigo \Rightarrow \texttt{o}\\
&substantivo \Rightarrow \texttt{gato} \mid \texttt{rato}\\
&verbo \Rightarrow \texttt{caçou}
\end{aligned}
$$

Figura 2.14 - Exemplo de Gramática.

Fonte: Pereira, 2013.

Neste caso, na Figura 2.15 é demonstrado a árvore de derivação (também conhecida como árvore gramatical ou árvore sintática) adquirida após o processo de análise sintática ou *parsing*, processo que utiliza a gramática, também conhecido como processo de derivação, da frase “o gato caçou o rato”, utilizando o método top-down [Aho; Sethi; Ullman, 2008].

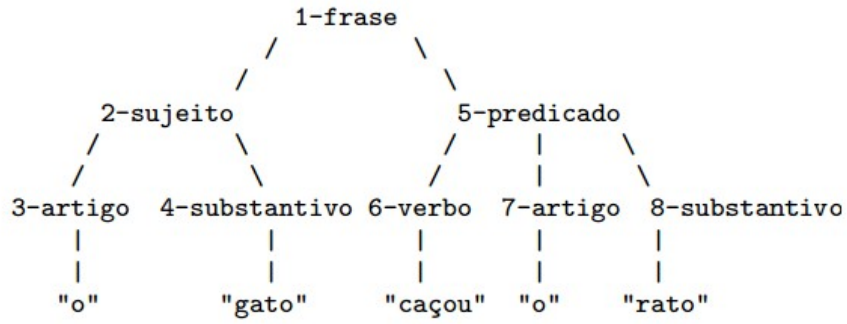


Figura 2.15 - Árvore de Derivação Gramatical.
Fonte: Pereira, 2013

2.5.2 Léxico

Scapini (1995) define o termo Léxico, como uma relação de palavras com suas categorias gramaticais e seus significados. No caso de um idioma, um léxico é o universo de todos os seus itens lexicais, que seus falantes utilizam, já utilizaram e podem vir a utilizar.

O termo dicionário, também chamado de dicionário, carrega tipicamente impresso o significado de vocabulário para leitores humanos. Em alguns casos, utiliza-se o termpo "léxico" para identificar o componente de um sistema de PLN com informações semânticas e gramaticais sobre itens lexicais.

De qualquer forma, o propósito dos dicionários é prover uma grande gama de informações sobre palavras, como etimologia, pronúncia, morfologia, sintaxe, entre outras. Ou seja, estes fornecem definições de seus sentidos [Guthrie, 1996].

No contexto do PLN surgem ainda os léxicos com capacidade de serem legíveis e tratáveis por máquina [Wilks1996]. Espera-se que informações lexicais em larga escala possam ser extraídas automaticamente através do que tem sido denominado de “dicionário legível por máquina” (machine-readable dictionarie — MRD), melhorando, assim, a uniformidade e a consistência da informação. A capacidade das máquinas de tratar dicionários, entretanto, vai além dos MRDs, com o surgimento dos “dicionários tratáveis por máquina” (machine-tractable dictionaries — MTDs). Os MTDs, contendo um grande conjunto de informações lingüísticas, viabilizam a conversão de um dicionário existente em uma forma apropriada para PLN.

Gonzalez e Lima (2003) citam ainda que, a estrutura sintática de uma sentença é obtida através do processamento sintático, sendo a representação desta estrutura regida por leis gramaticais, que são definidas na gramática. Outras informações necessárias a esta etapa, como as categorias morfológicas das palavras, são encontradas em um léxico. O mapeamento da estrutura sintática da sentença em sua forma lógica é realizado pelo processamento semântico e, nele, o léxico também exerce papel fundamental, com informações sobre o significado dos seus itens lexicais.

Cambria e White (2014) afirmam que com a grande quantidade de informação existente, principalmente ocasionadas pelo crescimento da internet, a tendência é aumentar ainda mais e exigir que as máquinas façam parte do trabalho do humano. Contudo, esta grande quantidade de informação,

é principalmente não estruturada (porque é produzido especificamente para consumo humano) e, portanto, não diretamente processáveis por máquina. A análise automática de texto envolve uma profunda compreensão da linguagem natural por máquinas, uma realidade da qual ainda estamos muito longe.

2.6 Resumo do Capítulo

O objetivo deste capítulo foi realizar uma descrição geral acerca estado da arte descrevendo e conceitualizando o Desenvolvimento Distribuído de Software, através da apresentação de ilustrações, as quais permitem o entendimento deste conceito e a visualização dos tipos de distância que existem em ambientes distribuídos. Além disso, ressalta as principais motivações e incentivos que fazem com que o DDS seja adotado por empresa, como também, apresenta a contextualização do que os estudiosos na área afirmam ser os principais desafios desta abordagem, como por exemplo: gerência de projetos, processo de software, distância sócio-cultural, motivação, distância geográfica e temporal, entre outros.

Também foi apresentado neste capítulo outros conceitos importantes inerentes a esse trabalho, tais como: Ontologias, suas definições e características, vantagens de utilização, componentes essenciais que uma ontologia deve possuir e os procedimentos necessários para contruir uma. Ainda assim, foram consideradas a aplicação de ontologias no contexto de DDS, caracterizando o cenário geral desse trabalho.

Outro conceito fundamental apresentado neste texto foi o de Raciocínio Baseado em Casos, inicialmente discutidos através de seus conceitos e aplicações, em seguida com seus passos fundamentais para o funcionamento, sendo eles: a Representação dos Casos, a forma de Recuperação, a Adaptação/Reutilização dos casos e por fim, a Revisão e Retenção. Para finalizar o capítulo, foi apresentado também, de forma bem específica, a visão da técnica de Processamento Natural de Linguagem, seu funcionamento, sua aplicação, sua gramática, seu léxico e exemplo de uso.

3. Métodos de Pesquisa

O primeiro passo realizado deste trabalho foi o mapeamento sistemático, com intuito de fazer o levantamento do estado da arte sobre ontologias em DDS, este foi um passo fundamental para entendimento e percepção de problemas no contexto. O segundo passo, referente à definição e construção da ontologia, começou a ser realizado em 2011 na disciplina de web semântica da pós-graduação do Centro de Informática, sendo mais uma etapa essencial para construção da presente pesquisa, mesmo por que foi critério decisivo para definição do problema de pesquisa. Em contrapartida e quase que em conjunto com essa primeira atividade existiu outro trabalho que foi a a implementação da DKDs, fundamental para testar e verificar se a ontologia em desenvolvimento estava de acordo com o esperado, essa atividade foi realizada com testes individuais sob a ferramenta em desenvolvimento.

O quarto passo foi o povoamento da ontologia, em seguida a implementação da ferramenta de RBC, contemplando a abordagem que envolve a ontologia e o conceito de RBC. Por fim, duas outras atividades foram realizadas, são elas: a construção da base de casos e a validação da proposta como todo. Esses passos foram necessários para que essa tese pudesse ser desenvolvida e avaliada.

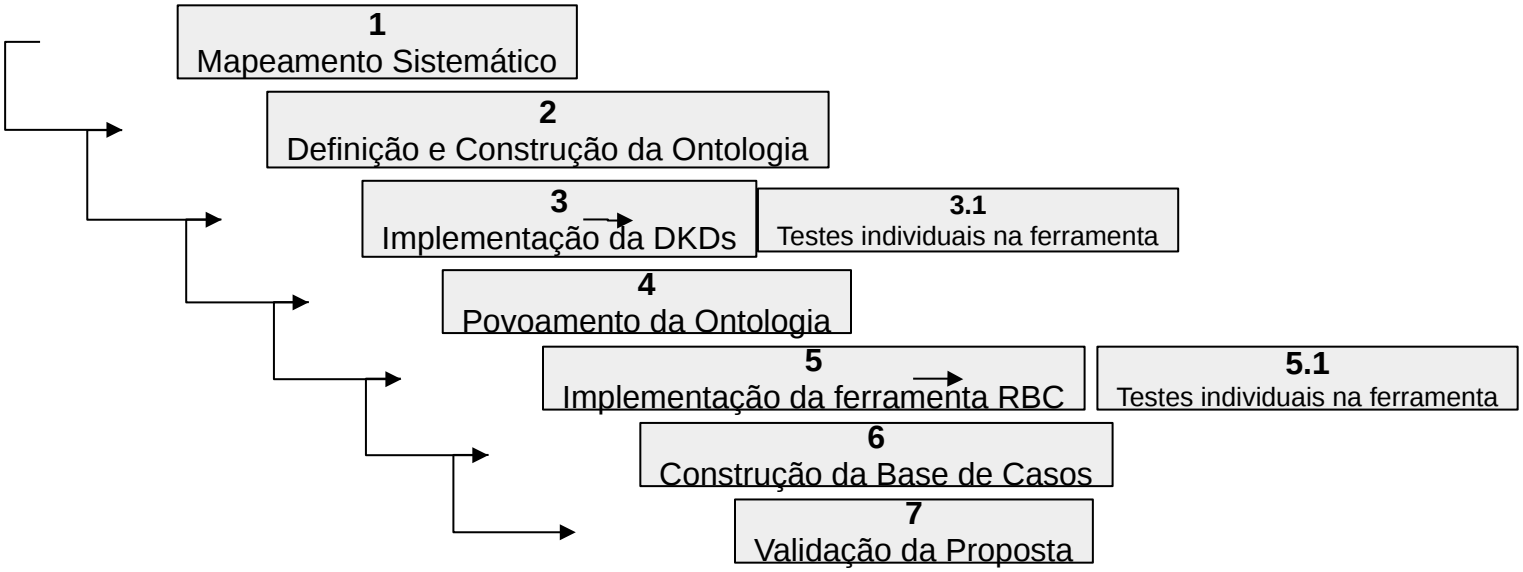


Figura 3.1 – Passos Utilizados para Realização deste Trabalho.

Os passos utilizados para este trabalho estão subdivididos em seções. A primeira traz o mapeamento sistemático da literatura pesquisada para desenvolvimento do tema, como mencionado, com o objetivo principal de levantar evidências a respeito da utilização de ontologias em projetos distribuídos. A segunda, por conseguinte, apresenta todos os passos para a extração de informação de artigos publicados em conferências, com intuito de se construir casos e de se levantar as informações mínimas necessárias para o povoamento inicial da ontologia.

A terceira e última seção, diz respeito a pesquisa realizada com profissionais da indústria, objetivando aproximar mais os casos utilizados com a realidade de projetos reais. Os outros três métodos de pesquisa – Construção da Ontologia, Implementação da DKDs e Implementação da ferramaneta RBC – são descritos no capítulo seguinte, por questões de organização do texto.

3.1 Mapeamento Sistemático da Literatura

No estudo feito sobre as diversas literaturas existentes sobre o tema, um mapeamento sistemático foi conduzido com o objetivo de identificar ferramentas, técnicas, boas práticas e modelos já existentes que utilizam ontologias para apoiar o Desenvolvimento Distribuído de Software. O protocolo utilizado como guia de execução do mapeamento foi baseado nos guias definidos por Kitchenham (2007) e por Travassos e Biolchini (2007). O protocolo completo pode ser visto no Apêndice E.

3.1.1 Questões de Pesquisa

As questões de pesquisa foram definidas a partir dos objetivos do mapeamento sistemático. Um fator importante neste processo foi a busca por opiniões e análises de especialistas da área sobre dúvidas ainda existentes e pontos a serem investigados a respeito de como os recursos ontológicos podem beneficiar o processo de Desenvolvimento Distribuído de Software.

Outro ponto bem definido nas questões foi a análise das demandas e pesquisas existentes neste contexto do DDS. Assim, pode-se definir as seguintes questões de pesquisa propostas:

1 Quais das ferramentas, técnicas, modelos e/ou boas práticas que utilizam ontologias têm sido propostas e/ou adotadas no apoio ao DDS?

2 Quais as ontologias propostas e/ou formalizadas no contexto de DDS?

3.1.2 Estratégia de Busca

Esta seção se dedica à descrição da estratégia de busca dos estudos primários utilizados para desenvolvimento do tema. Desse modo, o primeiro passo foi a construção da *string* de busca, com a identificação dos principais termos da pesquisa e seus sinônimos que se deu a partir das questões de pesquisa anteriormente definidas. Estas definições foram realizadas com base em outras revisões e mapeamentos sistemáticos que envolviam os termos de DDS e Ontologias, além da orientação de especialistas e pesquisadores da área.

A estratégia de definição da *string* de busca ainda envolveu uma etapa de testes, visando a refinar e obter a *string* que mais se adequava aos objetivos da pesquisa. Esses testes aconteceram através da utilização de diferentes versões da *string* na realização de buscas automatizadas em alguns engenhos de bibliotecas digitais selecionadas, tais como, o *IEEE Digital Library* e o *Elsevier Scopus*. Nessa etapa de testes, para ajustar a *string* de busca, foi verificado 10% dos artigos como amostra a fim de identificar quais *strings* obtinham melhores resultados.

Com isso, a primeira etapa das buscas de teste utilizou uma *string* mais completa, composta por diversos termos relacionados com ontologias e conhecimento como "*knowledge representation*", "*knowledge management*", "*knowledge transfer*", "*conceptualization*" e "*concepts formalization*". Essas buscas retornaram muitos trabalhos que apresentavam o termo *knowledge* que porém, não possuía relevância para a pesquisa, pois os artigos encontrados não abordavam o tema ontologia e por isso, todos os termos relacionados com *Ontologies* foram retirados, com exceção de *Ontology* e *Ontologies*.

Assim, com a *string* reduzida, a quantidade de trabalhos retornados também diminuiu, mas apesar disso, todos aqueles considerados relevantes para a pesquisa, encontrados nas primeiras buscas, continuaram sendo alcançados. Dessa forma, os testes com a versão reduzida da *string* se mostrou mais eficiente para a pesquisa e o resultado dessa *string* de busca obtida nesse processo está exposto na Tabela 3.1. Diante disso, foi possível observar que alguns dos artigos recuperados não foram relacionados com ontologias e desse modo, todos os termos de pesquisa relacionados com ontologias e conhecimento foram removidos, com exceção dos termos ontologia e ontologias.

Tabela 3.1. Termos de Busca

Referente	Termos
Ontologias	(Ontology or Ontologies)
DDS	AND (Distributed Software Development OR Global Software Development OR Collaborative Software Development OR Global Software Engineering OR Globally Distributed Work OR Collaborative Software Engineering OR Distributed Development OR Distributed Team OR Distributed Teams OR Global Software Team OR Global Software Teams OR Globally Distributed Development OR Geographically Distributed Software Development OR Offshore Software Development OR Offshoring OR Offshore Outsourcing OR Dispersed Team OR Dispersed Teams OR Distributed Software Project OR Multi-site Software Development OR Distributed Environment of Software OR Outsourced Software Project OR Virtual Team OR Virtual Teams)
Especifidade	AND (Technique OR Techniques OR Method OR Methods OR Tool OR Tools OR Software OR Program OR Programs OR System OR Systems OR Model OR

	Models OR Framework OR Frameworks OR Methodology OR Methodologies OR Good Practice OR Good Practices OR Best Practice OR Best Practices OR Lesson OR Lessons OR Learned OR Success Factor OR Success Factors)

A estratégia de busca utilizada para mapear os estudos primários envolve buscas automatizadas através de conhecidos engenhos de buscas para as bibliotecas digitais. Estas bases foram escolhidas baseadas na relevância para a comunidade de computação e disponibilidade em baixar os artigos na internet os quais tivessem parceria com a Universidade Federal de Pernambuco. A pesquisa foi executada nas seguintes bibliotecas digitais:

- ACM Digital Library (http://dl.acm.org);
- EI Compendex (http://www.engineeringvillage2.org);
- IEEE Digital Library (http://ieeexplore.ieee.org);
- Science Direct (http://www.sciencedirect.com);
- Scopus (http://www.scopus.com/home.url).

O processo de pesquisa ainda envolveu buscas manuais em anais de eventos da área da pesquisa. Nesta etapa, se considerou apenas algumas das principais conferências relacionadas ao assunto, julgadas como mais relevantes e de mais fácil acesso pelos pesquisadores envolvidos, sendo elas: a International Conference on Global Software Engineering (ICGSE) e Workshop de Desenvolvimento Distribuído de Software (WDDS).

Além da pesquisa em anais de eventos, de forma para acrescentar, a busca manual envolveu consultas a especialistas da área de DDS, que faziam parte da academia mas também prestam consultoria à indústria, que por sua vez, sugeriram alguns artigos que segundo o seu entendimento eram considerados importantes. Por fim, não foi utilizada nenhuma limitação no processo de busca em relação ao período inicial de publicação o qual tem como limite final de publicação dos artigos o mês de dezembro de 2012.

Em relação às buscas manuais, as conferências selecionadas foram pesquisadas a partir do seu primeiro ano de realização até a edição de 2013. Estudos publicados a partir de 2014 não foram considerados na pesquisa, de modo a produzir um resultado mais homogêneo e também a permitir uma futura atualização deste mapeamento que poderá considerar as publicações a partir desta data.

3.1.3 Estratégia para Seleção de Estudos Primários

Esta seção descreve os critérios de inclusão da pesquisa e o processo de seleção dos estudos primários o qual possui uma equipe composta por três pesquisadores que reunida para a realização deste processo, tem como integrantes: dois alunos de doutorado e um aluno de mestrado, além de dois professores os quais supervisionam todo o processo.

A inclusão de um trabalho é feita de acordo com sua relevância em relação às questões de pesquisa, nesse caso, os critérios de inclusão são: (1) o estudo deve apresentar técnicas, ferramentas, modelos e/ou boas práticas que utilizem ontologias no suporte ao DDS ou ontologias formalizadas neste contexto; (2) o estudo deve estar no formato de artigo completo, ou seja, resumos expandidos e apresentações não serão considerados; (3) o estudo não pode ser duplicado e neste caso, será desconsiderado o mais antigo e/ou menos completo; e ainda, (4) o estudo não pode ser repetido, sendo a primeira fonte de pesquisa considerada.

É importante ressaltar que estes critérios foram discutidos com todos os pesquisadores envolvidos no trabalho e assim, o processo de seleção de estudos primários foi realizado em três etapas, conforme apresentado a seguir:

- Seleção dos estudos primários potencialmente relevantes para a pesquisa;
- Seleção dos estudos primários através dos critérios de inclusão/exclusão;
- Resolução de conflitos e avaliação final dos estudos primários selecionados.

Na etapa inicial, os estudos primários encontrados após o processo de busca foram avaliados por um pesquisador, no caso o autor deste trabalho, levando-se em consideração apenas o título, o resumo e palavras-chaves. Somente os artigos claramente irrelevantes para a pesquisa foram excluídos e mesmo assim, desde que não restassem dúvidas sobre a sua irrelevância com o tema. Ao final desta etapa, todos os trabalhos potencialmente relevantes para a pesquisa foram selecionados para uma posterior análise.

Na segunda etapa, todos os artigos aprovados na seleção anterior foram avaliados por dois pesquisadores que se alternavam em duplas para ler os artigos, levando em consideração do mesmo modo o título, o resumo, as palavras-chave, e ainda, a introdução e a conclusão. Em alguns casos, foi necessária inclusive uma leitura mais aprofundada do artigo.

Cada pesquisador foi responsável por ler e avaliar individualmente os trabalhos, selecionando ou não cada artigo através dos critérios de inclusão adotados nesta pesquisa. Dessa maneira, caso algum artigo não se enquadrasse em pelo menos um critério de seleção, seria excluído do mapeamento. Ao final desta etapa, cada pesquisador elaborou um relatório com as decisões de inclusão ou de não inclusão de cada estudo primário na pesquisa, com as respectivas justificativas.

Na etapa final da seleção, os dois relatórios – um de cada pesquisador – com as decisões de inclusão ou exclusão de cada estudo primário foram confrontados. Nos casos de discordância de decisões, o terceiro pesquisador, responsável por fazer uma leitura mais aprofundada, opinava e pela inclusão ou não do trabalho. Ao término, dois pesquisadores fazem uma reunião para discutir e decidir de vez pela inclusão ou exclusão dos artigos que estavam em desacordo, chegando finalmente ao resultado deste processo que é um conjunto final de estudos primários já incluídos no mapeamento.

No total participaram desse mapeamento, 2 alunos de doutorado (incluindo o autor dessa tese), um aluno de mestrado e dois professores do Centro de Informática. Na segunda etapa do mapeamento, que foi a atualização para o ano de 2013, participaram do processo de inclusão e análise mais 2 alunos de graduação da UFRPE.

3.1.4 Extração e Síntese de Dados

Após a seleção dos estudos primários, é realizada a extração e análise dos dados relevantes para a pesquisa com adoção da ferramenta JabRef [Jabref, 2010] para organizar essa extração de dados, a análise e as sínteses dessa pesquisa, habilitando assim o modo de customização nas referências e auxiliando o processo de leitura, organização e revisão de artigos durante toda a análise.

Todos os estudos primários receberam um identificador único para indicá-los facilmente durante o processo. Por exemplo, PS01 (do inglês, primary study) significa o primeiro estudo primário selecionado. Para cada estudo primário, foram extraídas e registradas as seguintes informações:

1. **Título**;
2. **Autor (es)**;

3. **Local de publicado**: Conferência ou periódico onde foi publicado;
4. **Ano de publicação**;
5. **Tipo de estudo**: acadêmico ou industrial;
6. **Foco da pesquisa**: pessoas, organizações ou projetos;
7. **Método de pesquisa**: experimentação, *survey*, estudo de caso, estudo teórico ou relatório de experiência;
8. **Tipo de contribuição**: pesquisa validada, pesquisa em evolução, proposta de solução, trabalho teórico, trabalho de opinião ou trabalho de experiência;
9. **Respostas às questões de pesquisa**: evidências para responder as perguntas de pesquisa investigadas por este mapeamento;
10. **Informações adicionais**: evidências para permitir análises secundárias no mapeamento: (1) quais as fases de desenvolvimento que os recursos apoiam; (2) quais os problemas ou finalidades que levaram a proposta e/ou adoção de ontologias como formalismo de representação de dados no DDS; e, por fim, (3) quais as vantagens e desvantagens de utilizar ontologias no DDS.

O ítem 5, tipo de estudo, é apenas para definir se o estudo foi realizado dentro de uma universidade ou na indústria. Foi levado em consideração que se o estudo foi desenvolvido na universidade para ser utilizado na indústria, que este estudo ainda sim era acadêmico, pois o mesmo tem seu âmbito essas instituições de ensino e pesquisa, logo, seu cenário é completamente específico.

Nesta etapa também foi realizada a avaliação da qualidade de cada estudo primário, com o objetivo de ter uma visão geral da qualidade dos estudos selecionados e direcionar esforços para os artigos mais importantes durante o processo de análise dos resultados. Assim, os critérios de avaliação de qualidade utilizados foram: (1) Os objetivos ou questões do estudo são claramente definidos, (2) Existe uma clara descrição do contexto na qual a pesquisa foi realizada, (3) O estudo é coerente e bem estruturado, permitindo avaliação, (4) Os resultados encontrados estão relatados de forma clara, (5) A pesquisa apresenta comparação de seus resultados, (6) O estudo apresenta técnicas, ferramentas, modelos ou boas práticas que utilizam ontologias no apoio ao DDS e (7) O estudo apresenta ontologias formalizadas no contexto de DDS. Os formulários utilizados para extração de informações podem ser visto no Apêndice B.

Nesse processo, cada estudo primário recebeu pontuações de acordo com os critérios que alcançava, desta forma: 0 (estudo não atendeu ao critério), 0.5 (estudo atende parcialmente o critério) ou 1 (estudo atende o critério completamente) ponto para cada critério definido. Dessa maneira, os estudos primários podem ser classificados como: Ruim (0, 0.5 ou 1.5 ponto), Regular (2 ou 3 pontos), Bom (3.5 ou 4.5 pontos), Muito bom (5 ou 6 pontos) ou Excelente (6.5 ou 7 pontos). O formulário de coleta de dados e de avaliação podem ser vistos no Apêndice F.

3.1.5. Resultados do Mapeamento Sistemático

Essa seção apresenta os resultados do mapeamento sistemático, sendo analisados na subseção seguinte os resultados do processo de busca, nessa seção são apresentados e discutidos as informações gerais sobre os estudos primários, como ano de publicação, método de pesquisa e país de origem de cada estudo, e por fim, as respostas para cada questão de pesquisa são apresentadas na seção seguinte.

As buscas para a obtenção dos estudos primários foram conduzidas de acordo com os planos de pesquisa definidos no protocolo. O processo de busca conseguiu recuperar 2.138 estudos a partir das

bases de dados científicos escolhidas, e assim, depois de realizar o processo de seleção de estudos primários, foram selecionados 51 artigos relevantes.

Na Tabela 4.1 são resumidos os resultados do processo de seleção dos estudos primários, sendo a primeira coluna a apresentação das bibliotecas digitais utilizados neste estudo e as conferências onde foram realizadas pesquisas manuais e a segunda coluna o número de documentos recuperados no processo de busca. Na terceira coluna são exibidos os números de trabalhos selecionados após a primeira etapa do processo de seleção que consistiu em avaliar título, resumo e palavras-chave para excluir estudos claramente irrelevantes para esta pesquisa. A quarta, quinta e sexta colunas apresentam o número de artigos excluídos após o segundo estágio (processo de seleção), e finalmente, nas duas últimas colunas são exibidos, respectivamente, o número de artigos incluídos no mapeamento e a percentagem de inclusão por fonte de pesquisa .

Tabela 3.2. Resultado do Processo de Seleção dos Estudos Primários

Source	Search Results	Relevant Studies	Irrelevant	Repeated / Duplicated	Incomplete	Primary Studies	% Inc.
ACM	392	57	30	13	10	4	7.8%
EI Compendex	53	21	5	16	0	0	0
IEEE	680	164	106	15	16	27	52.9%
Science Direct	307	55	34	10	6	5	9.8%
Scopus	547	99	44	30	13	12	23.5%
Manual	157	15	11	3	0	1	1.9%
Experts	2	2	0	0	0	2	3,9%
Total	**2138**	**413**	**230**	**87**	**45**	**51**	**100%**

Analisando a Tabela 3.2, o EI Compendex parece ser a biblioteca digital menos eficiente, com o menor número de artigos retornados, no entanto, fornece uma lista mais precisa de artigos quando comparado a outras fontes de pesquisa.

Por ser a última pesquisa realizada, os 16 artigos encontrados na base EI Compendex que foram considerados relevantes para o estudo foram excluídos, por estarem duplicados, ou seja, foram devolvidos e contabilizados em outros motores de busca na pesquisa anteriormente feita. Isso foi ocasionado pela ordem de realização das buscas, no caso, o EI Compendex foi o útlimo.

Outras fontes que também obtiveram baixo desempenho foram a ACM e Science Direct, exigindo ainda mais trabalho na seleção de estudos, que ainda assim, obtiveram número baixo de artigos seleccionados a partir deles.

É interessante notar ainda, que o IEEE e Scopus são as bibliotecas que possuem o maior número de trabalhos publicados relacionados ao tema em estudo, representando quase 76,4% (39 artigos) do total de artigos analisados, com estudos primários incluídos combinado. Além dos resultados das buscas automáticas (94,1% - 48 artigos), pode-se observar que alguns estudos, mais precisamente, 5,9% deles (3 artigos), incluídos na pesquisa, foram encontrados a partir de pesquisas manuais ou através de indicações de especialistas em DDS.

Pela análise da Tabela 4.12 é também possível observar um pequeno número de estudos primários devolvidos pelos engenhos de busca das bibliotecas digitais e por pesquisas manuais, quando comparado com outros mapeamentos e revisões sistemáticas na área de Engenharia de Software, por exemplo, as pesquisas de Saraiva et al. (2012), de Almeida et al. (2011) e Kitchenham 'et al. (2011), que retornaram em suas pesquisas, respectivamente, 5,620 (cinco mil seiscentos e vinte), 7,101 (sete mil cento e um) e 2,506 (dois mil e quinhentos e seis) estudos primários.

Isso acontece, principalmente, devido ao fato de que o tema desta pesquisa é relativamente recente, com muitos estudos em andamento. Além disso, dos 2.138 documentos recuperados em pesquisas, apenas 51 foram incluídos na pesquisa, sendo desse modo, perceptível que as consultas apresentaram um considerável nível de ruído, uma vez que apenas 2,38% dos estudos devolvidos foram realmente relevantes para a pesquisa. Muitas questões podem contribuir para esse resultado, como o uso de uma *string* de busca inadequada ou a ineficiência dos motores de busca automatizados, como discutido por Kitchenham (2004).

Na análise dos resultados do processo de avaliação da qualidade, observa-se que 13 dos 51 estudos primários incluídos na pesquisa, que correspondem a 25,4% do total, foram classificadas como excelente, 19 (37.2%) como muito bom, 15 (29,4%) como bom e 4 (7,8%) como regular, mas nenhum foi considerado ruim.

Este mapeamento sistemático não restringiu o período de publicações, apesar de todos os estudos selecionados terem sido publicados entre 2001 e 2013, como mostrado na Figura 3.2. Isso evidencia que os estudos envolvendo o uso de ontologias para apoiar DDS ainda são recentes, e por esse motivo, a maioria dos estudos (60.7% - 31 artigos) foram publicados entre 2006 e 2013, o que retrata, portanto, a relevância que este assunto em particular tem adquirido recentemente.

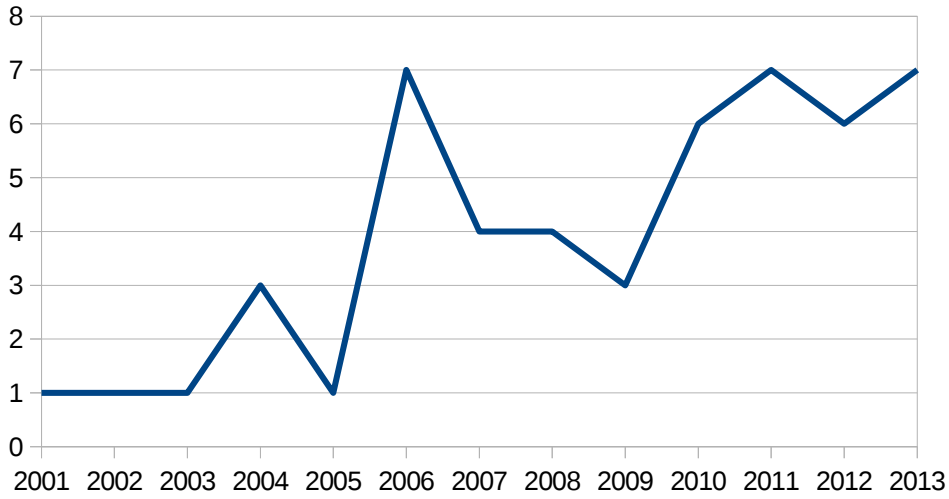


Figura 3.2. Intervalo de Tempo de Publicação.

Assim, o mapeamento contou 138 autores nos 51 estudos primários. A Tabela 3.3 apresenta os autores que possuem estudos sobre o uso de ontologias no DDS e a quantidade de vezes que cada pesquisador apareceu como autor principal de um artigo.

Pornpit Wongthongtham é o pesquisador que mais possui publicações sobre o tema, com autoria em sete periódicos. Este mesmo autor, também mostra que Tharam Dillon e Elizabeth Chang também possuem muitas publicações neste campo, nesse caso, é importante mencionar que os pesquisadores Elizabeth, Tharam e Pornpit trabalharam juntos na autoria de cinco estudos primários. Os autores com

mais artigos encontrados, Wongthongtham, Dillon and Chang possuem trabalhos com IA, demonstrando que fazem parte dessa área também e são professores da Universidade de Tecnologia Curtin, na Austrália.

Tabela 3.3. Lista de Autores

Artigos	Autores
1	Todos os outros
2	Camelia Chira, Frank Maurer, Harald Holz, Ian Sommerville, Thomas Roche.
3	Aurora Vizcaíno
6	Elizabeth Chang, Tharam Dillon.
7	Pornpit Wongthongtham.

As pesquisas se originaram em 18 países, e neste caso, torna-se importante tomar nota de que, em certas publicações, alguns pesquisadores têm sido conduzidos em outros países. Austrália e Alemanha são os países com o maior número de pesquisas e curiosamente o Brasil, a China, a Irlanda e a Índia, que são alguns dos principais destinos para as empresas que estão distribuindo seus processos de desenvolvimento de software em todo o mundo [Prikladnicki, 2003], possivelmente pela qualidade da mão-de-obra e por seu custo, apresentam poucos estudos na área. Conforme apresentado na Figura 3.3.

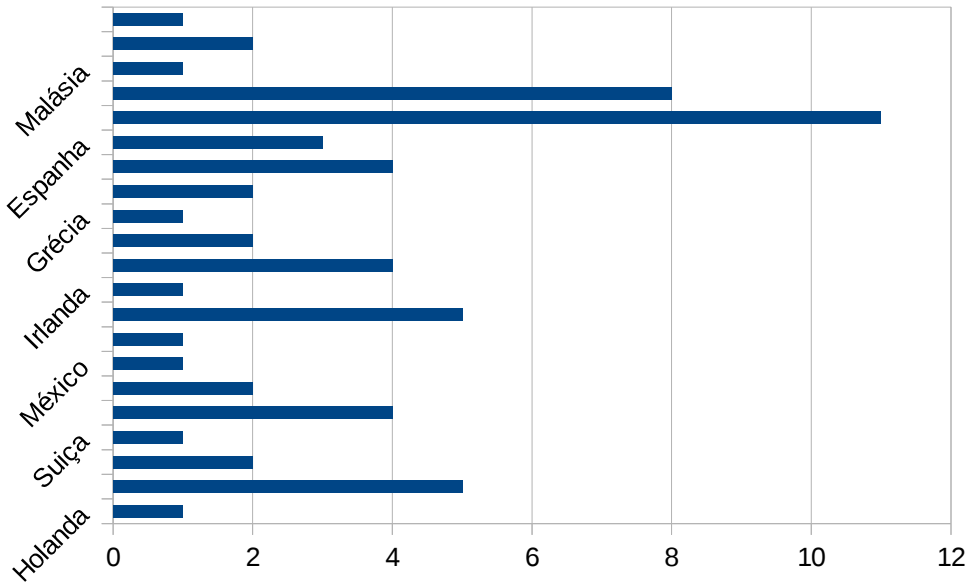


Figura 3.3. Países das instituições de pesquisas.

Na Tabela 4 é apresentada uma classificação dos estudos pelo local de publicação. É interessante notar que a maioria dos estudos foram publicados em anais de congressos na área (conferências, workshops e simpósios). Dos 51 estudos primários, 36 (70.5%) foram publicados neste tipo de veículo, enquanto que apenas 14 (27.4%) estavam em periódicos. É possível ressaltar ainda que uma das obras incluídas nesta pesquisa é uma dissertação de mestrado. Este resultado pode ter sido devido ao maior número de conferências e *workshops* na área de engenharia de software, quando comparado com a quantidade de periódicos.

A Tabela 3.4 também mostra a lista de conferências e revistas que publicaram trabalhos incluídos na pesquisa. Cada conferência publicou um artigo, com exceção de ICGSE, que apresentou duas ocorrências de estudo primário, quanto aos jornais, e as revistas futura geração de sistemas de computador - FGCS e Diário de Arquitetura de Sistemas - JSA publicou dois artigos cada um, enquanto o único remanescente.

Tabela 3.4 Lista de Conferências e Periódicos

Tipo de Publicação	Lista dos lugares de Publicação	Quantidade de Estudos	%
Conferência	WS, OTM, LSO, IIWAS, QUTE-SWA, ICEIS, IW3C2, ICGSE, MARK, CISIS, ICSM, EDOCW, WET-ICE, WIKIS4SE, WI-IAT, ICISTM, ICDCSW, ICSEA, ICSC, INDIN, PAKM, SITIS, SCC, BWCCA, WDSD, WICSA, ICSE, SAC, ICET, CAMP, COMPSAC, ICENCO, WEBIST	36	70.5
Periódico	FGCS, JSA, AES, ESA, RE, JIM, ASE, IET, CLEI, Expert Systems, Information and Software Technology	14	27.4
Teses / Dissertações	PPGCC-FACIN-PUCRS (Pontifical Catholic University of Rio Grande do Sul)	1	1.9

Uma análise do ambiente de desenvolvimento de cada pesquisa também foi realizada e com base nos resultados desta análise é possível notar que a maioria (46 artigos, 90.1%) dos estudos são desenvolvidos no meio acadêmico, enquanto que apenas 5 (8.9 %) na indústria, o que quer dizer que os estudos ainda são incipientes. Para essa afirmação foi observado o âmbito de desenvolvimento dos estudos.

Isso pode significar que é necessário um maior intercâmbio entre a academia e a indústria, de modo que as pesquisas no campo podem evoluir até chegar a um nível de maturidade que incentiva a adoção de recursos baseados em ontologias para apoiar o DSD, a fim de tirar proveito dos possíveis benefícios e melhorar a qualidade no mercado de software.

Outro resultado importante é com relação ao tipo de contribuição de cada estudo. A Figura 3.4 mostra que a maioria dos estudos são pesquisas em andamento (29 artigos, 57%) e apenas 39 % (20 artigos) são validadas.

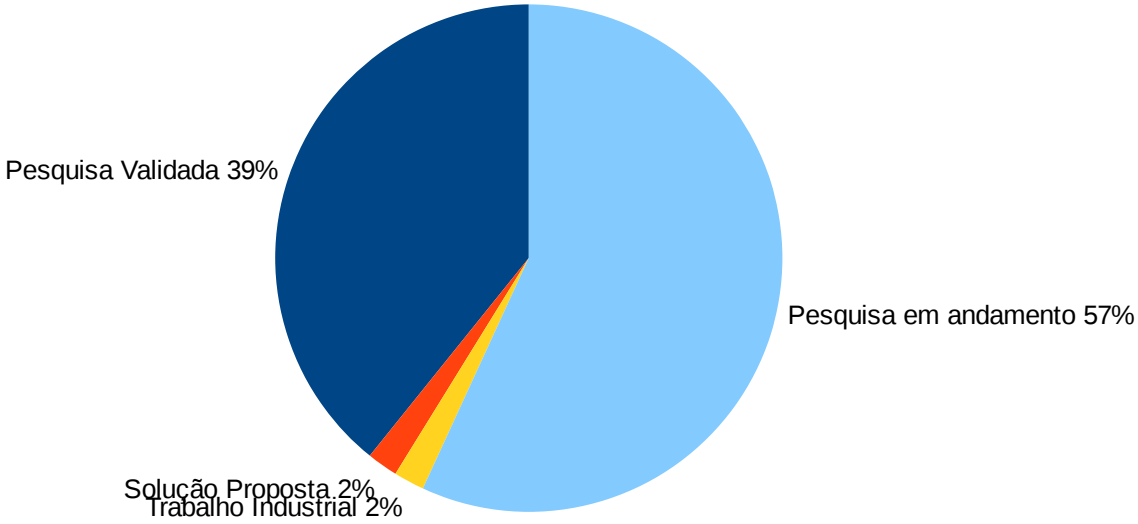


Figura 3.4. Tipo de Contribuição de cada estudo.

Além disso, apenas 3% (um artigo) propuseram abordagens e 3% (um artigo) expuseram experiência de trabalho. Isso mostra que as soluções propostas envolvendo ontologias foram sendo discutidas e desenvolvidas no ambiente de DDS, mas a maioria dessas propostas ainda precisam ser testados e validados para demonstrar a sua eficácia e alcançar os objetivos a que se propõem.

3.1.5.1 Respostas às Questões de Pesquisa

Dentre os estudos selecionados, a pesquisa encontrou evidências relevantes para responder satisfatoriamente as suas questões de pesquisa, dessa maneira, relaciona-se abaixo as principais evidências apresentadas para cada questão de pesquisa (as demais evidências estão resumidas no Apêndice G).

(1) Quais ferramentas, técnicas, modelos e boas práticas utilizam ontologias para apoiar o DDS?

A resposta para essa questão de pesquisa é uma lista de recursos que usam ontologias como suporte a Engenharia de Software no contexto distribuído. O mapeamento feito pela pesquisa, tentando responder a primeira questão de pesquisa, identificou e classificou os recursos encontrados em nove modelos, vinte e sete ferramentas, nove boas práticas e duas técnicas.

Essa primeira questão de pesquisa foi importande no sentido de fornecer meios que possibilitassem um entendimento maior sobre os recursos utilizados com o conceito de ontologias para apoiar o DDS. A partir dessas respostas foi possível analisar melhor as opções de ferramentas, práticas e modelos identificados. Essa análise apoiou a criação de um mecanismo único, uma ontologia de domínio (apresentada do capítulo 4) com intuito de contemplar as limitações encontradas nas respostas dessa questão de pesquisa, bem como na questão de pesquisa seguinte.

A Tabela 3.5 apresenta os códigos de cada estudo primário citado em que todos os recursos receberam essa identificação única para simplificar essa identificação. Dessa maneira, T01 significa a primeira ferramenta encontrada e BP01 significa a primeira boa prática.

Abaixo são apresentadas apenas as ferramentas e os modelos mais relevantes, sendo os critérios de relevância foram baseados nos critérios de qualidade utilizados nesta pesquisa, ou seja, apenas as ferramentas e modelos encontrados em estudos preliminares que foram classificados no processo de avaliação da qualidade como excelente.

Tabela 3.5. Códigos dos recursos encontrados

Tipo de Recurso	Identificação dos trabalhos
Ferramenta	T01 (PS01), T02 (PS02), T03 (PS03), T04 (PS04), T05 (PS06), T06 (PS08), T07 (PS09), T08 (PS11), T09 (PS12), T10 (PS13), T11 (PS14), T12 (PS15), T13 (PS16), T14 (PS18), T15 (PS19), T16 (PS20), T17 (PS23), T18 (PS24), T19 (PS25), T20 (PS26), T21 (PS27), T22 (PS29), T23 (PS30), T24 (PS33), T25 (PS34), T26 (PS36), T27 (PS37), T28 (PS42), T29 (PS43), T30 (PS44), T31 (PS46), T32 (PS48), T33 (PS49)
Modelo	M01 (PS05), M02 (PS07), M03 (PS17), M04 (PS21), M05 (PS22), M06 (PS28), M07 (PS31), M08 (PS32), M09 (PS35), M10 (PS39), M11 (PS45)
Boas Práticas	BP01 (PS01, PS21), BP02 (PS07), BP03 (PS02, PS06), BP04 (PS03, PS23), BP05 (PS05, PS13), BP06 (PS02, PS06, PS07, PS09), BP07 (PS16, PS29), BP08 (PS17), BP09 (PS20)
Técnicas	TE01 (PS07), TE02 (PS29), TE03 (PS41), TE04 (PS43)

Abaixo serão descritos os principais trabalhos encontrados para cada um desses itens, no caso, principais trabalhos referentes aos Modelos, Boas Práticas, Técnicas e Ferramentas encontradas. Esses trabalhos foram considerados principais pela avaliação de qualidade realizada durante o processo de extração do mapeamento. A Tabela 3.6 apresenta algumas das principais ferramentas que utilizam as ontologias como forma de apoiar o DDS encontradas nessa pesquisa. A primeira coluna contém os nomes das ferramentas e os estudos primários aos quais elas fazem parte e a segunda coluna apresenta a descrição de cada ferramenta.

Tabela 3.6. Principais Ferramentas

Ferramentas	Descrição
Ferramenta de Componente de Software [PS01]	Ferramenta para o acesso e manipulação de dados de uma ontologia de componentes de software. A ontologia define conceitos como atributos, operações, características e relacionamentos dos componentes. Assim, facilita a especificação, integração e reuso de componentes de software.
Ferramenta de Compartilhamento de Dados [PS02]	Ferramenta para auxiliar o acesso e compartilhamento de informações entre equipes distribuídas. Utiliza ontologias com dados do projeto e agentes de software para navegar entre os dados e retornar dados como resposta à perguntas levantadas por usuários.
Ferramenta de Suporte a Colaboração [PS03]	Ferramenta para acompanhamento de colaboração entre equipes distribuídas durante o processo de desenvolvimento de software. Uma ontologia define conceitos relacionados com aspectos de colaboração. Assim, permite avaliar a evolução e identificar padrões de colaboração.

PRIME [PS15]	Ferramenta para apoiar processos ágeis e distribuídos que consiste em distribuir informações relacionados com uma tarefa específica. São utilizadas ontologias para descrever e relacionar tarefas e recursos de informação. Assim, permite a organização de informações e facilita o acesso às informações mais relevantes para uma determinada tarefa.
Sistema de Recomendações de Engenharia de Requisitos [PS16]	Esta é uma ferramenta de recomendação utilizada no processo de especificação de requisitos em ambientes ágeis e distribuídos, na qual os analistas têm uma base de conhecimento associado ao domínio do projeto, que pode ser usada para tomar decisões no momento da elicitação de requisitos. Esta base de conhecimento do sistema integra quatro ontologias: a ontologia de domínio do negócio, uma ontologia de requisitos ágeis, uma ontologia requisito geral e uma estrutura de ambiente de um.
TeamWave [PS39]	Este é um framework de código aberto que oferece suporte ao acesso, organização e compartilhamento de conhecimento sobre o projeto entre as equipes de desenvolvimento de software distribuído. Uma de suas principais, ampliação automática da base de conhecimento por meio da captura e inferências sobre dados de interação do usuário.

Na tabela 3.7 são apresentados os principais modelos identificados nesse estudo que fazem uso de ontologias para apoiar o DDS, sendo a primeira coluna a apresentação do nome do modelo e a identificação do estudo primário que ele foi encontrado e a segunda, uma exposição sobre a descrição de cada modelo.

Tabela 3.7. Principais Modelos

Modelos	**Descrição**
SVTFM [PS05]	É um modelo para auxiliar a formação de equipes virtuais que envolve um método quantitativo, ontologias com dados do projeto e competências de profissionais, e um software de busca de informações de profissionais em fontes específicas da Web.
ESEM [PS17]	É um Modelo de engenharia de software empresarial para integração de informações relacionadas com os artefatos do projeto. O modelo envolve ontologias para definir terminologias e relações dos artefatos, além de ferramentas para acessar e compartilhar informações dos artefatos.
SWS-ASE [PS28]	Esse modelo automatiza a seleção e combinação de serviços de ferramentas de apoio ao processo de engenharia de software, com base em técnicas de web semântica e ontologias para definir conceitos e relações das ferramentas.
RE [PS31]	É um Modelo de Engenharia de Requisitos que usa técnicas para o

	Gerenciamento de Conhecimento e Onotlogias como suporte a fase de Requisitos. As Ontologias são usadas para formalizar o conhecimento relacionado aos requisitos do projeto e conhecimento organizacional.
DISEN [PS35]	Esse Modelo dissemina de informações textuais em ambientes de DDS que envolve ferramentas de colaboração e uma ontologia para descrever os conceitos e elementos compartilhados,

(2) Quais ontologias propostas e ou adotadas para o contexto de DDS?

Essa questão objetivou encontrar quais as ontologias formalizadas e adotadas para o DDS. Respondendo essa questão, 8 ontologias foram encontradas e as principais expostas na Tabela 3.8 onde são apresentadas as propostas de ontologia para o contexto distribuído, sendo a primeira coluna os nomes e a identificação de cada ontologia perante os estudos primários, e a segunda, a descrição de cada ontologia apresentada.

Tabela 3.8. Ontologias para o DDS

Modelos	**Descrição**
OFFLOSC [PS10]	Esta ontologia é formalizado para o contexto das comunidades de desenvolvimento de software de código aberto. Seu objetivo é ajudar a coordenar as atividades, gestão de recursos e de compartilhamento de conhecimentos sobre e para os membros das equipes. É composto por 46 classes e descreve os conceitos relacionados a comunidades *open-source*, como atores, artefatos, atividades, operações, relacionamentos e recursos.
Knowledge Management Ontologies [PS18]	Um conjunto de ontologias que formalizam conceitos estruturais de ambientes DDS, direcionadas ao gerenciamento de conhecimento. Esse conjunto de ontologias tem como principal função descrever os conceitos dos artefatos de software, problemas de ambiente, interação entre os companheiros de equipe distribuídos, infra-estrutura, regras de negócio e informações gerais do projeto.
Open Source Communities [PS32]	Esta ontologia também é formalizada no contexto de desenvolvimento de software de código aberto e seu principal objetivo é compor uma base de conhecimento do projeto, ter dados categorizados semanticamente relacionados, o que permite a realização de buscas semânticas e inferências de dados por agentes inteligentes. É composto por seis classes que descrevem conceitos de relações, regras, atividades, processos, artefatos do ator e ferramentas de projetos das comunidades de código aberto.
OntoDISEN [PS35]	Essa ontologia é formalizada para o cenário de projetos distribuídos e é usada para auxiliar o estabelecimento de comunicação entre equipes distribuídas. Ele é integrado a um modelo de divulgação de informação textual, permitindo o compartilhamento de informação em ambientes distribuídos para ser compreendida por todos os engenheiros de

	software, de forma clara e homogênea. Descreve os conceitos de elementos que são representados e compartilhados em um ambiente DSD, como usuários, ferramentas, outros ambientes, atividades e processos.

3.1.5.2. Análise de Dados

Esta seção discute os resultados encontrados neste mapeamento sistemático. Os recursos utilizados para apoiar a Engenharia de Software encontrados nesta pesquisa foram classificados de acordo com as áreas de conhecimento definidas no SWEBOK (2004). Essa seção também apresenta uma análise de todas as ontologias encontradas neste estudo, e por fim, discute os problemas que motivam a utilização de ontologias e as vantagens da utilização deste recurso no DDS.

A atuação das ferramentas, modelos, técnicas e boas práticas que utilizam ontologias no suporte ao DDS identificadas neste mapeamento foram levantadas com intuito de planejar quais áreas do desenvolvimento de software são consideradas, dessa forma, as que possuem mais e menos suporte. Os recursos foram categorizados de acordo com as 10 áreas de conhecimento de Engenharia de Software definidos no guia SWEBOK. É importante notar que alguns recursos foram inseridos em mais de uma área de conhecimento.

Desse modo, a Tabela 3.9 apresenta a classificação dos escopos dos recursos de acordo com o SWEBOK:

Tabela 3.9. Distribuição de recursos pelas áreas de conhecimento do SWEBOK

Áreas de Conhecimento	Ferramentas	Modelos	Boas Práticas	Técnicas
Requisitos	T7, T13, T18, T22, T23, T25, T27, T29, T32, T33	M4, M7	BP7	TE2, T304
Projeto	T3, T9, T21, T27	M10		TE03
Construção	T1, T4, T24, T28, T32	M4, M5	BP1, BP8	
Testes	T20, T23	M4		
Manutenção	T1, T11, T24		BP1, BP8	
Gerenciamento de Configuração	T01, T22		BP1, BP8	
Gerência	T1, T2, T3, T15, T16, T23, T28, T30, T31	M1, M2, M6	BP5, BP6, BP8, BP9, BP10	TE1
Process	T2, T3, T4, T6, T8, T10, T12, T13, T23	M3, M6, M9, M11	BP2, BP3, BP4	
Ferramentas e Métodos	T4, T5, T6, T8, T10, T(12-19), T22, T23, T25, T26	M2, M3, M5, M8, M9	BP4	

Qualidade		M3		

A partir da Tabela 4.8 é possível identificar que várias ferramentas, boas práticas e modelos foram classificados nas áreas de conhecimento, porém poucas técnicas foram encontradas. Acredita-se que seja mais fácil relatar uma ferramenta desenvolvida, um modelo criado e utilizado ou a adoção de boas práticas do que as técnicas em si, geralmente utilizadas pelas organizações para melhorar a produção.

Neste sentido, é importante considerar que a maioria dos recursos classificados nesta área também foram inseridos em outras áreas, por exemplo, Assawamekin et al. [PS29] apresentou uma ferrramenta (T22) para automatizar o processo de engenharia de requisitos, ou seja, um recurso com escopo nas áreas de Requisitos de Software e Métodos e Ferramentas.

Além disto, muitos dos recursos encontrados nesta pesquisa estão focados nas áreas de Gestão de Engenharia de Software e Processos de Engenharia de Software. Nesta última área, foram identificadas 9 ferramentas, 4 modelos e 3 boas práticas. Destas, duas ferramentas (T12, T13) são voltadas para processos ágeis e dois modelos (M03, M09) são focados na melhoria de processos através da utilização de ontologias para prover um ambiente de captura de conhecimento integrado ao processo de software.

No que diz respeito a área de gerenciamento, foram identificadas 9 ferramentas, 3 modelos e 5 boas práticas. Esses recursos focam em diferentes tópicos, como gerenciamento de colaboração (T15, T03, BP05), gerência de artefatos do projetos (T01, M01, BP08), relacionamento entre clientes de *offshore outsourcing* (T25) e aspectos relacionados ao gerenciamento de projeto (T02, T16, T23, M02, M06, BP06, BP09, BP10).

Outra área bastante beneficiada com a utilização de ontologias é ER, nela foram encontradas 10 ferramentas, 2 modelos, 2 técnicas e 1 boa prática que utilizam ontologias para estruturar conceitos de requisitos de software em ambientes distribuídos. Assawamekin et al. [PS29], por sua vez, apresentou o desenvolvimento de uma ferramenta (T22) que utiliza ontologias para gerar automaticamente a rastreabilidade de relações dos diferentes requisitos. Por outro lado, Aranda et al. [PS09] utiliza ontologias como um mecanismo para facilitar a definição e compartilhamento do conhecimento relacionado com requisitos entre equipes distribuídas.

Já Angrisani [PS31] apresenta em seu trabalho um modelo (M07) de gestão de conhecimento baseado em ontologias para auxiliar o trabalho de engenheiros de requisitos em de DDS e segundo ele, a utilização de ontologias no processo de requisitos permite compartilhar informações de forma padronizada e coerente, sendo possível diminuir a ambiguidade na intepretação de conceitos e permitir que as definições de requisitos possam ser mais bem controladas e gerenciadas.

Quatro ferramentas (T03, T09, T21, T26) foram projetadas para o suportar a área de Projeto. Tang et al. [PS37] apresenta um sistema (T27) para indexação de documentos de arquitetura de software e utiliza ontologias no processo de captura e relacionamento de informações arquiteturais do projeto. Este sistema foi testado e algumas vantagens com a utilização de ontologias nesta área da engenharia de software foram encontradas, são elas: a) melhor padronização e gerenciamento do conhecimento arquitetural; b) melhoria na tomada de decisões arquiteturais; c) melhor análise e avaliação da qualidade de design; e, d) facilidade na gestão de mudanças.

Baseado nesses resultados torna-se evidente que as fases do desenvolvimento são beneficiadas com o uso de ontologias, principalmente Processo, Gerenciamento, Requisitos e Projeto. Por outro

lado, alguns importantes campos não foram abordados, como por exemplo, Qualidade e Testes, que envolve um grande número de atividade de gerenciamento de informação e podem ser considerados uma evolução com a utilização de ontologias como forma de padronização, gerência e compartilhamento de conhecimento.

A partir do mapeamento sistemático é possível afirmar que diversos insumos importantes foram identificados, permitindo que esta pesquisa seja considerada inovadora na área e que possa ser utilizada como base para novas pesquisas sobre DDS, seja em um contexto mais geral ou mais específico.

Respondendo a segunda questão da pesquisa, foram encontrados quatro trabalhos que propõe algumas ontologias desenvolvidas especificamente para o desenvolvimento distribuído de software. Essas ontologias mencionadas foram desenvolvidas especificamente para equipes distribuídas, dessa forma, possuem conceitos e características para esse contexto. É possível observar que destas 4 ontologias, 2 são voltadas para comunidades de desenvolvimento de software *open-source*.

De acordo com Mirbel [PS10], a natureza dinâmica e livre deste tipo do ambiente distribuído traz desafios na coordenação das atividades e compartilhamento de conhecimento. Por isso, a utilização de ontologias no suporte ao processo de desenvolvimento de software em ambientes *open-source* facilita a gestão dos recursos de conhecimento da comunidade.

Também é interessante observar que muitos outros trabalhos utilizam ontologias para resolver ou mitigar desafios em ambientes distribuídos, porem, essas ontologias não são específicas para esse ambiente, são ontologias para a Engenharia de Software, que utilizadas num projeto Distribuído podem entra como uma forma de auxílio. Na Tabela 3.10 a seguir, constam as diversas ontologias encontradas neste estudo:

Tabela 3.10. Lista de Ontologias que são usadas para auxiliar o processo de desenvolvimento

Campo	**Estudos Primários (Primary Stydies)**
Componentes de Software	PS01
Domínio de Negócio	PS02, PS09, PS16, PS27, PS31
Engenharia de Software	PS02, PS06, PS07, PS08, PS21, PS28, PS30, PS38
Gerência de Configuração	PS02
Soluções e Problemas	PS02
Estrutura Colaborativa	PS03, PS04, PS14, PS23
Divisão de Time e atribuição de Funções	PS05, PS13
Dados Gerais do Projeto	PS06, PS08, PS19, PS20, PS22, PS25, PS27, PS30, PS31, PS33, PS34
Requisitos de Software	PS09, PS16, PS24, PS29, PS31
Código e Bugs	PS14
Artefatos de Software	PS17

Teste de Software	PS26
Arquitetura e Projeto de Software	PS27, PS37
Serviços Linguísticos	PS34
Constituição do Contrato Eletrônico	PS36

Como visto na Tabela 4.9, muitas são as ferramentas que utilizam ontologias não específicas ao DDS apenas para mitigar seus desafios e limitações, essas ferramentas são distribuídas nas mais diversas áreas existentes num projeto, desde áreas reais da ES, como áreas de atuação especificas que são utilizadas para dar suporte a uma área ou as atividades de uma ou mais áreas.

De modo como já citado anteriormente, diversos são os desafios enfrentados com o Desenvolvimento Distribuído de Software, assim, parte desta pesquisa busca analisar, através dos relatos dos trabalhos encontrados, quais os problemas relatados pelos participantes que terminam por ser a motivação principal para utilização de ontologias como recurso de apoio, dessa forma, traça-se uma lista de desafios:

- Falta de padronização dos conceitos e terminologias utilizadas por cada equipe distribuída;
- Compartilhamento de informações relacionadas com componentes de software, o que dificulta o entendimento, a integração, o tratamento de erros e o reuso de diferentes componentes criados por diferentes equipes, principalmente componentes terceirizados;
- Compartilhamento e gerenciamento de informações do projeto, resultando na falta de conhecimento dos papeis e pessoas do projeto, bem como as atividades desenvolvidas e as questões levantadas e resolvidas por equipes ou desenvolvedores;
- Tarefas mal interpretadas e dificuldades em seguir o plano de projeto;
- Menor interação entre as equipes, dificultando a troca de experiências, discussões e tomadas de decisões em tempo real;
- Dificuldade de gerenciar o processo de colaboração e comunicação entre equipes distribuídas;
- Dificuldade em integrar grande quantidade de informações originadas de diferentes fontes do projeto;
- Lacuna de diferentes línguas, culturas e sistemas sociais existentes entre os clientes e os engenheiros de software.

Dessa forma, é possível perceber que esses fatores citados pelos próprios trabalhos analisados, como motivos para se usar ontologias em equipes distribuídas, são comuns na Engenharia de Software convencional, ou seja, boas práticas e soluções para esses problemas já existem, porém, no contexto de DDS esses problemas se tornam mais acentuados, fazendo com que seja necessário maiores cuidados e atenção.

Outra importante resposta que o trabalho busca é saber quais as vantagens que as ontologias podem oferecer para ambientes distribuídos. Dessa maneira, segue abaixo o que os trabalhos relatam sobre tais vantagens:

- Aperfeiçoamento na definição e compartilhamento de conceitos do domínio de Engenharia de Software com as propriedades: possibilidade de máquina de processamento semântico, definição explícita, conhecimento consensual e modelo abstrato;
- Facilitar a definição, a padronização, o relacionamento semântico e o compartilhamento do conhecimento relacionado com os componentes de software desenvolvidos por equipes distribuídas, permitindo melhor descrição, integração e reuso dos componentes;
- Melhoria de qualidade no processo e, consequentemente, no produto, com um compartilhamento comum do conhecimento;
- Facilidade de agregar e relacionar semanticamente vários conhecimentos importantes para o processo de ES, possibilitando a criação de uma plataforma integrada de dados semânticos, que pode ser processada por agentes inteligentes de software para realizar raciocínios (inferências) de forma autônoma, distribuir informações do processo entre equipes distribuídas e gerar novos conhecimento;
- Maior riqueza semântica aos conceitos e informações relacionadas com o projeto, permitindo uma representação clara, intuitiva, precisa e não ambígua de conceitos, ideias, questões, soluções e documentações do projeto;
- Descrever uma estrutura genérica para relacionar logicamente um vocabulário padrão para a descrição de atividades dentro de uma rede de colaboração entre equipes distribuídas, assim é possível controlar (em tempo real) e melhorar as atividades de comunicação entre equipes distribuídas, além de permitir realizar raciocínio formal para identificar padrões de comunicação, derivar fatos implícitos e criar procedimentos de tomada de decisões;
- Organizar e relacionar informações dos engenheiros de software, como habilidades técnicas, conhecimento de ferramentas, habilidades gerais, entre outros, assim torna-se possível realizar inferências a fim de encontrar membros com habilidades específicas e fazer uma melhor distribuição das atividades do projeto;
- Estruturar e permitir buscas e consultas ao conhecimento arquitetural do projeto, além de auxiliar no processo de captura e relacionamento (*trace*) de requisitos de software e documentos arquiteturais de software;

Com esses resultados, pode ser observado que as vantagens de utilizar ontologias no suporte ao DDS são muitas, principalmente para gerar soluções visando a mitigação de problemas de comunicação, colaboração, gerenciamento de fluxo de conhecimento, coordenação de atividades, conhecimento do projeto e gerenciamento de processo para o DDS. O catálogo completo com os trabalhos citados sobre os desafios e vantagens estão disponíveis no Apêndice G.

A partir desse mapeamento foram realizadas diversas leituras, análises e observações sobre ontologias para DDS, ontologias para ES que são utilizadas para melhor aproveitamento no projeto de software, bem como sobre modelos, ferramentas, boas práticas e desafios. Isto permitiu a identificação do problema de pesquisa bem como a solução proposta neste trabalho. Através do mapeamento foi possível perceber que ontologias no contexto de DDS são estudos incipientes e que a maior parte dos trabalhos entre ES e ontologias, são

Os modelos, ferramentas, técnicas, boas práticas e desafios que envolviam DDS e ontologias foram identificados no sentido de fornecer um *background* sobre os recursos disponíveis nesses dois contextos. Para cada recurso, análises detalhadas foram feitas, indicando a aplicação destes e seus

resultados. Estes recursos serviram de base e insumo para a criação da DKDOnto, dessa forma, os conceitos envolvidos nessa ontologia foram também provenientes dos trabalhos que citavam esses recursos, bem como parte das regras de funcionamento do domínio. Outro fator positivo, é permitir que outros estudos se iniciem a partir desse *background*.

Dessa maneira, esse trabalho propõe a DKDOnto, uma ontologia de domínio específica para o ambiente distribuído, que utilizou como base informações propostas nos estudos primários para aproveitar melhor os recursos disponíveis. Ela é apresentada no capítulo seguinte.

3.1.6. Ameaças à validade

Essa seção discute as ameaças que podem ter alguma interferência nos resultados apresentados anteriormente. A primeira ameaça à validade considerada por esse trabalho foi a decisão por uma área relativamente nova e com questões de pesquisas bem específicas. Uma forma de minimizar seria deixar uma única questão de pesquisa e que esta fosse mais ainda mais ampla.

A estratégia de busca através de engenhos de busca automatizados, pode ser considerado como uma ameaça à validade, pois estudos relevantes podem não ter sido encontrados pelas seleções e buscas realizadas. Assim como estudos mais recentes, mesmo que dos anos atuais, podem não estar contemplados na base. Uma possível solução, seria uma busca manual, por referências ou por periódicos e conferências.

Outra ameaça foi a seleção de estudos primários referentes a diferenciação entre os trabalhos que seriam classificados como técnicas ou modelos. Pela similaridade ou dificuldade de identificação de ambos, esse foi um fator considerado. Porém, como ideia de minimizar isso, foi realizada a revisão da seleção por meio de outro membro da equipe.

Outro fator que pode ser visto como uma ameaça é a quantidade de resultados primários existentes, no caso apenas poucos estudos primários. O que prova também que a área teve aumento na quantidade de estudos nos últimos anos.

3.2. Extração de informação para povoamento da ontologia e construção de casos

Essa seção descreve o procedimento utilizado para a extração de informação cujo objetivo principal é extrair informações de artigos científicos objetivando a criação de casos para serem usados pela abordagem, bem como criar instâncias para a ontologia proposta.

Para a realização das buscas, por questão de padronização e também por ser uma área de pesquisa relativamente nova, foram contempladas para a seleção de artigos apenas os anos de 2006 até 2013 das conferências. Dessa maneira, durante o período de dezembro de 2013 a agosto de 2014 foi realizado a leitura e a extração de informações sobre artigos que descrevessem projetos reais no contexto distribuído. O objetivo dessa busca foi identificar as características dos projetos, os desafios e as soluções encontradas por eles.

Como fonte das buscas, foram definidas algumas das principais conferências nacionais e internacionais de Engenharia de Software e DDS, são elas:

- International Conference on Global Software Engineering (ICGSE)
- International Conference on Software Engineering (ICSE)
- Simpósio Brasileiro de Sistemas Colaborativos (SBSC)
- Worskshop de Desenvolvimento Distribuído de Software (WDDS)

- International Conference on Software Engineering and Knowledge Engineering (SEKE)

Desse modo, o processo de leitura e seleção de artigos que se relacionavam a projetos de DDS, e apresentavam de fato as características dos projetos e as descrições das experiências, se dividiu em duas etapas, a primeira de seleção dos artigos e a segunda de extração de informação.

Na primeira, os artigos eram analisados através de leitura de títulos, resumo e conclusões e se houvesse alguma informação a respeito de descrição de suas características, estas seriam inseridas. A segunda etapa foi a respeito da extração de informação, onde os artigos eram lidos detalhadamente e todas as descrições referentes ao projeto propriamente dito, contendo informações gerais sobre o projeto, bem como específicas a respeito de tecnologias, ferramentas, pessoas e papéis. Estas informações eram colocadas em uma planilha criada para permitir a visualização das informações sobre os projetos de forma resumida e organizada. Participaram do processo de busca e extração de informação cinco alunos de graduação em Ciência da Computação.

Os campos utilizados para retirar as informações dos artigos foram: Código artigo, Autor da extração, Título Artigo, Evento, Ano, Tipo do Projeto (indústria / academia), Breve Descrição Projeto, Num. de Participantes, Locais e Membros Envolvidos, Tecnologias Envolvidas, Artefatos Gerados, Papeis utilizados, Meios de comunicação empregados, Problemas conhecidos previamente, Problemas no decorrer, Adequações/Soluções, Certificações, Instituições Envolvidas

Ao final das seleções 112 artigos foram identificados estudos promissores, pois poderiam conter informações úteis para criar casos ou fazer povoamento inicial da ontologia. A partir desses, 45 casos foram gerados e as instâncias foram colocadas nos projetos.

3.3. *Survey* para Aquisição de Casos da Indústria

Para aproximar a criação dos casos com a realidade, foi realizado uma pesquisa com profissionais que trabalham/trabalharam com DDS. A partir disso, novos casos com descrições reais de cenários reais foram utilizados para este trabalho.

O questionário foi enviado para diversas listas de discussão sobre temas diversos na área de desenvolvimento de software, com intuito de que pessoas que já trabalharam com DDS respondessem, a fim de compartilhar os desafios já enfrentados. O formulário também foi submetido diretamente para profissionais específicos da área também.

O formulário, que pode ser visto no endereço https://docs.google.com/forms/d/16ykVLi_et670_5AiuuZN1NAvPWacs2WBuYXo7kyqQY0/viewform?usp=send_form, tinha três questões específicas sobre os desafios encontrados, as soluções que foram utilizadas e a efetividade dela, exatamente as mesmas informações utilizadas nos casos. Dessa maneira, quando o participante respondia essas três questões que eram de caráter obrigatório, era relatado exatamente um caso enfrentado por este participante. É importante ressaltar que essas informações independiam da empresa que ele trabalha ou trabalhou, ou seja, o participante colocaria os três casos que já enfrentou, e opcionalmente, adicionaria outros, porém, cada resposta sua poderia ser de empresas diferentes.

A partir das respostas ao questionário, 57 casos foram criados/utilizados e então, foi possível traduzir as respostas na íntegra para o inglês, sem comprometimento no momento da tradução. Uma das questões do questionário era identificar se o projeto era industrial ou acadêmico, dessa maneira, dos 57 casos válidos inseridos, 17 casos foram de cunho acadêmico e 40 casos de projetos industriais. No total, 21 participantes responderam ao questionário. Esse número se justifica pelo curto tempo em que o questionário se manteve disponível e também pela amplitude de empresas e pessoas foram abordadas.

Dessa forma, através das respostas, foram constatadas 23 empresas/instituições diferentes, envolvidas em 7 estados diferentes, incluindo uma delas dos USA, são elas: CESAR – PE, REDU – PE, COMPESA - PE, Motorola – PE, Dataprev – PB,Thoughtworks – PE, IFPE – PE, Facene - PB – RN, E.Life – PE/SP, Samsung – AM, HP– USA, Unimed – CE, Kurier – PE, IFPE - PE, Capital Login – PE, PARQTC–PB, UFPE - PE, BRQ – CE, UFCG – PB, Eco Engenharia – PB, Dax Tecnologia – PE, Faculdade Joaquim Nabuco – PE, Tribunal de Justiça – PB.

Entre os diversos papéis existentes no processo de desenvolvimento de software, foram identificados nessa pesquisa a participação de 9 papeis, são eles: Gerente de Projeto, Analista de Qualidade, Arquiteto, Programador, Analista de Testes, Analista de Sistemas, Auditor de Sistemas, Analista de Segurança e Designer.

No total, somando os 45 casos casos gerados na seção anterior com os 57 gerados pelo *survey*, o resultado é de 102 casos que são utilizados pela base de casos e testes realizados, descritos nos capítulos seguintes.

3.4 Resumo do Capítulo

Neste capítulo o objetivo foi fazer uma descrição geral e detalhada dos principais métodos utilizados no trabalho para que fosse possível descrever todas as grandes atividades envolvidas no decorrer deste trabalho de doutorado.

A primeira seção apresenta o mapeamento sistemático realizado, seus resultados e a análise dos resultados, apresenta também a atividade de seleção e extração de informação de artigos científicos, bem como um survey realizado para a construção de casos.

4. Abordagem Baseada em Ontologias e Raciocínio Baseado em Casos

Nesse capítulo são abordadas as principais contribuições do trabalho, iniciando pelos resultados do mapeamento sistemático, que contribuiu significativamente como insumo para planejamento e execução dessa tese. Em seguida, a ontologia DKDOnto é apresentada, em seguida com o sistema de acesso e manipulação das informações da ontologia, e por último, com o sistema de raciocínio baseado em casos. Para finalizar o capítulo, são apresentados os casos de testes realizados para validação dessa proposta.

4.1 Descrição Geral da Abordagem

A abordagem proposta consiste em dois sistemas que acessam a ontologia proposta para o contexto distribuído (DKDOnto). O primeiro sistema (DKDs) permite o acesso e manipulação dos dados contidos na DKDOnto. O segundo (CBR-DKDs) é responsável pela recomendação dos casos para os problemas que podem surgir no projeto, através das técnicas de ambos RBC e PNL, utilizando DKDOnto como sua base de conhecimento.

A Figura 4.1 mostra o fluxo de trabalho da abordagem proposta e um resumo de sua arquitetura, em que o usuário tem a opção de acessar as ferramentas: ou os DKDs ou o CBR-DKDs e realizar suas pesquisas e atividades.

Os dois sistemas propostos são complementares. O primeiro permite aos seus utilizadores executar pesquisas diferentes, assim como a manipulação de dados e é mais adequado para os gerentes de projetos ou líderes técnico, embora também possa ser usada por pessoas que executam outras funções no projeto. No segundo sistema, os usuários podem realizar buscas na base de casos e todos os interessados podem usá-lo. Ambos os sistemas têm acesso à ontologia (DKDOnto). As contribuições deste trabalho são descritos em mais detalhes nas seções seguintes.

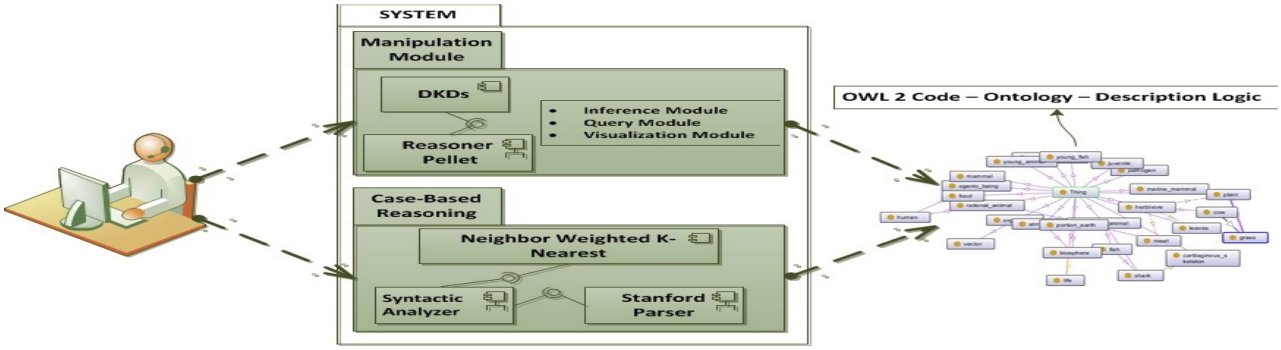


Figura 4.1. Funcionamento Geral da Abordagem

A ideia de combinar RBC com ontologias já foi resultado de algumas pesquisas [Beißel, 2011] [Assali, Lenne e Debray, 2009] [DeMiguel, Plaza e Días-Agudo, 2008] [Roth-Berghofer, Adrian e Dengel, 2010], com o objetivo de explorar melhor o recurso de ambas as técnicas. Porém, Kowalski *et al.* (2013) de maneira mais formal e com avaliações mais validadas, propõe um protótipo em que o sistema RBC pode ser integrado a ontologia demonstrando a viabilidade dessa integração. Dessa maneira esse trabalho faz a integração desses conceitos visando explorar melhor a efetividade dos dois

conceitos. Segue nas próximas seções a descrição, funcionamento geral e recursos utilizados para a construção da abordagem com RBC.

Seguindo a ideia de Kowalski et al. (2013), após as características da descrição do novo problema serem extraídas é realizada uma busca nos resultados retornados na fase de análise sintática retornada pelo Stanford Parser, usando as tags dentro de uma mesma expressão ou frase gramatical, pesquisando-as uma a uma na ontologia (nas classes específicas) e assim compararando seus significados e distância delas utilizando o algoritmo de similaridade adotado nesta tese.

Dessa maneira, o módulo RBC também utiliza classes específicas da ontologia, especificamente três classes, a classe de Challenges, BestPractices e Effectiveness. Nessas classes estão as instâncias referentes aos desafios enfrentados nos projetos, suas soluções utilizadas e suas respectivas efetividades. Assim, o sistema RBC possui uma funcionalidade que traz todas essas instâncias armazenando-as no banco, permitindo que este faça a busca por similaridade das sentenças informadas através do banco.

4.2. DKDOnto: Ontologia Proposta

A DKDOnto foi desenvolvida utilizando as metodologias da Engenharia de Ontologias, Methontology [Fernandez, Gomez-Perez e Juristo, 1997] e IEEE Standard (1997) para desenvolvimento de sistemas de informação baseado em conhecimento e o método 101, proposto no trabalho Noy e Mcguinness (2003) como um complemento ao Methontology.

Dessa forma, a linguagem utilizada para construção da ontologia foi o OWL que incorpora facilidades para a publicação e compartilhamento de ontologias OWL (2009), além de ser proposto como padrão para o W3C, absorvendo os pontos fortes das linguagens anteriores.

Conforme mencionado, essa ontologia se baseou em recursos e informações propostos nos artigos de outras ontologias, porém, não foi possível inicialmente utilizar esses recursos diretamente das ontologias porque as mesmas não estão disponibilizadas, impossibilitando maior detalhamento sobre seus recursos e funcionamento. Também foi planejado reutilizar termos e recursos prontos de outras ontologias, que não tivessem relação com DDS mas que fossem úteis por já serem validadas e consolidades, porém, para o projeto inicial, o escopo não contemplou isso.

Esta ontologia pode ser útil para pesquisadores das áreas envolvidas, mas principalmente dos pesquisadores em DDS. A ideia inicial é que o sistema e a ontologia sejam disponibilizados para testes e download, de duas maneiras: o primeiro, o sistema como todo pode ser utilizado por uma empresa a fim de verificar seu funcionamento e seus resultados; e o segundo, a ontologia pode ser baixada isoladamente, para realização de pesquisa, melhorias e sua utilização em si. Inicialmente, conforme já citado, ela foi instanciada apenas para realizações de testes com poucos dados, retirados de projetos relatados no capítulo 3. A DKDOnto será disponibilizada no endereço www.rgcrocha.com/dkdonto, para maiores informações e acesso a esta.

A OWL é uma linguagem de ontologia (*Semantic Web*, Berners-Lee e Lassila, 2001) com expressividade extrema e poderosa para inferência de conhecimento e para editá-la, foi utilizado o Protégé (2009) que é um editor de ontologia gratuito, livre, baseado em Java e um framework para conhecimento.

Seguindo o procedimento inicial, foram criadas 20 questões de competências que identificam a motivação do cenário através dessas questões, ou seja, o que a ontologia deve responder. Neste caso, as 20 questões são respondidas sobre o contexto no qual a ontologia foi fundamentada, esta etapa

possibilitou a definição da terminologia, ou seja, seus objetos, atributos e relações. Segue abaixo como exemplo, 4 das 20 questões de competência. As demais questões podem ser visualizadas no Apêndice J.

- Quais os problemas de comunicação entre idiomas nos emails, workspaces dos artefatos e conversas por telefone ou VOIP, em um projeto que é distribuído entre 5 países diferentes e utiliza a língua inglesa como idioma principal?
- Quais as fases, os papéis e os artefatos de uma metodologia específica de desenvolvimento de software para um projeto que está distribuído entre cidades distintas em um mesmo país, porém com o cliente localizado em outro país?
- Em um projeto no qual os participantes estão distribuídos entre 4 países diferentes, como seria possível formar as equipes e distribuir as atividades com base nos conhecimentos com o cuidado nos respectivos fusos horários?
- Quais os riscos enfrentados por um gerente em um projeto distribuído sem o conhecimento dos membros que compõe os times do projeto em cidades e regiões distintas, porém em um mesmo país?

Dessa maneira, a DKDOnto possui aproximadamente 50 classes, mas por questão de espaço, apenas as principais classes são apresentadas nessa seção, uma maior quantidade de classes pode ser vista no Apêndice C.

- Project: a classe principal dessa base de conhecimento, responsável por armazenar todas as informações referentes às configurações do projeto, desde os membros alocados até fases, atividades e artefatos utilizados.
- Member: é uma subclasse de Resource na qual membro é um indivíduo que tem acesso ao ambiente e pode ser alocado em projetos. Um membro possui habilidades, trabalha em um lugar e participa diretamente do projeto, reportando aos usuários ou gerência as melhores práticas e desafios, utilizando e criando os artefatos.
- BestPractice: todas as soluções e melhores práticas usadas para qualquer problema devem ser armazenadas nessa entidade. Essa classe é responsável por evitar desafios e problemas encontrados e reportados por um membro durante a execução de suas atividades, e também resolver esses problemas e desafios.
- Challenge: todos os desafios encontrados por membros devem ser armazenados nessa classe. Um desafio pode utilizar boas práticas para ser solucionado. Essa entidade é fundamental porque os desafios possuem soluções ou boas práticas associadas com algumas práticas que podem ser utilizadas e disponibilizadas para outros membros com os mesmos problemas.
- Skill: todo o conhecimento dos membros é armazenado nessa entidade. As habilidades dos membros permitem que os problemas sejam evitados ou que os mesmos sejam também resolvidos. Essa classe também permite que as atividades sejam distribuídas para os membros de acordo com suas habilidades.
- Place: é uma classe fundamental para definir exatamente onde os membros envolvidos no projeto se encontram. Essa entidade armazena todas as informações sobre a localização dos membros, definindo qual o nível de dispersão e a distância temporal do projeto.
- Artifact: é a classe utilizada por quase todas as outras classes principais, pois dá suporte aos

membros e suas atividades. As ferramentas também podem usar artefatos em atividades específicas.

- Tool: classe responsável por todas as ferramentas envolvidas no projeto a qual permite que todos os usuários saibam quais ferramentas são utilizadas por outros membros e outros projetos. Dessa forma, é possível seguir os padrões e encontrar informações e instruções específicas para o uso dessas ferramentas.
- Workspace: é uma classe que contém os artefatos e as ferramentas que os usuários podem usar e criar em suas atividades. Todos os usuários alocados no projeto podem acessar a Workspace para *commit* e *checkout* de todos os documentos, artefatos ou ferramentas e possui como objetivo principal, o armazenamento dos artefatos em que os membros utilizam a Workspace de um projeto específico.

Na Figura 4.2 são apresentados alguns relacionamentos entre as principais classes definidas na DKDOnto. O diagrama foi gerado a partir do plugin Ontoviz do Protégé. Por questão de espaço, apenas um restrito conjunto de classes foi escolhido para ser exibido. Todas as demais classes são expostas no Apêndice C.

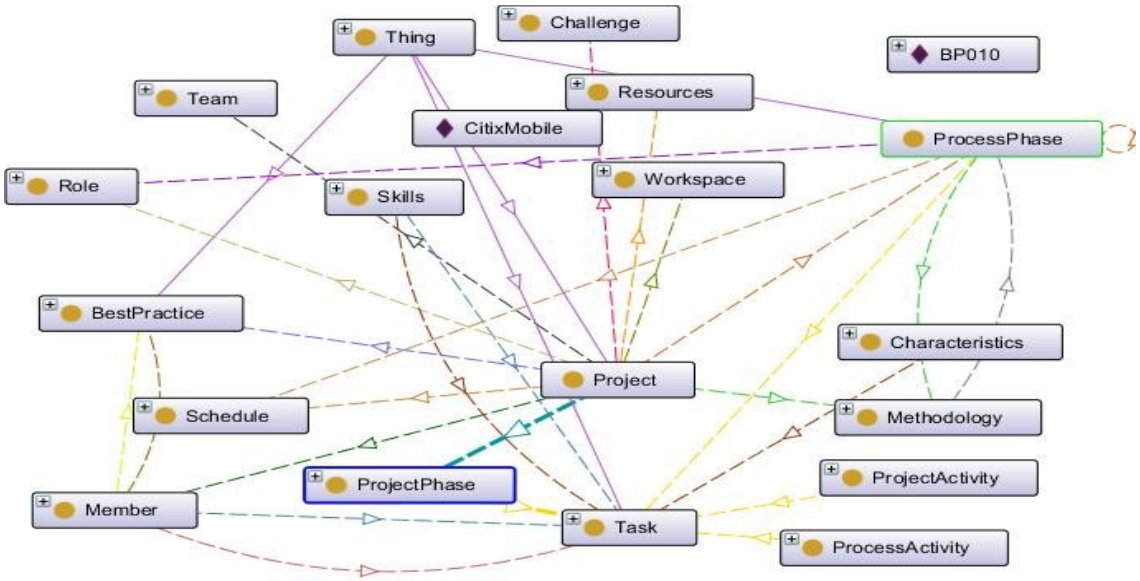


Figura. 4.2. Principais Classes e Relacionamentos da DKDOnto.

Parte das classes da DKDonto foram criadas com base em outras classes, principalmente encontradas em outras ontologias. Nenhuma classe teve reuso direto, ou seja, reutilizar a classe já existente de outra ontologia, porém, algumas foram criadas com muita semelhança. As classes também foram criadas com base nos conceitos identificados a partir dos recursos encontrados no mapeamento sistemático, no caso, baseado nas ferramentas, boas práticas, técnicas, modelos e desafios.

Essa Ontologia utiliza duas classes fundamentais, Challenge e BestPractice para o sucesso de sua proposta as quais são responsáveis pelo armazenamento de todas as informações sobre os problemas e soluções que aparecem no decorrer do projeto. Dessa forma, as consultas dos usuários permitem a visualização das respostas dos desafios, pois a base de conhecimento vai retornar as melhores práticas encontradas para a mesma configuração de projeto e pode ser aplicada para auxiliar

nos desafios encontrados e são também utilizadas por outros times envolvidos com o mesmo projeto ou ainda outros times de diferentes projetos.

Outro importante conceito sobre ontologias são os axiomas, importantes para descrever os relacionamentos através dos conceitos do domínio. A DKDOnto utiliza um conjunto de axiomas para garantir que as regras foram atendidas para o domínio de conhecimento. Os axiomas da DKDOnto foram desenvolvidos inicialmente na disciplina de web semântica do programa de pós-graduação deste centro de informática, cursada em 2011, sob a orientação do professor da disciplina e em conjunto com outro aluno de doutorado.

Para uma melhor contextualização, alguns axiomas são apresentados abaixo, na Figura 4.3.

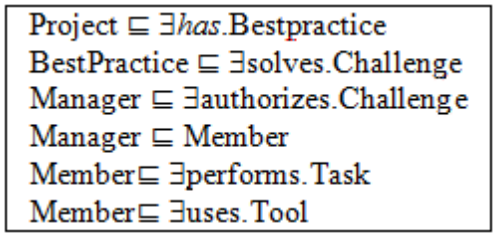


Figura 4.3. Alguns Axiomas da DKDOnto.

Todos os principais conceitos usados nos axiomas supracitados são referentes às classes fundamentais da DKDOnto. O primeiro significa que um projeto (Project) possui boas práticas (BestPractice), o segundo significa que as boas práticas (BestPractice) podem resolver os desafios (Challenge). O gerente (Manager) autoriza ou não o relato de desafios que são relatados por outros membros (Member), sendo também um membro, que executa as atividades (Task) e usa ferramentas (Tool).

Foi verificado que a criação de um quantificador existencial nos axiomas gerados indica que a lógica de descrição foi aplicada. O código OWL utilizado para representar o primeiro axioma é apresentado abaixo, o que significa que as duas classes (Project e BestPractice) possuem relacionamento e isso é uma regra de funcionamento.

```
<owl:Class rdf:about="#?project">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="has"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:about="#?bestpractice"/>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

4.3. DKDs: Ferramenta de acesso a Ontologia

A DKDs é a ferramenta utilizada para criação e manipulação das informações na base de dados, relatando e consultando as tecnologias envolvidas, o funcionamento geral e suas principais caracterísicas. Desenvolvido para auxiliar na transmissão, geração e distribuição de conhecimento da ontologia proposta, é uma ferramenta de apoio à tomada de decisões em projetos no DDS, com base

nos recursos e informações do contexto do projeto, o sistema sugere aos usuários as possíveis soluções para os problemas encontrados. Dessa forma, esse sistema acessa a base de dados com as experiências de projetos distribuídos, configurações desses projetos e desafios encontrados juntamente com as soluções utilizadas.

Para maiores informações e acesso a ferramenta, esta será disponibilizada no endereço www.rgcrocha.com/dkds onde terá o material para acesso a ferramenta bem como informações necessárias para utilizá-la. São apresentadas no Apêndice C algumas imagens como hierarquia de classes, relacionamento de classes e diversos axiomas.

Essa ferramenta tem como objetivo principal apoiar todo o processo de desenvolvimento de software distribuído, oferecendo recomendações com base na configuração do projeto e nas experiências técnicas, não técnicas e organizacionais. É através dela que os usuários da ontologia podem obter importantes informações a respeito do projeto.

Para o desenvolvimento do DKDs foi adotado J2EE (2013) como plataforma geral, os frameworks para aplicação web Grails (2013) e GWT (2013), para persistência o Hibernate (2013) e para manipulação da ontologia foi utilizado o framework OWL API (2013), responsável também pela construção e manipulação de gráficos RDF.

Entre as principais características do DKDs é possível destacar as mais importantes: utilizar o motor de inferência Pellet para dedução de fatos a partir das informações armazenadas anteriormente na base de conhecimento, gerar relatório com os possíveis erros que devem ser evitados no início do projeto, apontar quem são os membros mais indicados para resolverem problemas técnicos, sugerir as possíveis práticas ou técnicas para evitar os desafios que forem consultados.

A máquina de inferência faz deduções em cima do código da ontologia, no caso o OWL DL, quando esta possui suas regras de relacionamento descritas através de axiomas. Dessa forma, alguns dos axiomas a seguir garantem que as deduções sejam feitas de forma correta. Esse primeiro exemplo abaixo diz repeito ao código DL, da Classe Resource que ajuda a solucionar os desafios e auxiliam o membro com seus problemas.

$$resources \sqsubseteq (\exists helps.solveChallenge) \sqcap (\exists assists.member)$$

Já o segundo exemplo diz respeito aos pápeis executados no processo de desenvolvimento, onde papéis fazem parte do time e requerem habilidades específicas.

$$role \sqsubseteq (\exists composes.team \sqcap \exists requires.skill)$$

Nesse terceiro exemplo, é uma característica do membro, onde todo membro possui habilidades, dessa forma, essas habilidades ajudam a resolver possíveis problemas.

$$skill \sqsubseteq \exists helps.solvechallenge$$

Um outro exemplo é na classe methodology, onde a mesma tem relação direta com a classe de desafios e de fases de desenovlvimento. Nesse caso, a metodologia pode ajudar a resolver problemas e esta, possui fase também.

$$methodology \sqsubseteq (\exists helps.solveChallenge) \sqcap (\exists has.phase)$$

Outro caso interessante é o apresentado abaixo, referente as atividades. Atividades fazem parte das fases de desenvolvimento, após a execução de cada atividade, partes de artefatos ou artefatos são

$$task \sqsubseteq \neg phase \sqcap (\exists generate.artifact \sqcap \exists composes.phase)$$

gerados e fazem parte de uma fase também.

Um axioma um pouco mais complexo e que envolve mais classes e definições é o do exemplo abaixo, que envolve o gerente, que é subclasse de membro. Neste caso, o gerente autoriza as boas práticas e desafios que são sugeridos por outros usuários a virar boas práticas e desafios de fato no sistema. Assim como eles também podem sugerir estas para o sistema.

$$manager \sqsubseteq (\exists authorizes.(bestPractice \sqcap challenge \sqcap bestPractice)) \sqcap (\exists suggests.(bestPractice \sqcup challenge))$$

Essas regras gerais de definições, relacionamentos, classes e suas instâncias consequentemente, permitem que o pellet deduza novos fatos a partir de outros. Para melhor entendimento do funcionamento geral dessa aplicação, é necessário entender que ela é composta basicamente por três módulos:

- Módulo de Inferência: Permite a dedução precisa de informação sobre a DKDOnto em código RDF e OWL, utilizando o motor de inferência Pellet.
- Módulo de Consulta: módulo onde ocorrem todas as consultas realizadas pelos usuários. Como já mencionado, as consultas são realizadas através da linguagem SPARQL e são transparentes para o usuário.
- Módulo de Visualização: dar acesso aos relatórios estabelecidos de acordo com as necessidades dos usuários.
- Módulo de Gerenciamento: módulo que é responsável pelo acesso à ontologia, permitindo a inserção, remoção e edição dados nessa Ontologia.

Na Figura 4.4 é possível visualizar o funcionamento geral da ferramenta, onde os usuários têm a interface de acesso para realizar as funções mencionadas anteriormente, enquanto do outro lado, existe a inferência SPARQL (2013) do motor para consultar a DKDOnto, e o componente de API (OWL-API) que vai interagir com ambos contextos.

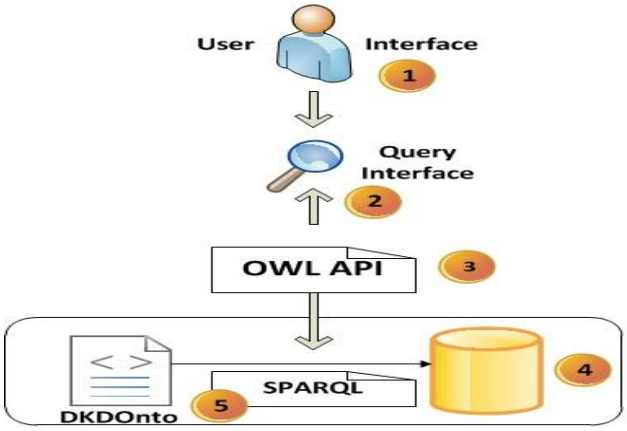


Figura 4.4. Funcionamento Geral da Ferramenta.

Uma importante funcionalidade do sistema é fornecer, por exemplo, opções de que práticas devem ser aplicadas quando o projeto possui determinadas características, facilitando a gerência ou membros da equipe sobre o que pode ser feito para aprimorar a realização das atividades.

A funcionalidade cuja importância e grau de similaridade é a mesma da funcionalidade supracitada é a de que o projeto receba uma lista dos possíveis problemas que podem aparecer. Essa lista será gerada através do conhecimento que a ferramenta terá sobre as características do projeto com

base no conhecimento já adquirido de outros projetos. Exemplos dessas duas telas são apresentados na Figura 4.5.

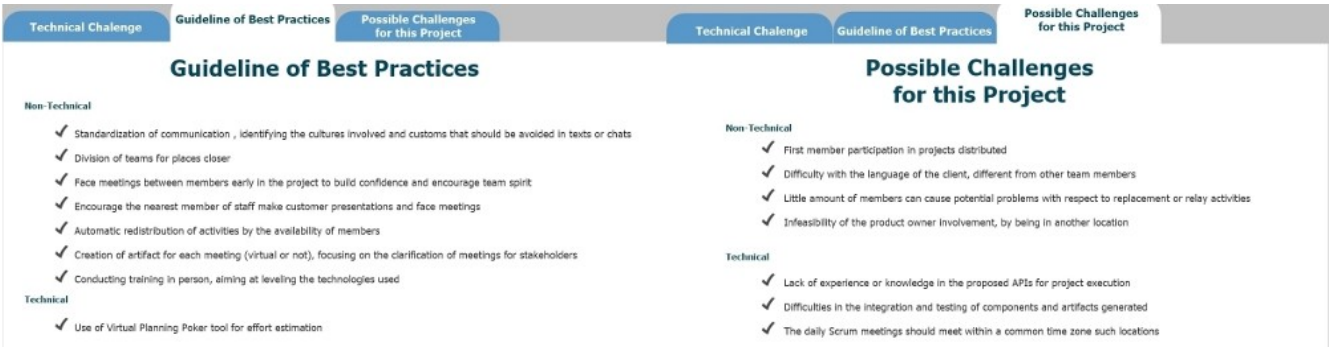


Figura 4.5 – Telas de Boas Práticas e dos Possíveis Problemas para o Projeto.

Na Figura 4.6 é possível visualizar a tela de busca por membros de equipes que são mais experientes para os problemas técnicos que foram pesquisados. Dessa maneira, existe um campo onde o usuário pode digitar qual o problema encontrado e o sistema retornará como resposta, os usuários que em seus perfis possuem os mesmos conhecimentos referente aos problemas informados.

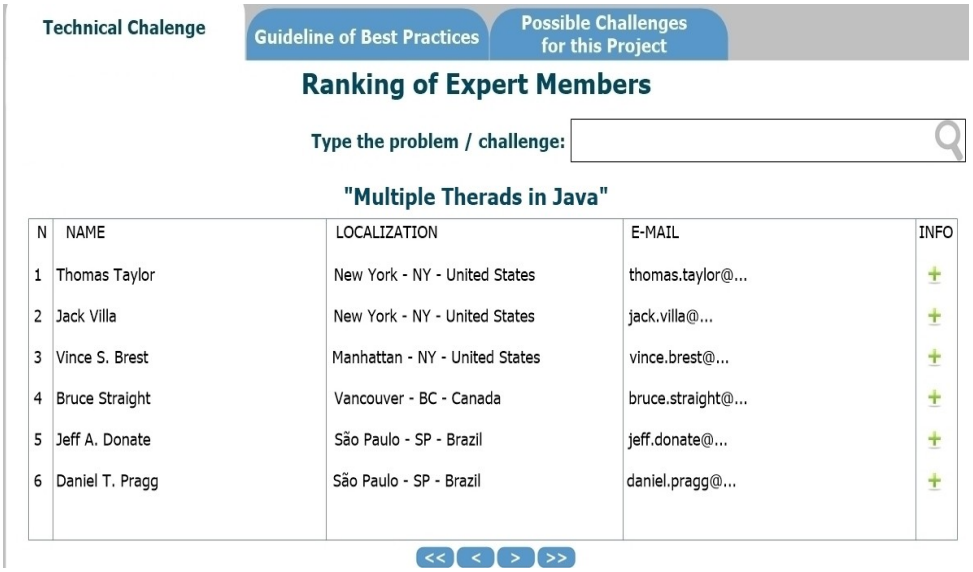


Figura 4.6 – Tela para Indicação de Membros para os Problemas Técnicos.

4.4 DKD-CBRs

Nesta seção são detalhadas as etapas do desenvolvimento da solução adotada para este módulo que utiliza o RBC para recomendação de casos mais similares aos desafios encontrados. São apresentadas então, a descrição da abordagem, a arquitetura aplicada e a implementação.

4.4.1 Descrição da Ferramenta

A ferramenta adotada neste módulo consiste em um sistema de RBC integrado a um componente de PLN para extração de características e recomendação de soluções para problemas de equipes de DDS. A intenção da aplicação proposta e descrita nesta seção consiste em apoiar as resoluções de novos problemas oriundos de uma situação em um novo projeto.

O fluxo de funcionamento da aplicação, ilustrado na Figura 4.10, ocorre basicamente em 4 etapas, sendo elas:

- No primeiro momento, a descrição do novo problema é inserida na interface de interação entre o usuário e o sistema em linguagem natural através de texto;
- Nessa segunda etapa, a descrição do novo problema em DDS, em linguagem natural através de texto tem suas características textuais extraídas pelas técnicas empregadas no módulo de PLN da solução;
- Nesse momento em que as características extraídas na segunda etapa são utilizadas pelas técnicas empregadas no módulo de RBC, como forma de representação do caso e o cálculo de similaridade com os casos da base de casos é realizado para assim recuperar casos que tenham características similares às descritas no primeiro momento, com a finalidade de recomendar soluções que sirvam de apoio ao usuário na resolução desse novo problema;
- O último passo na aplicação é a realização da revisão e retenção dessa nova experiência (caracterizada como novo caso), aumentando cada vez mais a base de casos da solução. O que proporciona um sistema atualizado e dinâmico.

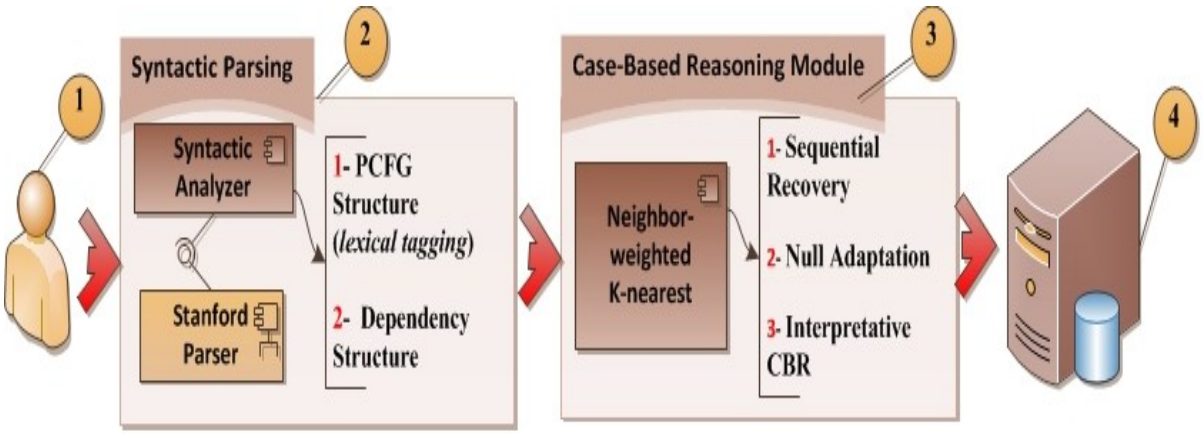


Figura 4.7. –Fluxo de Funcionamento da Ferramenta RBC.

A arquitetura conceitual do sistema desenvolvida é composta por dois módulos identificadas anteriormente no fluxo de funcionamento da Figura 4.6, o módulo de *Parser de* Linguagem Natural e o módulo Raciocínio Baseado em Casos. Estes são descritos nas subseções posteriores.

4.4.1.1 Módulo do *Parser* de Linguagem Natural

Neste módulo, como reconhecedor das regras gramaticais do PLN para a análise sintática, foi utilizado um método estatístico e a gramática escolhida foi a *Probabilistic Context Free Grammar* [Chi, 1999]. O inglês foi escolhido como idioma de interação entre o usuário e o sistema em linguagem natural por ser o principal idioma utilizado na comunidade acadêmica e com o intuito de que a ferramenta possa ser utilizada pela tanto pela indústria nacional, como também pela internacional. As três principais atividades realizadas pelo módulo de processamento de linguagem natural estão descritas a seguir.

A atividade de *Scanning* ou *Tagging*, é a realização de análise léxica propriamente dita. Momento este no qual as palavras são identificadas (separadas) no texto, assim como suas categorias gramaticais (considerado como *tag*).

Para uma melhor contextualização, é possível utilizar um exemplo de *tagging* para o seguinte caso: “S*imulate the Daily Scrum Meeting in distributed project using adaptation of the SCRUM”*. Sendo a identificação (separação) dessas palavras realizadas, conforme exemplo abaixo. Algumas das principais *Tags* utilizadas e seus significados podem ser visualizadas na Tabela 4.1.

- “*Simulate*/**VB** *the*/**DT** *Daily*/**NNP** Scrum/**NNP** *Meeting*/**VBG** *in*/**IN** *distributed*/**VBN** *project*/**NN** *using*/**VBG** *adaptation*/**NN** *of*/**IN** *the*/**DT** SCRUM/**NNP**.

Tabela 4.1 – Algumas *Tags* e Significados (BIES *et al.*, 1995).

Tag	Meaning
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
NN	Noun, singular or mass
NNS	Noun, plural
RB	Adverb
RBS	Adverb, superlative
VB	Verb, base form
VBD	Verb, past tense
VBN	Verb, past participle

Já a atividade de *parsing* tem por objetivo realizar a análise sintática, em que o texto será totalmente analisado seguindo as regras sintáticas definidas pela gramática escolhida e sua árvore gramatical de derivação será então obtida, como já citado no capítulo de referencial teórico. Para uma melhor caracterização, um exemplo pode ser visualizado na Figura 4.8 utilizando o mesmo texto anterior.

```
(ROOT
  (S
    (VP (VB Simulate)
      (NP
        (NP (DT the) (NNP Daily) (NNP Scrum))
        (VP (VBG Meeting)
          (PP (IN in)
            (NP
              (NP (VBN distributed) (NN project))
              (VP (VBG using)
                (NP
                  (NP (NN adaptation))
                  (PP (IN of)
                    (NP (DT the) (NNP SCRUM))))))))))
    (. .)))
```

Figura 4.8 – Árvore Gramatical do Exemplo.

Como última atividade do módulo de *Parser* de Linguagem Natural, a atividade de dependências tem como objetivo fazer a identificação das dependências existentes entre as palavras dentro do texto e os seus tipos.

Para contextualizar, é possível utilizar o mesmo texto utilizado nos dois últimos exemplos, “S*imulate the Daily Scrum Meeting in distributed project using adaptation of the SCRUM”*, destacando algumas das dependências identificadas:

- det(Scrum-4, *the*-2), nn(Scrum-4, *Daily*-3), dobj(*Simulate*-1, Scrum-4), partmod(Scrum-4, *Meeting*-5), prep(*Meeting*-5, *in*-6), amod(*project*-8, *distributed*-7), pobj(*in*-6, *project*-8), partmod(*project*-8, *using*-9).

Dessa forma, a dependência é colocada antes dos parênteses, e dentro deles, são colocadas as palavras e alguns números os quais determinam a posição daquela palavra na sentença. Algumas das principais dependências utilizadas e seus significados podem ser visualizados na Tabela 4.2.

Tabela 4.2 – Algumas Dependências e Significados. Fonte: Marneffe e Manning (2008).

Dependência	Significado
Abbrev	Abbreviation modifier
Acomp	Adjectival complement
Advmod	Adverbial modifier
Amod	Adjectival modifier
Conj	Conjunct
Csubj	Clausal subject
Nsubj	Nominal subject
Nn	Noun compound modifier
Nsubjpass	Passive nominal subject
partmod	Participial modifier
Pcomp	Prepositional complement
Possessive	Possessive modifier
Prepc	Prepositional clausal modifier
Xsubj	Controlling subject

4.4.1.2 Módulo de Raciocínio Baseado em Casos

De forma simplificada, para desenvolver um sistema de RBC, é necessário realizar a modelagem dos casos, o que consiste em identificar como representar o conhecimento, desenvolver um mecanismo de recuperação, um mecanismo de adaptação e um mecanismo de aprendizagem.

Neste contexto, para a representação dos casos foi elaborada uma modelagem de dados de forma a capturar as características extraídas para futura utilização, tais como palavras, classes gramaticais e dependências entre palavras dentro do texto, do mesmo modo como os textos completos são os dados que compõem cada caso em um banco de dados relacional modelado. Detalhes da representação dos dados modelada para este trabalho podem ser vistos no Apêndice D.

Para a fase de recuperação do RBC, a abordagem de recuperação utilizada foi à recuperação sequencial [Wangenheim, 2003] que é uma etapa que tem a finalidade de encontrar casos similares na base de casos que possam ser facilmente adaptados para o problema atual, objetivando sua solução.

Na fase de adaptação/reutilização, a abordagem escolhida foi a adaptação nula [Wangenheim, 2003], fase que compreende a ação de adaptar automaticamente o caso recuperado para o problema atual. Logo, o tipo de abordagem escolhida para a realização do RBC foi a interpretativa, aquela em que o engenheiro do conhecimento (usuário do sistema) recebe casos similares recomendados pelo sistema e realiza a adaptação das soluções dos casos recomendados com base em seu conhecimento profissional.

A abordagem utilizada na fase de reparação e revisão da solução proposta com base na similaridade entre o problema atual e o caso recuperado foi o próprio engenheiro do conhecimento realizar a ação, assim como na fase de adaptação, esta fase avalia a solução gerada pelo reuso.

Por fim, na última fase do RBC, na fase de retenção é realizado em uma base de dados relacional criada e modelada conforme a representação dos casos. A retenção (armazenamento) será realizada após o usuário avaliar, reparar e revisar a solução para o novo problema.

Com a finalidade de ter uma melhor organização, manutenção e possível futura evolução, a ferramenta foi definida em camadas seguindo o padrão MVC (*Model-view-controller, ou em português*, Modelo-visão-controlador), essa arquitetura possui três componentes distintos [Krasner; Pope, 1988]:

- *Model*: camada que consiste na modelagem dos dados da aplicação, manipulação e geração.
- *View*: camada onde a informação é exibida sem a preocupação em como a informação foi obtida.
- *Controller:* camada que determina o fluxo da apresentação servindo como uma camada intermediária entre a camada de apresentação e a lógica, realizando também as regras de negócio.

Na camada *Controller* ficam os pacotes *Reasoning* e PLN, pacotes estes que contém as classes que realizam as operações do modelo proposto para este trabalho.

No pacote *Reasoning* são realizadas todas as atividades relativas ao raciocínio baseado em casos e no pacote PLN são realizadas as operações de extração de características da linguagem natural. Mais detalhes de especificações como diagrama de classes, organização dos pacotes do sistema nas camadas do MVC e modelagem da base de dados podem ser vistos no Apêndice D.

Para o desenvolvimento desta módulo, diversas tecnologias foram adotadas e em seguida são apresentadas as ferramentas e técnicas utilizadas. Por questões de praticidade e contexto de desenvolvimento, essa módulo não utilizou todas as tecnologias utilizadas no módulo de acesso a DKDOnto, por exemplo, o Grails. A linguagem de programação Java (Java Oracle, 2013) foi escolhida e utilizada na implementação da solução proposta, assim como as demais ferramentas e tecnologias:

- NetBeans IDE 7.1.2 (NetBeans IDE – Oracle, 2013), como ambiente de desenvolvimento;
- The Stanford Parser 2.0.5 (The Stanford Parser: A statistical parser, 2013), ou mais conhecida como API Stanford (conjunto de padrões e rotinas estabelecidos que abstraem critérios de funcionalidade e focam no propósito de seu serviço) como biblioteca para auxiliar no módulo de Processamento de Linguagem Natural;
- PostgreSQL 9.0 (PostgreSQL, 2013) e pgAdmin III (pgAdmin PostgreSQL Tools, 2013), para persistência na base de dados relacional, criação e manutenção;
- Mozilla Firefox 23, como browser para testes principal, Google Chrome 28 e Internet Explorer 10, como browsers de teste adicional;
- API Java Servlet[3] e Java Server Pages[4], como tecnologias Java para desenvolvimento web;

[3] http://www.oracle.com/technetwork/java/index-jsp-135475.html

[4] http://www.oracle.com/technetwork/java/javaee/jsp/index.html

- Apache Tomcat 7.0.22.0 (Apache Software Foundation, 2013), como servidor de aplicação;
- JPA 2.0 (Java Persistence API, 2013), como *framework* de mapeamento objeto-relacional;
- JQuery 1.10.2 (The jQuery Foundation, 2013), como biblioteca JavaScript.

Nas subseções posteriores são descritas as técnicas utilizadas no desenvolvimento dos módulos de extração de características de texto, recomendação de casos com base em similaridades e retenção de dados.

Neste módulo utilizou-se uma técnica de PLN na qual possui o objetivo de extrair características do texto de descrição do novo caso recebido. Para auxiliar nessa ação, foi utilizada a API Stanford como citada anteriormente.

Essa API é uma implementação Java de *parsers* (software que realiza análise sintática) de linguagem natural probabilístico, como *parsers* que implementam uma PCFG (*Probabilistic Context Free Grammar*) otimizados, parsers de dependência léxica e parsers PCFG léxico. O *parser* implementa um modelo de produto consignado, que separa a estrutura da frase em PCFG e em dependências léxicas, cujas preferências são combinadas por um eficiente método de inferência, utilizando um algoritmo A*.

Um *parser* de linguagem natural é um programa que determina a estrutura gramatical de sentenças, tal como, quais grupos de palavras formam frases e quais palavras são sujeito ou objeto de um verbo. *Parsers* probabilísticos usam conhecimento da linguagem adquirido de sentenças passadas para tentar produzir a melhor possível análise de novas sentenças.

Os *parsers* estatísticos ainda apresentam alguns pequenos erros, mas funcionam bem em muitos casos. Seu desenvolvimento foi um dos maiores marcos no processamento de linguagem natural nos anos 90 [The Stanford Parser: A statistical parser, 2013].

Na Figura 4.9 é possível observar um exemplo de saída do módulo desenvolvido, o texto que teve suas características extraídas foi: “*Process of collaborative software development for geographically distributed teams*”. É possível ver as classes gramaticais das palavras, a árvore gramatical, as dependências entre palavras e seus tipos.

Como exemplo para extração de classes gramaticais das palavras, é possível visualizar na Figura 4.9 que as palavras *process*, *software* e *development* foram definidos como NN (*Nominal Noun*, ou, Substantivo Nominal) e *distributed* como VBN (*Nominal Verb*, *past participle*, ou, Verbo Norminal, no particípio passado).

Na extração das dependências entre as palavras, pode ser visualizado que foi identificada uma relação de dependência do tipo prep_of (preposição) entre as palavras *process* e *development*, e uma do tipo amod (adjetivo modificador) entre as palavras *development* e *collaborative*.

```
Classes gramaticais das palavras:
process/NN
of/IN
collaborative/JJ
software/NN
development/NN
for/IN
geographically/JJ
distributed/VBN
teams/NNS


Árvore Gramatical do Texto:
(ROOT
  (FRAG
    (NP
      (NP (NN process))
      (PP (IN of)
        (NP (JJ collaborative) (NN software) (NN development))))
    (PP (IN for)
      (NP (JJ geographically) (VBN distributed) (NNS teams)))))


Dependências entre palavras e seus tipos:
root(ROOT-0, process-1)
amod(development-5, collaborative-3)
nn(development-5, software-4)
prep_of(process-1, development-5)
amod(teams-9, geographically-7)
amod(teams-9, distributed-8)
prep_for(process-1, teams-9)
```

Figura 4.9 - Extração de Características do Texto de Exemplo.

A recomendação de casos passados que possam ser utilizados pelos usuários para os ajudar na resolução de um novo problema é uma das atividades principais deste trabalho. Ela é realizada com base em um cálculo de similaridade feito entre as características textuais extraídas pelo módulo de PLN do novo problema descrito pelos usuários e as experiências anteriores, já processadas e com suas características textuais já modeladas como casos, contidas na base de casos. Este cálculo de similaridade é a atividade principal ocorrida na fase de recuperação do RBC.

O cálculo da similaridade do RBC é dividido em: Similaridade Global, que representa a similaridade no nível dos casos e a Local, que representa a similaridade no nível de atributos. Para o cálculo da Similaridade Global a abordagem foi decidido utilizar uma abordagem que fosse bastante difundida e já validada, dessa forma, utilizada foi a *Nearest Neighbor* [Cover e Hart, 1967], que é bastante utilizada por outros trabalhos. E nesse caso, ocorrências em uma base de casos são vistas como pontos em um espaço multidimensional, a distância espacial entre as representações dos casos mostra a similaridade entre eles e a busca reduz-se à determinação do vizinho geometricamente mais próximo [Wangenheim, 2003].

Mais precisamente foi utilizada a *Nearest Neighbor Ponderada* [Cover e Hart, 1967], que pode ser observada na Figura 4.13. Nela foram utilizados para representação dois casos C1 e C2, com atributos a_1, a_2, a_3...a_p arbitrários.

A abordagem faz o cálculo das similaridades locais para cada atributo dos casos, os multiplica pelos seus pesos, realiza o somatório representado na Figura 4.9, encontrando assim o valor de similaridade. Logo depois, realiza a divisão pela soma de todos os pesos, o que caracteriza a normalização da média ponderada e o real valor de similaridade entre os casos. Este valor varia entre 1 e 0, onde 1 é a similaridade total (100%) e 0 é a ausência total de similaridade (0%).

A função sim_j representa a medida de similaridade entre o C1 e C2 para o atributo j, w_j indica o peso (ou importância) do atributo j para a determinação da similaridade, se o valor de w_j for 0 o atributo não é considerado para o cálculo da similaridade.

$$SIM(C1,C2) = \sum_{j=1}^{p} w_j sim_j(C1,C2)$$

Figura 4.10 - *Nearest Neighbor* Ponderada. (Wangenheim, 2013)

Os cálculos das similaridades locais devem ser definidos de acordo com o tipo do atributo específico e no contexto da aplicação em questão. Logo, ele foi feito seguindo os critérios escolhidos para esta solução e não são de definição absoluta, podendo ser melhorados, retirados ou novos cálculos podem ser adicionados, no futuro.

Na ferramenta, como mencionado no parágrafo anterior, foram definidos quatro atributos, o que no geral é uma prática comum para sistemas RBC, são eles:

- Atributo 0: Calcula o número de palavras iguais entre os dois casos;
- Atributo 1: Calcula a quantidade de palavras, se essa quantidade entre os dois casos é igual;
- Atributo 2: Calcula o número de verbos iguais entre os dois casos;
- Atributo 3: Calcula a quantidade de palavras iguais que causam dependência na descrição dos casos.

```
Similaridade Atributo 0: 0.42857142857142855
Similaridade Atributo 1: 0.6428571428571429
Similaridade Atributo 2: 0.25
Similaridade Atributo 3: 0.2727272727272727
Similaridade Global: 0.3998917748917749

Similaridade Atributo 0: 0.15384615384615385
Similaridade Atributo 1: 0.6923076923076923
Similaridade Atributo 2: 0.5
Similaridade Atributo 3: 0.0
Similaridade Global: 0.2670940170940171

Similaridade Atributo 0: 0.18181818181818182
Similaridade Atributo 1: 0.8181818181818182
Similaridade Atributo 2: 0.25
Similaridade Atributo 3: 0.0
Similaridade Global: 0.25883838383838387

Similaridade Atributo 0: 0.07142857142857142
Similaridade Atributo 1: 0.32142857142857145
Similaridade Atributo 2: 0.0
Similaridade Atributo 3: 0.0
Similaridade Global: 0.08531746031746032
```

Figura 4.11 - Valores do Cálculo de Similaridade entre Novo Caso e Casos da Base.

É possível observar na Figura 4.10 um exemplo de saída do módulo desenvolvido, a imagem mostra os valores encontrados no cálculo de similaridade entre um novo caso descrito como "*Process of collaborative software development for geographically distributed teams*" e alguns casos da base de casos. Na tabela, se pode observar valores de similaridade para os atributos locais e o valor final de similaridade global, todos variando de 0 a 1, onde zero é a não similaridade total e 1 é a similaridade total.

Na Figura 4.10 observa-se que a similaridade local na primeira comparação para o atributo 0 foi de 0.42 (42%), para ao atributo 1 foi 0.64 (64%), para o atributo 2 foi 0.25 (25%) e para o atributo 3 foi 0.27 (27%). Após o cálculo de similaridade global foi definido aproximadamente 0.40 (40%) de semelhança entre os dois casos, os pesos utilizados foram: 8 para o atributo 0, 3 para o atributo 1, 3 para o atributo 2 e 4 para o atributo 3, com valores variando de 1 a 10.

Outra atividade importante deste módulo é a atividade de retenção de dados. A operação nada mais é do que o ato da inserção dos dados na base de dados, conhecida como base de casos em RBC.

Os dados juntos formam as informações do novo caso a ser retido, e este, posteriormente, é inserido como um novo caso na base de casos do sistema. Esta atividade compreende a fase de retenção do RBC, por isso o nome escolhido. Ao contrário das outras atividades descritas anteriormente, está é uma atividade relativamente simples, segue os critérios básicos de inserção utilizando uma *framework* de persistência, o ORM (*Object-relational mapping*, 2013) como JPA.

4.4.1.3 Operação do Lado do Usuário

Nas seções seguintes serão abordadas as principais funcionalidades da ferramenta ilustradas através da utilização da ferramenta em execução sob um servidor local *Apache Tomcat*. São elas, a retenção de um novo caso, a busca de casos similares e a escolha de solução proposta e revisão.

Assim como a atividade inicial mais básica, a retenção de novos casos é uma atividade importante em todo o processo de utilização da solução. A aba ***New Case*** do sistema apresenta o formulário a ser preenchido pelo usuário com as informações do novo caso.

Esse formulário possui os campos ***Description***, ***Adopted Solution*** e ***Effectiveness***, campos de texto que possibilitam o uso de linguagem natural escrita. Desse modo, após os dados serem submetidos ao servidor, ele realiza a extração das características de linguagem natural contidas nos textos, efetua a inclusão do novo caso na base de casos e retorna a mensagem de sucesso para o usuário. Na Figura 4.11 é possível visualizar a tela em uma exemplificação do formulário de retenção de novo caso preenchido.

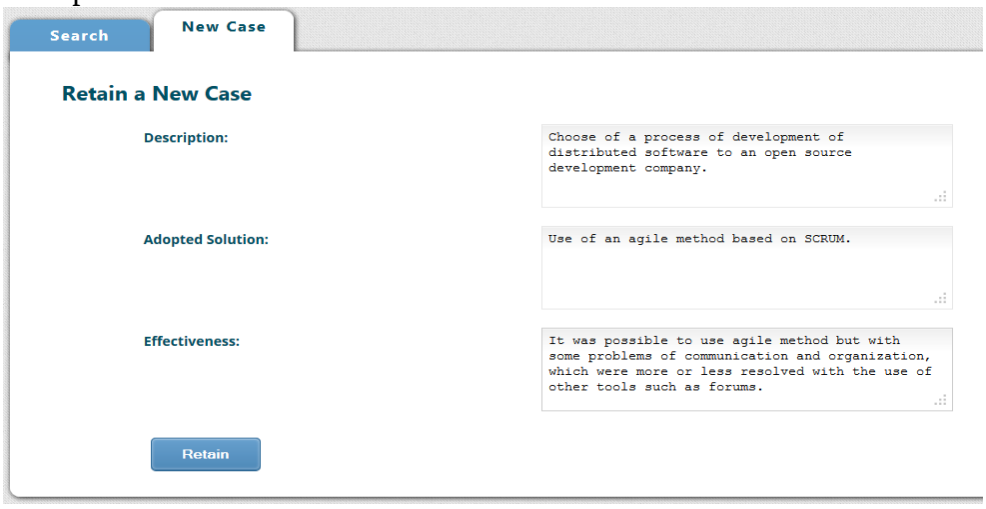


Figura 4.12 - Tela de Retenção de Novo Caso.

Podendo ser considerada como a atividade mais importante da solução, a busca de casos similares é uma atividade que merece ser bem detalhada. Na tela inicial da solução a aba ***Search*** já vem selecionada e é nela que fica o formulário de informações para busca.

A descrição em texto do novo problema em DDS deve ser descrita na língua inglesa. Esta descrição posteriormente terá suas características extraídas pelo processamento de linguagem natural da solução e será comparada com as características da descrição dos casos existentes na base de casos, conforme as métricas definidas, para o cálculo de similaridade entre eles. Um exemplo de preenchimento é visto na Figura 4.12.

Figura 4.13 - Tela de Busca por Casos Similares da Solução.

A solução oferece a possibilidade da busca apenas pelo caso mais similar e a busca de todos os casos similares conforme um mínimo de similaridade escolhido entre uma variação de 0% a 100% de similaridade como filtro para o retorno de casos, conforme visto em um filtro de busca chamado *Similarity Minimum* na Figura 4.13. Na busca de apenas o caso mais similar, o sistema suprime a possibilidade de escolha de mínimo de similaridade relativo a outra funcionalidade.

Para um maior nível de refinamento ou especificidade, a ferramenta permite também a definição dos pesos de cada métrica utilizada no cálculo de similaridade da fase de recuperação do RBC, trazidos do conceito utilizado na própria aplicação. A escolha de valor de peso varia entre 1 e 10 para cada métrica de similaridade. Por padrão, o sistema define as métricas de número de palavras e número de verbos iguais como peso 3, o número de palavras dependentes similares como peso 4 e o número de palavras iguais como peso 8, estas escolhas foram feitas conforme uma medida de importância padrão identificada nas métricas.

Para uma maior contextualização, na Figura 4.14 é apresentado um exemplo do resultado de uma busca realizada pelo sistema. O sistema apresenta os casos em lista, todas as informações relevantes aos casos encontrados, como descrição, solução adotada, efetividade da solução, data de inclusão e porcentagem de similaridade de cada caso com o novo problema, disposta em ordem decrescente ao grau de similaridade, do caso mais similar ao menos similar.

Figura 4.14 - Resultado da Busca Exemplo da Solução.

A atividade de escolha de uma solução proposta para sua posterior revisão pode ser considerada como uma continuação da atividade anterior a de busca de casos similares, dessa forma, optou-se pela abordagem de adaptação nula na fase do RBC posterior a recuperação, ou seja, logo após o sistema apresentar a listagem de casos similares, o usuário, caracterizado com o engenheiro do conhecimento, escolherá aquela que apresentar descrição, solução e efetividade mais apropriada para serem revisadas para o seu problema, e assim, interpretará e aplicará adaptação em seu problema real, e somente depois avaliará sua efetividade.

Com estas informações em mãos, a próxima atividade será a revisão da solução e efetividade do caso para o novo problema e, por fim, a retenção desse novo caso. Como já mencionado, o sistema permite a escolha de um caso para posterior interpretação e revisão para o novo problema. O sistema apresenta o novo problema descrito, a descrição do caso semelhante recuperado da base de casos, a solução adotada nesse caso recuperado, a efetividade dessa solução adotada e o grau de similaridade entre o novo problema e o caso recuperado. Assim, o engenheiro do conhecimento (usuário do sistema) editará a solução para o novo problema, tal como a avaliação de efetividade e realizará a operação de retenção desse novo caso ocorrido, conforme ilustra a Figura 4.15.

Figura 4.15 - Revisão do Caso Escolhido na Lista Recuperada.

4.4.2 Execução de Sentenças para Testar a Efetividade da Ferramenta

Alguns testes foram realizados tentando observar os possíveis comportamentos do sistema, verificando apenas a sua credibilidade, por exemplo, alguns casos utilizados se basearam na descrição do novo problema (sentença a ser processada), os subtextos das descrições dos casos citados anteriormente tinham como finalidade a recomendação pela ferramenta exatamente dos casos os quais as descrições dos novos problemas foram retirados.

Por fim, outros testes utilizaram como descrição do novo problema textos completos de casos contidos na base de casos, com a finalidade de que os casos recomendados fossem os mesmos com 100% de similaridade.

Tabela 4.3. Testes realizados para validar ferramenta

Sentença Processada	**Mais Similar**
During the execution of a project, there is a lack in communication between developers, analysts, managers, managers, clients and users.	57.37%
Business processes are not mapped, which incurs a systemic solution that sometimes does not meet.	70.76%
Lack of communication in software development teams separated geographically.	71,43%
In a project with an unknown number of members working in 4 different countries there existed problems with the delegation of tasks.	87.44%
In a project in which nor the number of members nor the cities involved were reported, problems happened due to the addition of new requirements (both functional and non-functional) during the architecture and modeling steps of the project.	100%

Finding an infrastructure to support distributed development projects based on components.	100%
The functionalities were performing actions incorrectly due to noise communication between the user and the body of systemic solution (analysts and developers) working remotely.	100%

Nos testes realizados apresentados nas quatro primeiras linhas, foram utilizadas como descrição do novo problema (sentença a ser processada) partes (literalmente) das descrições dos casos, ou seja, partes de casos já existentes na base eram colocados como problemas. O objetivo final era que a ferramenta recomendasse exatamente os casos dos quais as descrições dos novos problemas foram retirados.

As buscas executadas e apresentadas nessas quatro primeiras linhas demonstram que a recomendação dos casos foi positiva, no sentido de que foram retornados os dados esperados, em que os primeiros casos retornados eram sempre os mesmos dos quais parte do problema foi retirado para que a pesquisa fosse realizada. Dessa forma, foi demonstrado que os casos foram recuperados corretamente e com o grau de similaridade relativamente alto.

Para complementar, foram realizadas buscas que utilizaram como descrição do novo problema textos completos de casos contidos na base de casos com a finalidade de que os casos recomendados fossem os mesmos com 100% de similaridade. Foi escolhido para ser recomendado apenas o caso mais similar e os pesos utilizados nas métricas de similaridade mais uma vez foram o padrão do sistema desenvolvido.

As buscas realizadas são apresentadas nas três últimas linhas da Tabela 4.3 acima, apresentando resultados positivos, no sentido de que foram retornados os dados esperados. Demonstrando que os casos iguais foram recuperados normalmente e com o grau de similaridade de 100%.

4.5 Casos de Teste

Para atender a seguinte questão de pesquisa, a seguinte hipótese foi formulada: “Demonstrar que um método baseado em Ontologias, RBC e PLN é uma solução viável para extração e recomendação de experiências no apoio de decisões e soluções de problemas em projetos de DDS” com intuito de dar um melhor direcionamento aos testes executados.

Como já mencionado no Capítulo 3, esse experimento inicial teve como amostra 102 casos de problemas e soluções extraídos de artigos científicos de conferências da área bem como de um survey realizado com profissionais de desenvolvimento que trabalharam com DDS. A lista com todos os casos pode ser visualizada no Apêndice A.

Para a realização dos testes, foi solicitado a um gerente de projetos da Dataprev – PB com experiência em DDS que descrevesse alguns problemas que já tinha enfrentado como membro de uma equipe distribuída. Ele teve o período de 20/01/2015 até 30/01/2015 (10 dias) para descrever esses casos. Dessa maneira, 12 sentenças foram relatadas e usadas para a realização dos testes da abordagem.

4.5.1 Resultados dos testes

A Tabela 4.4 abaixo apresenta de forma resumida, quais foram os casos utilizados nos testes e qual a porcentagem de similaridade do caso mais parecido. Nesse contexto, as sentenças foram escolhidas baseadas em casos já existentes na base de casos, e dessa maneira, o valor de similaridade

diz respeito apenas ao caso mais similar encontrado, o que não quer dizer que outros casos não tenham sido indicados.

Tabela 4.4. Testes realizados

Sentença Processada	**Mais Similar**
Staff with problem expression of ideas because of the difficulty of understanding of English among distributed teams in Japan and China.	52.86%
W/riting the documentation requirements in a project distributed with national distance and high levels of temporal dispersion between team members.	65.82%
Team divided in two different countries (England and Brazil) of a large company face problem in the definition of the artifacts that will be generated, because in Brazil the custom of the teams was to generate specific artifacts and England the same company and its teams were used to generate other artifacts.	51.13%
Communication problems due to failures on the internet with one of the teams in the state two different states from Brazil	52.56%
Reducing of productivity and decreasing of teamwork because the communication regarding meetings and discussions between project members almost doesn't exist.	57.95%
Problem for defining roles for teams located in different countries.	49.54%
Communication between members of a small team in a distributed software development project.	70.02%
The deadlines were not met in a project with 12 members from 3 different countries because of time wasted with unnecessary tasks.	61.55%
Difficulty in performing the "Daily Scrum" because part of the team works on different locations and works with a difference of 3 hours.	74.72%
In a project with an unknown number of members working in Brazil, communication problems among the teams existed.	93.33%
In a project with a unknown members team working in different cities, a reduction in the number of members happened.	93.2%
The deadlines were not met in a project with unknown members from different countries because of time wasted with unnecessary tasks.	63.54%

Dos 12 (doze) testes realizados, 11 (onze) utilizaram a recomendação de casos similares de forma eficiente e com sucesso, casos que realmente eram semelhantes ao novo problema descrito pelo usuário e tinham soluções úteis para serem adaptadas e utilizadas, formando um total de 91.67% de acerto. Dos 11 (onze) acertos apenas 1 (um) não realizou a melhor recomendação real possível, e assim, dos testes com efetivo sucesso, aproximadamente 88,89% tiveram acerto real e do conjunto total de testes (12), 80% tiveram acerto real. Lembrando que em todos os testes realizados a base de dados completa foi utilizada com 102 casos.

O mínimo de similaridade escolhido dos casos a serem recomendados foi de 30% e os pesos utilizados nas métricas de similaridade foram o padrão do sistema desenvolvido, número de palavras e número de verbos iguais com peso 8 (oito), número de palavras dependentes similares com peso 6 (seis) e número de palavras iguais com peso 2 (dois).

Tabela 4.5 - Resultado do Processamento do Caso 1

Sentença Processada:
Staff with problem expression of ideas because of the difficulty of understanding of English among distributed teams in Japan and China.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **52,86%, 46,81% e 46,72%,** respectivamente. -- Caso 1 -- **Description:** Lack of standardization of artifacts used in the context of the development cycle of geographically distributed teams. **Adopted Solution:** Assess the need for standardization of common software development process between sites. In other words, evaluate the need to reach consensus on the processes and methods used by teams, which are common to all groups. **Effectiveness:** The solution aims to minimize the effort for integration of artifacts and communication between team members. -- Caso 2 – **Description:** In a project with an unknown number of members working in one country, communication problems among the teams existed. **Adopted Solution:** Assign a leader for each team, who are to be responsible for the communication among all teams. **Effectiveness:** -- Caso 3 -- **Description:** There is a need to manage the schedule of activities performed by a team of developers distributed in different cities. **Adopted Solution:** The usage of online and shared tools among team members, for example, google calendar and google docs. Furthermore, conducting online meetings, via Skype or Gtalk, and in-person monthly meetings.

Effectiveness:
The use of tools to assist the monitoring of project activities for all its members was essential. Since the tools are shared online, all team members were able to monitor what activities the members were executing and what tasks should be performed in the future. Nevertheless, monthly meetings were needed to complement the development and clarify any problems regarding the understanding of the project requirements.

Na Tabela 4.5 apresentada acima há um exemplo de resultado para o processamento de uma sentença em que é possível visualizar qual foi a sentença utilizada, qual caso ou quais casos e seus resultados com a respectiva taxa de similaridade. Na tabela também são apresentados para cada caso encontrado, a descrição do caso, a solução adotada e sua respectiva efetividade. Nesse caso, o caso mais similar obteve 52.86% de similaridade, 46.81% o segundo mais similar e por fim, 46.72% o último. Enquanto a Tabela 4.6 abaixo, já trouxe o caso mais similar, com taxa de similaridade mais alta, no caso com 65.82%.

Tabela 4.6 - Resultado do Processamento do Caso 2

Sentença Processada:
W/riting the documentation requirements in a project distributed with national distance and high levels of temporal dispersion between team members.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **65,82%, 51,67% e 51,01%,** respectivamente. -- Caso 1 -- **Description:** Performing documentation and validation of requirements in project with national distance, high levels of temporal and geographical dispersion. **Adopted Solution:** Making use of StoryBoards. **Effectiveness:** Allowed the sharing of understanding of the requirements among distributed teams. Showed decreased amount of duplicate requirements and not corresponding to the modules developed. Occurred mostly use cases ready to move to the design phase. -- Caso 2 -- **Description:** There is a need to manage the schedule of activities performed by a team of developers distributed in different cities. **Adopted Solution:** The usage of online and shared tools among team members, for example, google calendar and google docs. Furthermore, conducting online meetings, via Skype or Gtalk, and in-person monthly meetings. **Effectiveness:** The use of tools to assist the monitoring of project activities for all its members was essential. Since the tools are shared online, all team members were able to monitor what activities the members were executing and what tasks should be performed in the future. Nevertheless, monthly meetings were needed to

complement the development and clarify any problems regarding the understanding of the project requirements.

-- Caso 3 –
Description:
In a project with an unknown number of members working in one country, communication problems among the teams existed.

Adopted Solution:

Assign a leader for each team, who are to be responsible for the communication among all teams.

Effectiveness:

No quesito avaliação dos resultados, para facilitar a classificação, as recomendações foram classificadas em: fraco (0% - 22%), regular (23% - 45%), bom (46% - 68%), muito bom (69% - 90%) e excelente (92% - 100%). Esses critérios foram definidos baseados na divisão dos 5 indicadores, porém, foi considerado que o indicador excelente teria um peso menor, já que um problema para ser muito similar precisa ser quase idêntico ao descrito, o que dificilmente será possível dada a liberdade que a linguagem natural permite. Dessa maneira, a Tabela 4.5 apresenta um resultado caracterizado como bom e na Tabela 4.6 o resultado foi classificado como muito bom.

Tabela 4.7 - Resultado do Processamento do Caso 3

Sentença Processada:
Team divided in two different countries (England and Brazil) of a large company face problem in the definition of the artifacts that will be generated, because in Brazil the custom of the teams was to generate specific artifacts and England the same company and its teams were used to generate other artifacts.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **51,13%, 49,22% e 49,19%,** respectivamente. -- Caso 1 -- **Description:** Management of activities. The project in question used the Scrum agile method and had something around 45 members, divided into five teams. One problem was to define the time required to perform certain tasks and use this information to make an adequate distribution of tasks between the teams. **Adopted Solution:** To manage and distribute the activities Mantis tool was used. To set the sizes of the activities the poker tool was used. The scrum masters of each teams met to define the macro activities and, subsequently, each team met to divide the macro activities into smaller activities, also making use of the planning poker technique. **Effectiveness:** The activities began to be better distributed. Besides, we had greater control of the activities that were being carried out and what still needed to be done. -- Caso 2 --

Description:
Another problem in the design of teams of various nationalities is the difference in culture that affects the understanding of business rules. Russians, for example, treated the registration of users differently from the way performed by Americans and Brazilians. It was a culture established of the software factories there.

Adopted Solution:

Attempt to standardize the understanding across the internal trainings and better description of the use cases.

Effectiveness:

Description of well-defined business rules and the staff did not use vices related to development and test.

-- Caso 3 --
Description:
During the project, people needed to work in a distributed manner and carry meetings on specific dates to present the results.The definitions of activities and allocations were made in these meetings. However, other needs arose later. Hence, we faced the difficulty of organizing the new tasks.

Adopted Solution:

It was used e-mails and tools such as Google Talk. For collaborative writing of documents, it was used email services. Thereby, when someone finishes a task, he alerts another team member in order to continue the tasks.

Effectiveness:

Once a tool for collaborative documentation was not used, there was a case that information was lost because one of the authors added new content, overwriting a previous content that should have been kept.

As buscas executadas e apresentadas nas Tabelas 4.6 e 4.7 foram consideradas dentro do critério de avaliação como bons, sendo os resultados da Tabela 4.6 considerados como muito bom. A Tabela 4.8 apresenta o resultado do processamento da *string* "Communication problems due to failures on the internet with one of the teams in the state two different states from Brazil" e obteve 52.56% do caso com maior nível de similaridade, sendo considerado como bom também.

Tabela 4.8 - Resultado do Processamento do Caso 4

Sentença Processada:
Communication problems due to failures on the internet with one of the teams in the state two different states from Brazil
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **52,56%, 50,56% e 47,47%,** respectivamente. -- Caso 1 – **Description:** In a project with an unknown number of members working in one country, communication problems among the teams existed. **Adopted Solution:** Assign a leader for each team, who are to be responsible for the communication among all teams. **Effectiveness:**

-- Caso 2 --
Description:
In a project with an unknown number of members working in one country, geographical distance led to communication problems among the teams.

Adopted Solution:

Keep documentation of each team up-to-date and available to all members of the project in order to minimize the number of travels and meetings.

Effectiveness:

-- Caso 3 --
Description:
Difficulty in performing the "Daily Scrum" because part of the team works on different shifts or in different locations.

Adopted Solution:

Perform the Daily Scrum during two times a day, the meetings in the morning and afternoon were used to cover the members working in different schedules; and perform Daily Scrum via SKYPE (using text only) for those who were at different locations.

Effectiveness:

Tabela 4.9 - Resultado do Processamento do Caso 5

Sentença Processada:
Reducing of productivity and decreasing of teamwork because the communication regarding meetings and discussions between project members almost doesn't exist.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **57,95%, 46,9% e 45,41%,** respectivamente. -- Caso 1 – **Description:** Communication regarding meetings and discussions between project members. The development team was composed by four developers who communicated remotely. **Adopted Solution:** We used email, Google talk, Skype and, less frequently, phone. **Effectiveness:** In order to conduct meetings for planning, monitoring activities and definitions of what was being developed, these tools (google talk, skype etc) were efficient. We had a bit of trouble when it was necessary to carry day-do-day meetings, ask questions about the code, solve impasses, configure the system, etc. -- Caso 2 -- **Description:** Reduce loss of productivity and decreasing of teamwork because team members work in a distributed manner.

Adopted Solution:

Scheduling of weekly in-person meetings.

Effectiveness:

Even after the scheduling of weekly meetings, some of the members did not attend the meetings and still maintained low productivity.

-- Caso 3 --
Description:
Development of a project in which members were in different cities and only one was in the same city of the customer.

Adopted Solution:

Process control by those who were located in the city of the customer, use of a support tool to delegate tasks, use of software version control for programmers and designers, and individual meetings via google talk.

Effectiveness:

Everything occurred as expected and all the challenges were identical to those that could happen to a local team.

A Tabela 4.9 contém 3 casos similares, onde o primeiro teve similaridade de 57.95% e os demais 46.9% e 45.41%. Dessa forma, os resultados dos três casos recomendados, seguindo os critérios de avaliação, são respectivamente Bom, Bom e Regular.

Tabela 4.10 - Resultado do Processamento do Caso 6

Sentença Processada:
Problem for defining roles for teams located in different countries.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **49,54%, 43,4% e 41,67%,** respectivamente. -- Caso 1 – **Description:** The biggest problem for project teams of various nationalities regards communication. **Adopted Solution:** In this case, we established a common language (English). Although It seems a simple task, some cultures tend to not accept this change. **Effectiveness:** -- Caso 2 -- **Description:** The project was developed with only one developer located in another workspace. **Adopted Solution:** The local development team remained motivated during the project. However, the developer that worked in another location was unmotivated.

Effectiveness: Daily meetings via skype and technological help via gtalk. -- Caso 3 -- **Description:** Schedule and hold meetings with team members in different geographical locations. **Adopted Solution:** To schedule meetings use Google Calendar and to make meetings use Skype or MSN Messenger or Google Talk with audio and video conversations. **Effectiveness:** Sometimes the communication was not very good and the members did not seem to work in groups, so sometimes it was necessary to make physical meetings.

Os casos recomendados e apresentados na Tabela 4.10 tiveram resultados de similaridade mais baixo, onde o caso mais similar foi de 49.54%, o que o mesmo pode ser considerado bom. Os outros casos recomendados são considerados regulares, no sentido de que obtiveram menos de 46% de similaridade.

Na Tabela 4.11 o nível de similaridade mais alto foi de 70.02% e o segundo caso recomendado foi de 65.77%. Dessa maneira o resultado do primeiro foi considerado muito bom e o segundo foi considerado bom.

Tabela 4.11. Resultado do Processamento do Caso 7

Sentença Processada:
Communication between members of a small team in a distributed software development project.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **70,02%, 65,77% e 59,31%,** respectivamente. -- Caso 1 – **Description:** Communication among small team members in distributed project using a RUP adaptation. **Adopted Solution:** Utilization of Messenger. **Effectiveness:** Presented several problems such as the impossibility of sharing code. -- Caso 2 -- **Description:** Choose of development software process for a small team using distributed software development. **Adopted Solution:** Adaptation of the RUP development process. **Effectiveness:**

It had many problems, but with adaptations this could be used.

-- Caso 3 --
Description:
Communication between product owner and team project in a distributed project using an adaptation of SCRUM.

Adopted Solution:

Utilization of a forum, email or asynchronous communication tools.

Effectiveness:

Os resultados apresentados na Tabela 4.12 tiveram 3 resultados recomendados para a sentença processada, com os níveis de similaridade de 61,55%, 60,93% e 56,13%, sendo todos considerados bons.

Tabela 4.12. Resultado do Processamento do Caso 8

Sentença Processada:
The deadlines were not met in a project with 12 members from 3 different countries because of time wasted with unnecessary tasks.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **61,55%, 60,93% e 56,13%,** respectivamente. -- Caso 1 – **Description:** In a project with a 9 members team working in 3 different cities, team members were not meeting the deadlines. **Adopted Solution:** Deadline estimations were assigned to the member responsible for each activity. **Effectiveness:** -- Caso 2 -- **Description:** In a project with 24 members from 11 different countries deadlines were not met. **Adopted Solution:** New scheduling method: two deadlines per project phase. **Effectiveness:** -- Caso 3 -- **Description:** In a project with 202 members working in two different countries, there existed problems in the use of different Process Models. **Adopted Solution:** Implement the same Process Model for all teams in the project. **Effectiveness:**

Na Tabela 4.13 são apresentados 3 casos recomendados, um deles com a taxa de similaridade relativamente alta, sendo 74.72%, o que é considerado como muito bom. O segundo caso em 66.43% e o último 50.82%, ambos considerados como bons. Logo, essas recomendações encontradas podem ser consideradas úteis para a resolução do problema em questão.

Tabela 4.13. - Resultado do Processamento do Caso 9

Sentença Processada:
Difficulty in performing the "Daily Scrum" because part of the team works on different locations and works with a difference of 3 hours.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **74,72%, 66,43% e 50,82%,** respectivamente. -- Caso 1 – **Description:** Difficulty in performing the "Daily Scrum" because part of the team works on different shifts or in different locations. **Adopted Solution:** Perform the Daily Scrum during two times a day, the meetings in the morning and afternoon were used to cover the members working in different schedules; and perform Daily Scrum via SKYPE (using text only) for those who were at different locations. **Effectiveness:** -- Caso 2 -- **Description:** We had problems related to the synchronization of working hours because part of the team works in a different time zone with 4 hours difference. **Adopted Solution:** We define what we call "core hours", which is the period of time of intersection of the entire team. During this period, all team members try to stay dedicated to the project. **Effectiveness:** This increased interaction between all staff (local and remote), which helped not only to improve communication, but also in the formation of pairs (pair programming) remotely. -- Caso 3 -- **Description:** In a project with a 9 members team working in 3 different cities, lack of knowledge on the technologies used in the project was reported. **Adopted Solution:** Face-to-face training sections and workshops at the begining of the project. **Effectiveness:** It was difficult to draw up a schedule fit for all members. Discussed material was sent for absent members.

Na Tabela 4.14, são apresentadas recomendações com a taxa de similaridade relativamente alta. Nesse caso, o caso mais similar dos testes realizados foi recomendado com a taxa de 93.2%, sendo o segundo com 72.2% e o terceiro 68.4%, o primeiro considerado como excelente e os demais considerados como muito bom.

Tabela 4.14. Resultado do Processamento do Caso 10

Sentença Processada:
In a project with a unknown members team working in different cities, a reduction in the number of members happened.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **93,2%, 72,2% e 68,4%,** respectivamente. -- Caso 1 – **Description:** In a project with a 9 members team working in three different cities, a reduction in the number of members happened. **Adopted Solution:** Reassignment of roles. **Effectiveness:** Problems reported by the client were fixed first, which led to an early release of some features and the postponement of some others. -- Caso 2 -- **Description:** In a project with a 9 members team working in 3 different cities, team members were not meeting the deadlines. **Adopted Solution:** Deadline estimations were assigned to the member responsible for each activity. **Effectiveness:** -- Caso 3 -- **Description:** In a project with an unknown number of members working in one country, communication problems among the teams existed. **Adopted Solution:** Assign a leader for each team, who are to be responsible for the communication among all teams. **Effectiveness:**

Mesmo com o grau de similaridade baixo na Tabela 4.15, os casos apresentados recomendam problemas e soluções que podem ser consideradas úteis, caracterizando um resultado regular. Dessa maneira, pode-se dizer que este não é um resultado completamente inválido, mas para a avaliação dos testes, serão considerados apenas os resultados classificados como bom, muito bom e excelente, e assim, os resultados classificados como fraco e regular não serão considerados.

Tabela 4.15. - Resultado do Processamento do Caso 11

Sentença Processada:

Absence by customer from a different place to go with the project manager. Hindering the elicitation of requirements.

Resultado

Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de 45,56%, 45,1% e 45%, respectivamente.

-- Caso 1 --

Description:

There is a need to manage the schedule of activities performed by a team of developers distributed in different cities.

Adopted Solution:

The usage of online and shared tools among team members, for example, google calendar and google docs. Furthermore, conducting online meetings, via Skype or Gtalk, and in-person monthly meetings.

Effectiveness:

The use of tools to assist the monitoring of project activities for all its members was essential. Since the tools are shared online, all team members were able to monitor what activities the members were executing and what tasks should be performed in the future. Nevertheless, monthly meetings were needed to complement the development and clarify any problems regarding the understanding of the project requirements.

-- Caso 2 --

Description:

Development of a project in which members were in different cities and only one was in the same city of the customer.

Adopted Solution:

Process control by those who were located in the city of the customer, use of a support tool to delegate tasks, use of software version control for programmers and designers, and individual meetings via google talk.

Effectiveness:

Everything occurred as expected and all the challenges were identical to those that could happen to a local team.

-- Caso 3 –

Description:

The documentation of the use cases is outdated or nonexistent due to flaws in the initial project settings.

Adopted Solution:

Identification of features that are not covered by the documentation and the process of decision making.

Effectiveness:

The development team has now a valid referential regarding what the system executes and the users may now solve their doubts by their own.

Na Tabela 4.16, a descrição do caso mais similar com **62.04%** de similaridade, foi “In a project with 24 members from 11 different countries deadlines were not met.”, de fato um caso bastante similar ao novo problema, o que demonstra que sua solução pode ser facilmente adaptada para o novo problema. O mesmo ocorre para o segundo caso da lista, já que sua taxa de similaridade é muito próxima a do primeiro caso.

Tabela 4.16. Resultado do Processamento do Caso 12

Sentença Processada:
The deadlines were not met in a project with unknown members from different countries because of time wasted with unnecessary tasks.
Resultado
Os casos com descrições mais similares recomendados apresentaram taxa de similaridade de **63,54%, 62,04% e 57,67%,** respectivamente. -- Caso 1 – **Description:** In a project with 24 members from 11 different countries deadlines were not met. **Adopted Solution:** New scheduling method: two deadlines per project phase. **Effectiveness:** -- Caso 2 -- **Description:** In a project with a 9 members team working in 3 different cities, team members were not meeting the deadlines. **Adopted Solution:** Deadline estimations were assigned to the member responsible for each activity. **Effectiveness:** -- Caso 3 -- **Description:** In a project with 4 members from the same country time was wasted with unecessary tasks. **Adopted Solution:** Determine which tasks were unecessary and the time spent with them, for improving the project management. **Effectiveness:**

Para validação desses casos em si, o teste binomial unilateral à direita [Gamerman; Migon, 1993] foi aplicado à amostra de acertos encontrada nos testes. Esse teste de proporção foi escolhido por ser comum e de frequente utilização no meio acadêmico. Os resultados obtidos foram: limiar de 80%, o nível de significância de 5% e o p-valor 10%, dessa forma, como o p-valor é menor que o nível de significância a hipótese nula não é aceita, ou seja, estaticamente a abordagem proposta recomenda corretamente casos similares em mais de 80% dos casos de utilização e é viável. (0.8 < taxa de sucessos < 1; p-value=0,0107).

Assim, a taxa de acertos de 91,67% foi estatisticamente superior à 80% e de fato significativa, então, é possível perceber que em 91,67% dos 12 (doze) testes realizados, a abordagem proposta extraiu as características da descrição em texto e recomendou de forma coerente casos similares para apoiar na resolução dos novos casos, se tornando uma abordagem viável para auxílio nesse contexto.

4.6 Considerações Finais

Este capítulo descreveu todo o processo de funcionamento da abordagem desenvolvida neste trabalho, apresentando as três principais contribuições do trabalho, as principais características e fundamentos da DKDOnto, o funcionamento geral e funcionamento da ferramenta DKDs, que manipula as informações da ontologia proposta e a descrição geral do funcionamento e detalhes de implementação da DKD-CBRs, ferramenta esta que utiliza RBC e PLN para auxiliar projetos distribuídos.

Para finalizar o capítulo, também foi apresentada a demonstração dos testes executados com intuido de validar a proposta criada. Esses testes foram realizados apenas com uma amostra limitada, com objetivo de ter conhecimento sobre a viabilidade da abordagem.

5. Trabalhos Relacionados

Nesse capítulo serão apresentados os trabalhos sob o mesmo tema desta pesquisa, com objetivos e resultados similares. Estes trabalhos tiveram em comum o fato de serem propostas ferramentas fortemente baseadas no uso de Ontologias e websemântica, bem como RBC para auxiliar em cenários comumente enfrentados não somente por equipes distribuídas, mas também por equipes co-alocadas.

Com a análise desses artigos relacionados percebeu-se que são poucas as ontologias propostas para Engenharia de Software, e menos ainda são as específicas para o cenário de desenvolvimento distribuído, fato que motivou o estudo da ontologia proposta nessa pesquisa com intuito de inseri-la no contexto distribuído.

Tabela 5.1. Matriz comparativa entre trabalhos relacionados

Trabalhos / Características	**Contexto de Aplicação**	**Área ES**	**Técnicas Empregadas**	**Experimentação ou Validação**	**Uso na Indústria**
Dillon e Simmons (2008)	Desenvolvimento Tradicional	Toda área	Ontologias	Não	Não
Maalej e Happel (2008)	Desenvolvimento Tradicional	Toda área	Ontologias	Não	Não
Brady A. et al (2010)	Desenvolvimento Tradicional	Estimativas de Esforço	RBC e PLN	Sim	
Marques, A. B. et al (2013)	DDS	Alocação e Atribuição de Tarefas	Ontologias	Sim	Sim
Urnau, Kipper, Frozza (2014)	SI	-	RBC	Não	Não
Rocha (2015)	DDS	Toda área	Ontologias, RBC e PLN	Sim	Não

A Tabela 5.1, apresenta algumas características envolvidas nos trabalhos relacionados, de maneira que possa ser feita uma breve avaliação das comparações entre os trabalhos. Na primeira coluna da tabela se encontram os trabalhos utilizados, sendo o último, referente a essa tese. A segunda coluna diz respeito ao foco do trabalho, o que ele propõe tentar solucionar, a terceira qual a área de aplicação do trabalho dentro da engenharia de software, a quarta coluna se refere as técnicas que foram empregadas, a quinta coluna se houve algum tipo de simulação, validação ou experimentação para aquele trabalho e por fim, a última coluna identifica se o trabalho possuiu validação ou utilização na indústria. Maiores detalhes sobre cada trabalho são descritos nas seções seguintes.

5.1 Dillon e Simmons (2008)

Em seu trabalho, Dillon e Simmons (2008) descrevem o uso de ontologias em diferentes aspectos da Engenharia de Software com alguns importantes destaques que serão citados a seguir:

- Proporcionar um mecanismo de comunicação robusto e não ambíguo além de meios de referência para engenheiros de software trabalhando em múltiplos locais para o desenvolvimento de software;

- Fornecer um mecanismo mediador para o acesso de dados e fontes de informação heterogêneos, particularmente na Web;
- Permitir a construção de aplicações na Web fornecendo semântica claramente definida para web-services;
- Fornecer uma base comum de conhecimento para multi-agentes trabalhando num domínio particular;
- Fornecer semântica definida claramente e confiança para interações na Web, mais especificamente: Sistemas de Confiança e Reputação e sistemas baseados em privacidade;
- Fornecer semântica definida claramente para o conhecimento em um número de diferentes domínios

Dessa maneira, o trabalho apresenta uma das primeiras contribuições no quesito de ontologia de Engenharia de Software, como também uma ontologia de gerenciamento de projeto em conjunto com uma ontologia de domínio que são usadas pra fornecer suporte ao DDS, ajudando na resolução dos quatro problemas enfrentados pelo DDS: comunicação e coordenação, compartilhamento de conhecimento unificado, plataforma de compartilhamento de conhecimento e adaptação de metodologia e validação.

A ontologia de Engenharia de Software é então descrita como sendo divida em várias sub-ontologias:

- A ontologia de requisitos de software que consiste nas quatro sub-ontologias seguintes: uma sub-ontologia de requisitos, uma sub-ontologia de elicitação de requisitos, uma sub-ontologia de análise de requerimentos e uma sub-ontologia de especificação de requerimentos.
- A ontologia de projeto de software que significa uma sub-ontologia de atividade de projeto, uma sub-ontologia de projeto arquitetural, uma sub-ontologia de projeto detalhado, e uma sub-ontologia de métodos e estratégias de projeto.
- A ontologia de construção que é uma ontologia de linguagem de construção, uma sub-ontologia de codificação e uma sub-ontologia e reuso.
- A ontologia de teste de software que se desenvolve nas seguintes sub-ontologias: uma sub-ontologia de problemas de teste, uma sub-ontologia de alvos de testes, uma sub-ontologia de objetivos de testes, uma sub-ontologia de técnicas de teste e uma sub-ontologia de atividades de teste.
- A ontologia de métodos e ferramentas de software que se divide em sub-ontologia de ferramentas de software e uma sub-ontologia de métodos de software.

Desse modo e como uma forma de auxílio a problemas mais complexos, uma das possíveis propostas desse trabalho apresentado nessa seção é utilizar um sistema multi-agentes na intenção de mitigar o problema da certificação de que as bases de conhecimento dos diferentes agentes são coerentes e consistentes umas com as outras.

Conforme descrito acima, o trabalho de Dillon e Simmons (2008) dá suporte a Engenharia de Software no geral, porém, não possui nada específico para o desenvolvimento de software com equipes distribuídas, o que não traz garantias que o uso desse conjunto possa ser realmente efetivo para um projeto totalmente distribuído.

Dessa forma, um ponto negativo a ser citado presente no trabalho desses autores é o fato de ele

não apresentar um estudo de caso com a utilização do conjunto proposto de soluções para equipes distribuídas, comprovando sua real eficácia.

Outro ponto importante que merece destaque está na não disponibilização de *link* ou material mais específico sobre suas ontologias, restringindo assim, o acesso a maiores informações e até mesmo a reutilização destas para melhores fins.

Diante disso, é possível afirmar que esta presente pesquisa possui algumas características particulares quando comparada ao trabalho desenvolvido por Dillon e Simmons (2008), mesmo por que utiliza uma ontologia específica para o DDS com classes específicas que determinam quais são os maiores desafios enfrentados e as melhores soluções para os problemas que surgem, e desse modo, essa base de conhecimento permite que os projetos possam se utilizar de informações já armazenadas para indução das melhores práticas desde o início do projeto até o seu término, e também para evitar o surgimento de novos problemas. Além disso, com uma ontologia específica a ser aplicada em DDS encontram-se classes específicas para a distribuição das equipes.

5.2 Maalej e Happel (2008)

Em seu trabalho, Maalej e Happel (2008) determinaram a comunicação informal como o ponto chave para a distribuição de conhecimento em equipes de desenvolvimento de software co-localizadas, e desse modo, para que seja possível a distribuição do conhecimento dentro dessas equipes distribuídas, as infraestruturas de desenvolvimento precisam criar pontes com articulação leve e meios de compartilhamento para a evolução do desenvolvimento de conhecimento.

Segundo os autores, a ausência de suporte para o compartilhamento de software limita a eficiência e a produtividade dos desenvolvedores distribuídos exatamente por o desenvolvimento de software se tratar de uma atividade intensiva em conhecimento, em que a tentativa e erro é uma técnica dominante para a resolução de problemas.

Nesse sentido, estes autores propuseram um *framework* baseado em ontologias para a captura, acesso e compartilhamento de experiências de desenvolvedores numa maneira descentralizada e contextualizada e assim concluíram que *framework* “permite que equipes distribuídas se tornem mais eficientes ao aprender com as experiências uns dos outros”.

Como um primeiro passo para a conceitualização de um *framework* para o compartilhamento de conhecimento são descritas as capacidades desejáveis do sistema e deficiências no acesso e compartilhamento de conhecimento e identificados os habilitadores para esse *framework*. Assim, a partir de uma perspectiva de processo, o compartilhamento de conhecimento pode ser definido como dois problemas, o problema de se procurar (buscando-se e identificando-se) e o de transferir (mover e incorporar) conhecimento ao longo de subunidades organizacionais. Ele atua como mediador entre dois papéis: o consumidor de conhecimento que se beneficia numa situação específica e o fornecedor de conhecimento que contribui para a experiência do time.

De acordo com esse aspecto, o acesso e compartilhamento de conhecimento são os processos-chave para se oferecer suporte a esses papéis, para os autores existem cinco habilitadores comuns para o suporte ao compartilhamento de conhecimento em equipes distribuídas: a contextualização, a personalização, a assistência proativa, a descentralização e a integração.

Eles vão além e descrevem como as tecnologias semânticas são usadas como um importante habilitador e assim introduzem sua arquitetura para o compartilhamento de conhecimento. Tecnologias semânticas para o contexto identificado pelos autores são definidas como tecnologias de software que permitem que o significado e as associações entre informações sejam conhecidos e processados em tempo de execução, baseados sempre em ontologias.

Devido a sua natureza formal, mas expansível, tecnologias semânticas fornecem mecanismos para a solução dos problemas mencionados anteriormente, nesse sentido, primeiramente as ontologias fornecem o nível apropriado de abstração requerido para uma estruturação eficiente de conhecimento com a capacidade de expansão e alteração de modelos subjacentes.

Em seu âmbito, as experiências de desenvolvedores podem ser capturadas sem destilação e novos conceitos, como novas tecnologias, tipos de erros ou fontes de conhecimento podem ser facilmente adicionados sem que seja necessário um redesenho dos modelos semânticos subjacentes.

Assim, as informações semânticas que descrevam as experiências dos desenvolvedores, a exemplo da mensagem de erro: "A" foi resultado da ação de construção "B" usando-se a infraestrutura de construção "C", podem então ser descritas ou comentadas usando-se conceitos ontológicos.

Por outro lado, as ontologias permitem a representação formal de relacionamento, em particular, taxonomias, estas necessárias para similaridades semânticas, para conhecimento requerido e fornecido, além disso, apresentam também um vocabulário unificado que assegura comunicação sem ambiguidades entre membros de comunidades heterogêneas.

De forma geral, o *framework* proposto é descrito como composto de três camadas lógicas: o Modelo de Conhecimento Distribuído, o Sistema Contextual e o Desktop Contextual incluindo o Fornecedor de Conhecimento e o módulo de Captura de Conhecimento. O Modelo de Conhecimento Distribuído é uma camada constituída de um conjunto de Ontologias, de um Armazenamento local de Metadados e uma Infraestrutura P2P. Seu turno, o Sistema Contextual é uma camada que permite a captura da experiência de um usuário e sua análise para prepará-la para reuso futuro e é constituída de três módulos: Monitor Contextual, Interpretador Contextual e Perfilador.

A maior inovação do trabalho desenvolvido pelos citados autores é a combinação de tecnologias semânticas, consciência do contexto e infraestruturas P2P as quais permitem uma aquisição e compartilhamento de conhecimento personalizado e proativo. O *framework* proposto por Maalej e Happel (2008) é baseado em ontologias para a captura, acesso e compartilhamento de experiências de desenvolvedores numa maneira descentralizada e contextualizada, porém, este estudo é mais um exemplo que não utiliza a experimentação de sua proposta em equipes reais ou em simulações, nem sobre as ontologias propostas e nem sobre o *framework* em si, o que limita significativamente sua avaliação e análise.

Além disso, o trabalho desenvolvido pelos autores, não apresenta um link ou referências específicas sobre suas contribuições, não permitindo que qualquer estudo se iniciasse por meio dele. Outro fator importante e mais um ponto negativo dentre os acimas descritos identificado durante a análise é o fato de os referidos autores não utilizarem uma ontologia propriamente de DDS, o que pode não cobrir todas as demandas de um projeto distribuído.

Assim como o trabalho anteriormente comparado, o que diferencia a DKDOnto dessa abordagem acima citada é o desenvolvimento de classes específicas para a distribuição de equipes, como as classes de localização, classes de habilidades e classes de problemas e soluções.

5.3 Brady A. et al (2010)

Brady et al (2010) afirmam que pesquisas existentes sobre estimativas de esforços focam em sua maioria na obtenção de estimativas baseadas em dados anteriores de projetos através do uso de modelos paramétricos, raciocínio baseado em casos (RBC) ou algoritmos genéticos. Entretanto, ressaltam que tais pesquisas não indicam como alterar um projeto de forma a reduzir estimativas de esforços. Dessa forma, os autores apresentam o "W", um sistema de raciocínio baseado em casos e

contrast set learning [Webb, Butler e Newlands, 2003] o qual sugere mudanças de projetos visando à redução de estimativas de esforço.

Uma das vantagens do "W" sobre soluções propostas anteriormente reside no fato de ele não necessitar de um modelo paramétrico subjacente o que possibilita sua aplicação em mais bases de dados visto que tais dados não precisam estar no formato requerido por modelos de processo de software padrão.

Os autores nesse trabalho comparam uma ferramenta própria chamada STAR/NOVA, proposta anteriormente da seguinte forma: W é descrito como possuidor de implementação e uso mais simples, com centenas de linhas de código na linguagem de script AWK, enquanto a solução anterior possui milhares de linhas de C++ e LISP.

Devido a aspectos negativos do STAR/NOVA, tais como, dependência de um modelo, dependência de dados formatados de acordo com esse modelo adotado, inflexibilidade reportada na etapa de codificação dos modelos de processo, performance, tamanho e manutenção do projeto, os autores decidiram adotar o raciocínio baseado em casos, culminando na criação do "W".

Eles descrevem a técnica de raciocínio baseado em casos com fundamento na adaptação direta de casos anteriores, relacionando a similaridade de tais casos com a situação atual. As quatro etapas do ciclo de funcionamento de um sistema RBC (recuperação, reutilização, revisão e retenção) não incluem cálculos baseados em nenhum modelo explícito, o que proporciona duas vantagens: permite que se opere independentemente de modelos em utilização, possibilitando assim, a análise de mais bases, e também proporciona uma melhoria de desempenho dada a fase de recuperação possivelmente ser mais eficiente que os cálculos realizados pela ferramenta STAR/NOVA.

A aplicação de RBC em "W" ocorre da seguinte forma, as vizinhanças locais são divididas em um conjunto de atributos com os melhores valores de classe e um conjunto com os atributos restantes. Assim, um *constrast set* que separa essas regiões é aprendido.

Por conseguinte, "W" finalmente busca um subconjunto do *contrast set* que seleciona de forma mais eficiente e a região com as menores estimativas de esforço, desse modo, "W" encontra *constrast sets* usando busca gulosa, em que candidatos a *contrast sets* são ranqueados pelo log de suas razões de possibilidades. Os resultados reportados pelos autores da utilização de "W" em bases de dados de projetos provenientes de entidades como ISBSG, Desharnais, Cocomo e NASA permitiram a observação de que na maioria dos casos, W encontrou tratamentos que melhoraram tanto a mediana como a variação das estimativas de esforço, apenas em alguns casos não houve melhoria da mediana, e a variação teve uma diminuição de apenas 18%.

No quesito de RBC em si, as duas abordagens são similares, porém o foco do trabalho apresentado nessa seção é em redução de esforço no desenvolvimento de software, mas em se tratando da contribuição como um todo, o trabalho proposto nessa tese engloba o cenário distribuído, através da ontologia específica de DDS e do sistema RBC que utiliza casos do cenário distribuído também. Ou seja, tem como objetivo auxiliar em todo processo de desenvolvimento de software distribuído e não em um único ponto específico.

A maior limitação dessa ferramenta comparando com a proposta por esta tese é que a “W” não abrange o ambiente distribuído, ela é focada apenas para o desenvolvimento co-localizado. Outro fator que pode ser citado como ponto negativo é o fato de que utiliza um sistema típico de RBC, não utilizando outros recursos, como a ontologia por exemplo.

5.4 Marques, A. B. et al (2013)

O artigo propõe a criação de uma ontologia para alocação e atribuições de tarefas para o contexto distribuído a qual leva em consideração as características da equipe, as diferenças entre as equipes, tanto técnicas como culturais, e também o próprio produto em desenvolvimento, (complexidade, dependências, habilidades necessárias).

Neste trabalho agora analisado, a atribuição de tarefas é vista como um processo que decide a maneira pela qual as tarefas do projeto serão executadas, como e quais os recursos necessários. Mostrando que o plano de atribuição de tarefas deve sempre considerar as características das equipes distribuídas e do produto a ser desenvolvido, características que estão embutidas em três aspectos de muita importância: duração do projeto, qualidade do produto e a comunicação. A atenção a esses aspectos é indispensável para a obtenção de sucesso no momento da criação do plano de atribuição de tarefas.

Dessa maneira, é possível afirmar que não existe consenso na literatura sobre fatores, critérios e métodos que devem ser consultados no momento do planejamento de alocação de tarefas para equipes distribuídas. Porém, algumas propostas foram apresentadas com o objetivo de trazer tal mecanismo para a literatura, cada uma focando em alguns dos aspectos citados anteriormente.

A ontologia de domínio proposta tem como finalidade proporcionar uma referência para melhorar a compreensão dos conceitos existentes que estão relacionados com a atividade de alocação de tarefas para equipes distribuídas, sugerindo as melhores escolhas para identificar e analisar as diferentes alternativas para a alocação dessas tarefas.

Assim, os conceitos abordados estão ligados ao planejamento da alocação de tarefas e aos aspectos que influenciam a tomada de decisões durante esse planejamento. O estudo da ontologia foi obtido com base em resultados de um mapeamento sistemático da literatura, opiniões de especialistas e em duas outras ontologias.

Esse citado trabalho abordou a ontologia em fragmentos organizadas pelos principais conceitos que a compõe: artefatos, atividades, competências, equipes, organizações e equipes de projeto:

- Um artefato representa o produto resultante para o processo de desenvolvimento podendo ser visto como o projeto final (alto nível) ou cada artefato parcial do produto resultante (baixo nível). Um artefato possui importância e prioridade, onde esses são conceitos inter-relacionados, no entanto são atributos independentes. Um artefato também pode ser decomposto em sub artefatos independentes, e possuem natureza própria;
- Atividade representa as restrições que uma tarefa pode ter, possui um tipo, competências, procedimentos adotados, restrições, subatividade e pré-atividades (atividades dependentes de outras atividades);
- Competências devem ser consideradas afim de identificar qual time tem as competências necessárias para o desenvolvimento da atividade. As pessoas adquirem competências ao longo da sua carreira profissional, tornando os recursos humanos capazes de execução das atividades de um projeto. Assim, esse conceito pode ser classificado como conhecimentos, habilidades (ou habilidade), e experiências. Outros pontos importantes desse fragmento é o histórico de desempenho e o nível de fluência em determinadas linguagens;
- A equipe é um fragmento que representa as características das equipes e a colaboração entre elas, incluindo conceitos relacionados às diferenças geográficas e culturais, histórico, experiência com outras equipes, bem como seu período de trabalho, alinhamento cultural e custo;
- Outro fragmento é a organização, que pode participar do mesmo projeto ou exercer funções de organização, fornecendo o escritório de sede, subsidiária ou escritório subcontratado. Esse conceito,

dentro da ontologia, esclarece como as organizações são compostas, sua localização, e os relacionamentos entre as organizações que participam do projeto;

- Por fim, o último fragmento é equipes de projeto, que através da contratação de recursos humanos, tarefa realizada pelas organizações que participam do projeto, as equipes de projetos são montadas com o objetivo de realizar todas as atividades do projeto, atividades que foram atribuídas a cada equipe distribuída. Seu objetivo específico são as metas, que podem ser decompostas em submetas, e que por sua vez tem níveis de prioridade diferentes. O objetivo com maior prioridade são chamados de *postgoals*.

Para esse trabalho foram considerados aspectos que representam a dispersão geográfica, as distintas culturas, a diferença de custo, e as características voltadas à colaboração entre equipes e entre as organizações em um projeto distribuído. A validação da ontologia foi feita de forma preliminar, utilizando opiniões de especialistas que nesse caso indicaram que os relacionamentos representados são válidos para o ambiente real, permitindo assim que a ontologia auxilie a tomada de decisões no processo de planejamento de alocação de tarefas.

A ontologia estudada aqui fornece uma compreensão comum sobre o relacionamento entre os conceitos e à alocação de tarefas, porém, mesmo que existam algumas propostas na literatura para sustentar este processo, ainda não existe uma concordância sobre o que deve ser respeitado no momento do planejamento da alocação de tarefas.

Nesse sentido, este trabalho fornece um importante recurso para o desenvolvimento de software com equipes distribuídas com objetivo de auxiliar a distribuição de atividades para as equipes, permitindo que a gerência tome decisões com maior embasamento no momento de alocar tarefas para os membros.

O trabalho proposto por esta tese se difere do aqui apresentado no sentido de que pode envolver todas as fases do desenvolvimento de software, podendo ser utilizado pela gerência de desenvolvimento, bem como pelos desenvolvedores como um todo, além de utilizar um sistema de RBC para recomendações de casos. Um fator limitante encontrado no trabalho anteriormente analisado é a colaboração apenas na alocação de tarefas, deixando outras atividades tanto de gerência como de desenvolvimento no geral em aberto.

Ademais, é também importante ressaltar que a ontologia pode ser melhor validada através de ferramentas específicas, verificando suas classes, relacionamentos, axiomas e propriedades, o que não acontece com o sistema desenvolvido pelo autores em análise.

5.5 Urnau, Kipper, Frozza (2014)

Este trabalho descreve o desenvolvimento de um sistema para apoio à tomada de decisão percorrendo os temas que abordam o conhecimento, as estratégias utilizadas nas empresas e o uso da técnica de raciocínio baseado em casos.

A proposta deste trabalho foi de utilizar os recursos da inteligência artificial, com o uso da técnica de sistemas baseados em conhecimento, para analisar continuamente informações oriundas do planejamento estratégico da empresa, no setor comercial, com a finalidade de identificar padrões de conhecimento existentes nos resultados e nas decisões tomadas.

Assim, será possível realizar análises detalhadas e não somente sintéticas, contribuindo para a qualidade da decisão tomada pelo gestor da empresa. A visualização destas informações permite os gestores fazem associações na busca por uma solução mais adequada para o problema enfrentado.

A metodologia utilizada foi do tipo exploratório e descritiva. É exploratória porque realizou investigações por meio de visitas, observações, aplicação da técnica de Brainstorming e de conversas informais. E é também descritiva porque faz o registro e o relato, de forma sistemática, descrevendo como foi realizada a pesquisa em suas etapas.

Esse trabalho utilizou como algoritmo de similaridade o vizinho mais próximo, como também os modelos: linear da faixa de interesse para cada atributo, matriz simétrica de similaridade e tabela de definição de valores por atributos. Com intuito de obter melhoria na acurácia das informações em tempo real, auxiliando na tomada de decisões.

É um sistema de RBC utilizado para tomada de decisões num ambiente comercial. Não tem vínculo com o desenvolvimento de software, principalmente com equipes distribuídas. Não possui um experimento demonstrando a viabilidade dessa ferramenta, apenas algumas telas e suas descrições.

Um outro ponto negativo observado neste trabalho, é que o mesmo possui uma quantidade pequena de casos que foram usados nos testes e também não explica como esses casos foram criados.

5.6 Resumo do Capítulo

Este capítulo teve como objetivo, apresentar quais foram os trabalhos relacionados identificados sobre o mesmo tema ou tema similar a esta pesquisa. É importante salientar que estes estudos foram encontrados a partir da realização do mapeamento sistemático da literatura.

Durante as quatro seções do capítulo foram apresentados os objetivos gerais e os resultados, fornecendo assim um panorama geral das contribuições propostas por estas pesquisas. Para cada seção, além da breve descrição dos trabalhos, foi realizada uma análise geral com levantamento de pontos negativos dos trabalhos relacionados, bem como as principais diferenças entre as contribuições já existentes na literatura com relação às contribuições do presente trabalho.

6. Considerações Finais

Baseados nos resultados obtidos, nesse capítulo serão apresentadas as considerações finais do trabalho, divididas da seguinte forma: as conclusões, descritas na seção 6.1, as principais limitações enfrentadas no desenvolvimento do trabalho estabelecidas na seção 6.2, desde as atividades iniciais até as finais e por fim, e as especificações dos trabalhos futuros encontrados na seção 6.2.

6.1 Conclusões

Com toda evolução tecnológica, o software se tornou um diferencial competitivo, tornando cada vez mais as organizações dependentes dele. Nessa perspectiva, o número de empresas que desenvolvem sistemas e que os utilizam cresceu significativamente nas últimas décadas. Desse modo, com todo esse crescimento do mercado de software as organizações têm buscado, cada vez mais, a implantação de processos produtivos visando a melhora na qualidade e na redução de custos. Dessa forma e objetivando necessariamente a redução dos custos, a melhoria na qualidade do produto, aumento na produtividade e competitividade global, várias empresas optam por distribuir seus processos de desenvolvimento em lugares diferentes, criando assim o DDS.

Mesmo com diversos fatores contribuindo para o crescimento do DDS, assim como do desenvolvimento co-localizado, construir sistemas de software não é uma tarefa simples, e no cenário de desenvolvimento distribuído a complexidade tende a aumentar. Isso por que, projetos de software assumem perspectivas diferentes e consequentemente, novos riscos, nesse sentido, caso não haja um bom conhecimento dos fatores que possam vir a influenciar no projeto, este terá mais chances de não obter sucesso.

Para reduzir a complexidade trazida com a aplicação do DDS, a utilização e junção de técnicas de IA podem ser úteis. Além dessas, ontologias podem ser aplicadas principalmente para conceder semântica para ferramentas auxiliares, favorecendo e construindo uma forte comunicação entre seus membros baseada no conhecimento, centralização e disponibilidade de informações bem como, fornecer uma base de conhecimento para que agentes de software capazes de formar um ambiente automatizado, possam apoiar o DDS. Ainda pode ser utilizadas para construção do DDS a combinação das técnicas de PLN e RBC as quais fornecem a liberdade da expressividade que a linguagem natural permite, como o uso de um conhecimento único que pode ser reusado ou não, de acordo com o caso em questão.

Para a realização desse trabalho foram utilizados alguns métodos de pesquisa, o primeiro, um mapeamento sistemático da literatura com o objetivo identificar modelos, ferramentas, técnicas e boas práticas com utilização de ontologias no apoio ao DDS, além de ontologias formalizadas neste contexto. Nesse primeiro método, um total de 51 (cinquenta e um) estudos foram selecionados e diversas evidências foram apresentadas sobre cada artigo coletado, sendo a maioria dos trabalhos analisadas publicados a partir de 2006 até os dias atuais. Com isso, foi possível notar que a pesquisa e o desenvolvimento dessa área são bastante recentes, e por este motivo, o número de ferramentas, modelos e outros recursos encontrados não foi tão significativo.

Em seguida, o método utilizado foi a definição e construção da ontologia, logo, foi necessário a implementação da ferramenta que acessa a DKDOnto, o quarto passo foi a implementação do sistema RBC.

Por fim, dois métodos citados no capítulo 3 foram executados para viabilizar os testes e funcionamento da abordagem, no caso a construção da base de casos e o povoamento da ontologia. Para a base de dados e povoamento da ontologia, se utilizou da seleção e leitura de artigos para a

criação de casos. Estes retirados de algumas conferências específicas da área de ES e de Gerência de Conhecimento. A partir disso, 45 (quarenta e cinco) casos foram criados e utilizados para avaliação da abordagem proposta na pesquisa e diversas informações dos projetos descritos nos artigos foram incluídas como instâncias na ontologia.

Desse modo, e na tentativa de gerar avaliações mais próximas da realidade, uma pesquisa (questionário) foi realizada com profissionais que já trabalharam com DDS a fim de que estes viessem a informar os desafios já encontrados nos projetos por eles desenvolvidos e as soluções utilizadas, participando dessa pesquisa o total de 21 (vinte e um) profissionais, o que tornou possível a utilização e criação de mais 57 (cinquenta e sets) casos. No total a base de casos foi formada por 102 casos.

A principal contribuição desse trabalho girou em torno de uma abordagem baseada em ontologias e RBC para apoiar o DDS, nesse aspecto, uma ontologia foi proposta, um sistema de acesso a ela e um sistema de RBC. Esse mecanismo contém informações sobre os projetos no contexto distribuído incluindo desafios encontrados e soluções utilizadas. Dessa forma, através desse mecanismo, qualquer membro da equipe ou preferencialmente a gerência pode consultar a ferramenta e a base, para tentar evitar ou remediar problemas durante a execução do seu projeto.

Assim, a complexidade de um projeto pode ser reduzida porque os responsáveis por este podem consultar a base em busca de informações sobre soluções para possíveis problemas em qualquer circunstância do projeto, seja na fase de implementação, testes, requisitos, entre outros, bem como identificar melhores práticas para o desenvolvimento daquela configuração de projeto. Dessa forma, esse trabalho pode ser utilizado por organizações ou por projetos que utilizam o conceito de DDS para desenvolver software e que através da base de conhecimento e suas ferramentas possam ser utilizadas como fonte de pesquisa objetivando sugerir soluções para mitigar possíveis problemas.

A união dos conceitos de ontologia, RBC e PLN permitiram que essa proposta pudesse extrair e recomendar informações relevantes para apoiar decisões em projetos de software com times distribuídos. Permitindo o apoio a resolução dos desafios e limitações desse ambiente. Assim, a abordagem desenvolvida, com a DKDOnto e suas ferramentas cumprem com o objetivo para o qual foram propostos, consistindo um instrumento computacional que pode ser usada para o tratamento, a análise e o uso de informações sobre projetos de software com equipes distribuídas. Dessa forma, essa abordagem permite que os atores deste cenário possam obter e dispor de informações e artefatos corretos, fornecendo um modelo de conhecimento de alto nível para os envolvidos.

Os resultados obtidos até o momento são expressivos, onde por exemplo, o gerente tem conhecimento real de quais culturas estão envolvidas nos times, que o permite tratar melhor casos específicos, bem como é possível que ele busque por soluções para seus possíveis desafios. Da mesma forma um líder técnico tem acesso ao conhecimento técnico dos participantes do projeto, podendo solicitar ou direcionar atividades específicas para membros especializados.

Como validação para essa proposta, os testes iniciais executados utilizaram como base 102 casos de problemas e soluções extraídos de artigos científicos e do *survey* realizado. Baseado nessa experiência foi possível afirmar que houve sucesso com uma taxa de 91,7% nas recomendações dos casos válidos do passado para os problemas pesquisados.

Dessa forma, esse trabalho pode ser utilizado por organizações ou por projetos que utilizam o conceito de DDS para desenvolver software e que através da base de conhecimento e suas ferramentas possam ser utilizadas como fonte de pesquisa objetivando sugerir soluções para mitigar possíveis problemas.

6.2 Limitações do Trabalho

Na realização do mapeamento sistemático, as maiores limitações foram com relação a encontrar mais artigos, pois na primeira etapa do mapeamento, que foi realizada em 2011, apenas 38 estudos primários foram considerados, já nessa segunda etapa, realizada este ano, mais 13 foram encontrados entre 2012 e 2013, porém, esse ainda é um número relativamente baixo se comparado a outros mapeamentos sistemáticos.

O *survey* executado com intuito de obter casos reais provenientes da indústria, realizado com os profissionais da área atingiu uma amostra de 21 participantes, e foi enviado para listas de discussão de desenvolvimento de software, e assim, para diversos profissionais no mercado e listas de discussões específicas de DDS, assim, pode-se dizer que a quantidade de participantes foi pequena. Essa quantidade se deve ao fato do curto período em que o questionário ficou disponível, bem como nas empresas e pessoas que receberam o formulário. Outro fator importante que deve ser ressaltado é que por essa amplitude de respostas, não se pode estender o contexto dos resultados para o mercado internacional, tendo em vista que a maior parte das respostas foram provenientes de uma região específica do país.

No caso da seleção e extração de informação dos artigos científicos com objetivo de construir casos e então iniciar o povoamento da ontologia, não ocorreu da forma desejada, no sentido de que os artigos não relatavam de maneira detalhada as situações. Assim, poucas informações realmente relevantes eram retiradas para povoamento da ontologia e para a construção dos casos, que ocorria de forma manual e interpretativa, despendendo tempo e trabalho além do esperado.

Para desenvolvimento da ontologia, houve a necessidade de obtenção de maior número de instâncias para que se pudesse testar a melhor a utilização desta. Dessa forma, o trabalho ficou limitado pelo povoamento, que mais uma vez se tornou algo bastante difícil já que há a exigência de que projetos inteiros sejam inseridos. Dessa maneira, o povoamento inicial se deu apenas através dos artigos científicos o que também deixou a desejar no que diz respeito à riqueza das informações.

Na abordagem RBC e na ferramenta de acesso à ontologia existiu mais uma limitação, pois estas não foram testadas pela indústria e assim não puderam ser comprovadamente verificadas dentro da realidade industrial. Além disso, a abordagem RBC possuía em seu planejamento o seu teste em conjunto com outros algoritmos de similaridade, o que não foi possível por mera questão de tempo.

6.2 Trabalhos Futuros

Essa seção descreve algumas as possíveis opções para trabalhos futuros no que diz respeito a todas as contribuições dessa tese, desde os passos iniciais até os últimos. Esta seção apresenta tais possibilidades, de modo que o trabalho continue em constante evolução, tanto por este autor, como por outros possíveis interessados.

Um dos possíveis desenvolvimentos futuros é a ampliação do mapeamento sistemático, visando à ampliação do campo de análise com a utilização de mais trabalhos relacionados a partir das referências dos estudos primários, coletando informações inclusive de forma manual nas principais conferências de DDS e de Gerência de Conhecimento. Com isso, poderão surgir novos trabalhos com maiores expressividades no sentido de explorar e dissecar essa área de atuação. Além disso, novas análises poderão ser feitas a fim de explorar outros recursos a partir dos dados obtidos.

Uma possibilidade para aumentar a base de casos seria deixar o sistema online para que fosse possível um melhor acompanhamento da utilização e funcionamento geral da abordagem, assim, instituições poderiam utilizá-lo, verificar seus problemas e adicionar mais casos, de forma que os dados

das instituições fossem preservados, o intuito seria feedback sobre funcionamento da abordagem bem como o armazenamento de informação, consequentemente o aumento da base, com isso casos novos e reais poderiam ser obtidos, de forma que pudessem ajudar melhor esse usuário como outros também.

Para evolução da DKDOnto, conforme planejado, após o seu desenvolvimento inicial, as próximas iterações contemplam estudo e análise de termos de outras ontologias, principalmente as citadas nos estudos primários encontrados, de forma que possam ser utilizados de modo que esta abordagem possa se utilizar de melhores recursos. Outra opção para o desenvolvimento da ontologia em si, é ser validada através de ferramentas específicas, verificando seus relacionamentos de classes, axiomas e propriedades.

Para melhorar a abordagem como um todo outro ponto pode ser aprofundado que é a projeção de outras possibilidades para melhorar a integração do sistema RBC com a ontologia em si, para que a partir disso se possam utilizar, talvez de forma mais completa, os recursos que a *web* semântica oferece, como por exemplo, os casos e suas devidas soluções fizessem parte de classes na ontologia. Outra opção mais simples, será a realização de testes com outros algoritmos de similaridade no RBC, a fim de obter recomendações ainda mais próximas.

Outra possibilidade para futuros trabalhos é que a abordagem proposta por essa tese de doutorado produza automaticamente casos com base nas informações adicionadas na DKDOnto para serem colocados à disposição para possíveis problemas, esses casos seriam construídos baseados nas informações dos projetos e dos problemas enfrentados.

Por fim, novas avaliações devem ser feitas utilizando uma maior variedade de casos, como por exemplo, aumentando a quantidade de buscas e analisando seus resultados. Uma opção também interessante e que já foi planejada, mas não cumprida nessa iteração, é a avaliação dessa ferramenta dentro das organizações, para que estas utilizem na prática a ferramenta como forma de consultar seus problemas, e assim resultar em *feedbacks* mais interessantes.

REFERÊNCIAS

Aamodt, A., Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. (AI communications, v. 7, n. 1, p. 39-59.)

ABES. Mercado Brasileiro de Software. (2013). Disponível em <http://www.abessoftware.com.br/>. Acesso em Jan-14.

Aho A., Sethi R., Ullman J. (2008). Compiladores: Princípios, Técnicas e Ferramentas. (São Paulo: Pearson Addison – Wesley.)

Allen, James. (1995) Natural Language Understanding. The Benjamin/Cummings (Publishing Company Inc.)

Allen, T. J. (1977). Managing the Flow of Technology. (Cambridge, MA: MIT Press.)

Almeida, A., Barreiros, E., Saraiva, J., Soares, S. (2011). Mechanisms to Guide Empirical Studies in Software Engineering: A Systematic Mapping Study. In Proceedings of the International Conference on Evaluation & Assessment in Software Engineering (EASE) (Durham, England, Apr. 11-12).

Angioni, M., Sanna, R., Soro, A. (2005). Defining a Distributed Agile Methodology for an Open Source Scenario. Proceedings of the First International Conference on Open Source Systems, (Genova. p. 209-214.)

Apache Software Foundation. (2013). In: Apache Tomcat. [S.1]: The Apache Software Foundation, 2013. Disponível em: <http://tomcat.apache.org/>. Acesso em: 30 de ago. De 2013.

Arksey, H. O'Malley, L. (2005). Scoping studies; Towards a Methodological Framework. International Journal of Social Research Methodology. 8:19-32.

Assali, A. A., Lenne, D., Debray, B.: Case Retrieval in Ontology-Based CBR Systems. In:

Audy, J., Prikladnicki, R. (2007). Desenvolvimento Distribuído de Software: Desenvolvimento de software com equipes distribuídas. (Rio de Janeiro: Elsevier).

Ayoma, M. (1998). Web-based Agile Software Development, IEEE Software, Vol 15, No. 6.

Bannerman, P.L., Hossain, E. Jeffery, Ross. (2012). Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage? 45th Hawaii International Conference on System Sciences.

Battin, R. D, Crocker, R.., Kreidler, J., Subramanian, K. (2001). Leveraging resources in Global Software Development. (IEEE Software March / April: 70-77).

Beißel, S. (2011). Ontology-based case-based reasoning – development and evaluation of semantic similarity metrics for the reuse of project knowledge in natural language. Doctoral dissertation, University of Duisburg-Essen. (Gabler, Wiesbaden. 2011.)

Berners-Lee, J, Lassila, O. (2001). The semantic web, Scientific American.

Berners-Lee, T., Lassila, O., Hendler, J. (2001) “The semantic web.” Scientific American, 5:34–43.1

Bichindaritz, I., Montani, S. (2012). Report on the Eighteenth International Conference on Case-Based Reasoning, (AI Magazine 33. p 79-82.)

Borges, A.., Rocha, R.., Costa, C.., Tomaz, H., Soares, S., Meira, S. (2013). Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study. In Proceedings of the International Conference on Evaluation & Assessment in Software Engineering (EASE). (Porto de Galinhas, PE, Brasil.)

Brady, A., Menzies, T., El-Rawas, O., Kocaguneli, E., Keung, J. Case-Based Reasoning for Reducing Software Development Effort. Journal of Software Engineering & Applications, 2010, 3, 1005-1014.

Bray, T. et al. (eds.). Extensible Markup Language (XML) 1.0, 2.ed., 6 Oct 2000. Disponível em: <http://www.w3c.org/TR/REC-xml>. Acesso em: 15 mar 2001.

Brickley, D., Guha, R..V. (2000). Resource Description Framework (RDF) Schema specification 1.0 27 março 2000, 1.ed. Disponível em: <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>. Acesso em: 12 abr 2001.

Brooks, F. P. (1978). The Mythical Man-Month: Essays on Softw. 1st. Addison-Wesley Longman Publishing Co., Inc.

Bugmann, R. (1999). Protótipo de sistema de informação para o plantio de árvores frutíferas usando Raciocínio Baseado em Casos. 73f. Trabalho de Conclusão de Curso - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

Cambria, E. e White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research. IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE.

Carmel, E. (1999). Global Software Teams: Collaboration Across Borders and Time Zones. Prentice-Hall, EUA.

Carmel, E., Agarwal, R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development, IEEE Software, Vol. 18, No. 2.

Carmel, E., Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. J. of IEEE Software. 18, 2, 22–29. Disponível em <http://dx.doi.org/10.1109/52.914734>.

Carmel. E., Tjia, P. (2005). Offshorting Information Technology: Sourcing and Outsourcing to a Global Workfoce. Cambridge. 2005.

Cataldo, Marcelo., Bass, Matthew., Herbsleb, James., Bass, Len. (2006). Managing Complexity in Collaborative Software Development: On the Limits of Modularity. Workshop on Supporting the Social Side of Large-Scale Software Development, Conference on Computer Supported Cooperative Work (CSCW'06), Banff, Alberta, Canada.

Chi, Z. (1999) Statistical Properties of Probabilistic Context-Free Grammars. Computational Linguistics, vol. 25, nº 1.

Cleland, D., Ireland, L. (2002). Gerência de projetos. 1° edição. Rio de Janeiro: Reichmann & Affonso Editores, 324 p.

Cover, T. e Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, IEEE Information Theory Society, vol. 13, p. 21–27.

da Silva, F.Q.B., Costa, C., França, A.C.C., Prikladinicki, R. (2010) .Challenges and Solutions in Distributed Software Development Project Management: A Systematic Literature Review. International Conference on Global Software Engineering (ICGSE).

da Silva, F.Q.B., Rafael Prikladnicki, A. César C. França, Cleviton V. F. Monteiro, Catarina Costa and Rodrigo Rocha. (2011). An evidence-based model of distributed software development project management: results from a systematic mapping study. Journal of Software: Evolution and Process Special Issue: Special Issue on Global Software Engineering. Volume 24, Issue 6, pages 625–642.

Damian, D., Zowghi, D. (2002). An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. Proceedings of the 36th Hawaii International Conference on Systems Sciences (HICSS'03), IEEE

Dantas, Vanessa. (2003). Wide Work Web – Uma Metodologia para o Desenvolvimento de Aplicações Web num Cenário Global. Dissertação de Mestrado, Universidade Federal de Campina Grande, Campina Grande - PB, Brasil.

DeMiguel, J., Plaza, L., Díaz-Agudo, B.: ColibriCook. (2008). A CBR system for ontology-based recipe retrieval and adaption. In: Schaaf, M. (ed.): ECCBR 2008: The 9th European Conference on Case-Based Reasoning – Workshop Proceedings, pp. 199-208. Tharax Verlag, Hildesheim.

Dillon, T. S., Simmons, G. (2008). Semantic Web support for Open-source Software Development. In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS).

Domholdt E. (2005). Rehabilitation research: principles and applications. Missouri: Elsevier Saunders.

Dos Santos, Leonardo., L'erario, Alexandre. Gengiver, Elias C., Dos Santos, Andre., FABRI, José Augusto. (2014) . Improved communication in distributed agile software development. 9th Iberian Conference on Information Systems and Technologies (CISTI), p. 1.

Ebert, C.; De Neve, P. (2001). Surviving global software development. IEEE Software 18(2): 62-69.

Estler HC, Nordio Martin, Carlo A. Furia, Bertrand Meyer, Johannes Schneider. (2014). Agile vs. structured distributed software development: A case study and time zones affect software development? a case study on communication. In: 6th IEEE international conference on global software engineering, (ICGSE). IEEE, Helsinki, Finland, Empirical Software Engineering. Volume 19, Issue 5, pp 1197-1224.

Evaristo, J., Scudder, R., Desouza, R., Sato, O. (2004). A dimensional analysis of geographically distributed project teams: a case study. Journal of Engineering and Technology Management, 21 (3), 175-189.

Franconi, E. (2001). Description Logics for Natural Language Processing. In: Description Logics Handbook. Cambridge University Press. Cap.18.

Freitas, A. (2005). APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software. Faculdade de Informática – UFRS – RS- Brasil. Dissertação.

Freitas, F. (2003). Ontologias e a web semântica. In: Renata Vieira; Fernando Osório. (Org.). Anais do XXIII Congresso da Sociedade Brasileira de Computação Campinas: SBC, 2003. v. 8, p. 1-52.

Gamerman, D., Migon, H. (1993). Inferência estatística: uma abordagem integrada. Instituto de Matemática, Universidade Federal do Rio de Janeiro.

Gao, J., Chen, C., Toyoshima, Y., Leung, D. (1999). Engineering on the Internet for Global Software Production, IEEE Computer, Vol.32, No. 5.

Gonzalez, M. (2002). Recuperação de Informação e Processamento da Linguagem Natural. Exame de Qualificação para Doutorado. Faculdade de Informática, PUCRS, Porto Alegre.

Gonzalez, M., Lima, V. Recuperação de Informação e Processamento da Linguagem Natural. CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 23. , 2003, Campinas. Anais do III Jornada de Mini-Cursos de Inteligência Artificial, Volume III, Rio de Janeiro: [s.n.], 2003. p.347-395.

Google Web Toolkit.0 (2013) Disponível em: <http://gwtproject.org>. Acesso em: jun 2013.

Grails. (2013). Disponível em: <http://grails.org>. Acesso em: jun de 2013.

Groovy. (2013). Disponível: <http://groovy.codehaus.org>. Acesso em: jun de 2013.

Gruber, T. (1995). Toward Principles for the Design of Ontologies used for Knowledge Sharing. In formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers.

Guizzardi. G. (2000). Uma abordagem metodológica de desenvolvimento para e com reuso, baseada em ontologias formais de domínio. Dissertação de Mestrado. Universidade Federal do Espírito Santo.

Guthrie, L., Pustejovsky, J., Wilks, Y., Slator, B. The Role of Lexicons in Natural Language Processing. Communications of the ACM, v. 39, n.1, janeiro 1996. p.63-72.

Herblesb. J. (2007). Global Software Engineering: The Future of Socio-technical Coordination. IEEE Computer Science. P188-198.

Herbsleb, J., Moitra, D. (2001). Guest editors introduction: global software development. IEEE Software.

Herbsleb, J., Mockus, A., Inholt T., Grinter, R. (2001). An empirical study of speed and communication in globally-distributed software development. IEEE Transactions on Software Engineering, 29:6, pp. 481-494.

Herbsleb, J., Moitra, D. (2001). Guest editors' introduction: global software development. IEEE Software.

Hibernate. (2013). Disponível em: <http://hibernate.org>. Acesso em: jun 2013.

Horridge, M. (2009). A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. Disponível em: <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial>. Acesso em: 5 de outubro de 2013.

Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., Harmelen , F., Klein, M., Staab, S., Studer, R., Motta, E. 2000. The Ontology Inference Layer OIL. Disponível em: <http://www.ontoknowledge.org/oil/TR/oil.long.html>. Acesso em: jun 2013.

IEEE. (1997). Standard for developing software life cycle processes. EEE Computing Society. p. 96. may 1997. Disponível em: <http://standards.ieee.org/catalog/olis/archse.html.[retrieved:06, 2013]>.

J2EE. (2013). JAVA Enterprise Edition. Disponível em: <http://oracle.com/technetwork/java/javaee/overview/index.html>. Aceso em: 06 de jun de 2013.

JABREF. (2013). Disponível em: < http://jabref.sourceforge.net/>. Acesso em: mar de 2010.

Java Oracle. (2013). In: Oracle. [S.1]: Oracle. Disponível em: <http://www.oracle. com/br/technologies/java/overview/index.html>. Acesso em: 30 de ago de 2013.

Java Persistence API. (2013). In: Oracle. [S.1]: Oracle. Disponível em: <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>. Acesso em: 30 de ago. de 2013.

Jimenez, M., Piattini, M. e Vizcaino, A. (2009). Challenges and Improvements in Distributed Software development: A Systematic Review. Advances in Software Engineering, Article ID 710971, pp. 1-14.

Junior, Ivaldir H de Farias., Azevedo, Ryan Ribeiro., Moura, Hermano e Silva, DS Martins. (2012). Elicitation of Communication Inherent Risks in Distributed Software Development. Global Software Engineering Workshops (ICGSEW). IEEE.

Karolak, D. (1998). Global software development – managing virtual teams and environments, IEEE Computer Society, EUA.

Khan, Siffat Ullah. Niazi, Mahmood e Ahmad, Rashid. (2011). Barriers in the selection of offshore software development outsourcing vendors: An exploratory study using a systematic literature review. Journal Information and Software Technology, Volume 53 Issue 7, July, Pp 693-706.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Joint Technical Report TR/SE-0401 (Keele) – 0400011T.1 (NICTA), Software Engineering Group – Departmen of Computer Science - Keele University and Empirical Software Engineering – National ICT Autralia Ltd, Keele/Sta s-UK and Eversleigh-Autralia.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical Report. Keele University at Staffordshire.

Kitchenham, B. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report. Keele University at Staffordshire and University of Durham at Durham.

Kitchenham, B., Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S. (2008). Systematic literature reviews in software engineering - A systematic literature review. J. Of Information and Software Technology. 51, 1, 7-15.

Koivunen, M-R., Muller, E. W3c semantic web activity. In Semantic Web Kick-Off in Finland - Vision, Technologies, Research, and Applications. 2001.

Kolodner, J. (1993). Case Based Learning. Boston, EUA: Kluwer Academic Publishers.

Komi-Sirvo, S; Tihinen M. (2005). Lessons Learned by Participants of Distributed Software Development. Journal Knowledge and Process Management, vol. 12 no 2 p. 108–122.

Kowalski, M., Klüpfel, H., Zelewski, S., Bergenrodt, D., Saur, A. Integration of Case-Based and Ontology-Based Reasoning for the Intelligent Reuse of Project-Related Knowledge. pp.289-299. Efficiency and Logistics, Springer Berlin Heidelberg. 2013.

Krasner, G. E. e Pope, S. T. (1988). A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. ParcPlace Systems, Inc, Mountain View, CA, USA.

Leake D. B. (1996). Case-Based Reasoning: Experiences, Lessons and Future Directions. MIT Press Cambridge, MA, USA.

Liddy, E. D. (1998). Enhanced Text Retrieval Using Natural Language Processing. In: Bulletin of the American Society for Information Science, v. 24, n. 4.

Lima, T. Entrelaçamentos semânticos modelando a integração da informação na Web Semântica. São Carlos, 2001. 117f. Tese (Doutorado em Ciência da Computação) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

Linde K, Willich SN. (2003). How objective are systematic reviews? Differences between reviews on complementary medicine. J R Soc Med. 2003; 96:17-22.

Lopes, L., Prikladnicki, R., Majdenbaum, A., Audy, J. (2003). Uma proposta para processo de requisitos em ambientes de desenvolvimento distribuído de software. In: 6th WER, Piracicaba. Brasil. p. 329-342.

Lopes, Leandro; Prikladnicki, R., Audy, J. L. N. (2004). Distributed Requirements Specification: Minimizing the Effect of Geographic Dispersion, In: ICEIS 2004, Cidade do Porto,Portugal.

M. Fernandez, A. Gomez-Perez, N. Juristo, (1997). Methontology: from ontological art towards ontological engineering. Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, US, pp.33–40.

Maalej, W., Happel, HJ. (2008). A Lightweight Approach for Knowledge Sharing in Distributed Software Teams. International Conference on Practical Aspects of Knowledge Management. Berlin.

Manola, F., Miller, E. Resource Description Framework (RDF) model and syntax specification.1.0, 10 Feb 2004. (Recomendação do W3C). Disponível em: <http://www.w3c.org/TR/REC-rdf-syntax>. Acesso em: 12 mai 2008.

Mariano, F., Assuncion, G. (2002). Overview and Analysis of methodologies for building ontologies. The Knowledge Engineering Review. Cambridge, v. 17. n. 2, p. 129-156.

Marneffe, M., Manning, C. (2008). Stanford Typed Dependencies Manual. In: The Stanford Parser [S.1]. Disponível em: <http://nlp.stanford.edu/software/ dependencies_manual.pdf>. Acesso em: 01 de ago. de 2013.

Marques, A., Carvalho, J., Rodrigues, R., Conte, T., Prikladnicki, R., Marczak, S. An Ontology for Task Allocation to Teams in Distributed Software Development International Conference on Global Software Engineering (ICGSE). p21-30. Bari, Itália. 2013.

Mattos, M., Simões, P., Farias, R. (2007). A metodologia Methondology na construção de Ontologias. RIC Vol. 5, No 1.

Melchiors, C. (1999). Raciocínio Baseado em Casos Aplicado ao Gerenciamento de Falhas em Redes de Computadores. 122f. Dissertação de Mestrado. Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Mendes, William Corrêa. (2013). Arquitetura Baseada em Ontologias de um Agente RBC. Dissertação de Mestrado. Universidade Federal do Maranhão.

Menegazzo, C. T. (2001). Raciocínio Baseado em Casos Aplicado a Diversos Domínios de Problema. 150f. Dissertação de Mestrado. Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Menezes, P. B. (2000). Linguagens Formais e Autômatos. Série Livros Didáticos, 3ª Ed. Porto Alegre: Editora Sagra Luzzatto.

Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 564–571. (Springer, Heidelberg. 2009.)

Meyer, B. (2006). The unspoken revolution in software engineering. IEEE Computer.

Mockus, A., Weiss, D. M. (2001). Globalization by Chunking: A Quantitative Approach, IEEE Software, Vol. 18. No. 2, March-April.

Mustapha, N. B., Zghal, H.B., Aufare, M., Ghezala, H. Semantic Search Using Modular Ontology Learning and Case-Based Reasoning. (2010). Proceedings of the 2010 EDBT/ICDT Workshops, article 3, ACM New York, NY, USA.

NetBeans IDE – Oracle. (2013). NetBeans IDE. [S.1]: Oracle Foundation, 2013. Disponível em: <https://netbeans.org/>. Acesso em: 30 de ago de 2013.

Nordio M, Estler HC, Meyer B, Tschannen J, Ghezzi C, Di Nitto E (2011). How do distribution

Noy, N. F. and McGuiness, D. L. (2003). “Ontology Development 101: A Guide to Create Your Firts Ontology”. Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford University. 25 p.

Object-relational mapping. (2013). In: What is Object/Relational Mapping. [S.1]: Hibernate, 2013. Disponível em: <http://www.hibernate.org/about/orm>. Acesso em: 30 de ago. de 2013.

Olson, J. S., Olson, G. M. (2004). Culture Surprises in Remote Software Development Teams. ACM Queue, New York, v. 1, n. 9, pp 52-59.

Orsoletta, Roni A. Dall e Prikladnicki, Rafael. (2013). Lições aprendidas com o uso da emulação de proximidade física no Desenvolvimento Distribuído de Software. Proceedings of the X Brazilian Symposium in Collaborative Systems.

OWL API. (2013). Disponível em: http://owlapi.sourceforge.net, Acesso em: jun de 2013.

OWL. (2009). Web ontology language overview. Disponível em:: <http://www.w3.org/TR/owl-features>. Acesso em: jun de 2013.

OWL. Web ontology language overview - w3c. Disponível: http://www.w3.org/TR/owl-features/ Acesso em: jul 2008.

Pantazi, S., Arocha, J., and Moehr, J. (2004). Case-based medical informatics. BMC Medical Informatics and Decision Making.

Parviainen, Päivi e Tihinen, Maarit. (2011). Knowledge-related challenges and solutions in GSD. Expert Systems.

Pereira. (2013). Processamento de Linguagem Natural. Disponível em: <http://www.ime.usp.br/~Slago/pl-12.pdf>. Acesso em jan de 2014.

Petersen, K. e Wohlin, C. (2009). Context in Industrial Software Engineering Research. Proceedings 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 401-404, Orlando, USA, October.

PostgreSQL. (2013). PostgreSQL. [S.1]: The PostgreSQL Global Development Group, 2013. Disponível em: <http://www.postgresql.org/>. Acesso em: 30 de ago de 2013.

Prikladnicki, R. (2003). MuNDDoS - Um modelo de referência para desenvolvimento distribuído de software. Master Thesis, Pontifical University of Rio Grande do Sul (PUC/RS) at Porto Alegre.

Prikladnicki, R., Audy, J. (2004). Munddos: Um Modelo de Referência para Desenvolvimento Distribuído de Software. 18o Simpósio Brasileiro de Engenharia de Software.

Prikladnicki, R., Lopes, L., Audy, J., Evaristo, R. (2004). Desenvolvimento Distribuído de Software: um Modelo de Classificação dos Níveis de Dispersão dos Stakeholders, I Simpósio Brasileiro de Sistemas de Informação, Porto Alegre.

Prikladnicki, Rafael. (2009). Padrões de Evolução na Prática de Desenvolvimento Distribuído de Software em Ambientes de Internal Offshoring: Um Modelo de Capacidade. Tese de Doutorado. PUCRS.

Prikladnicki, Rafael e Carmel, Erran. (2013). Is time-zone proximity an advantage for software development? The case of the Brazilian IT industry. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press.

Protégé. (2009). Protégé ontology editor. Online. Disponível em: <http://protege.stanford.edu/doc/users.html>. Acesso em: jun de 2013.

Ramasubbu, N., Cataldo, M., Balan, R. K. and Herbsleb, J. (2011). Configuring Global Software Teams: A Multi-Company Analysis of Productivity, Quality, and Profits. In Proceedings, International Conference on Software Engineering, Honolulu, HI, pp. 261-270.

Rautenberg, S., Todesco, J., Guathier, F. (2009). Processo de desenvolvimento de ontologias: uma proposta e uma ferramenta. Universidade Estaduao do Centro-Oeste do Paraná (Unicentro).

Roberto, E., Richard, S. (2000). Geographically distributed project teams: a dimensional analysis. In: HICSS, 2000, Havaí. Proceedings, EUA, p. 1-15.

Rocha, R. Arcoverde D. Brito, R. Aroxa, B. Costa, C. Silva, F. Q. B. Albuquerque, J. Meira, S. R. L. (2008). “Uma Experiência na Adaptação do RUP em Pequenas Equipes de Desenvolvimento Distribuído”. II Workshop de Desenvolvimento Distribuído de Software (SBES-SBBD). pp. 87-96.

Rocha, R., Costa, C., Rodrigues, C., Azevedo, R., Farias, I., Prikladnicki, R., Meira, S. (2011). Collaboration Models in Distributed Software Development: a Systematic Review. CLEI Electronic Journal, v. 1, p. 1-12.

Roth-Berghofer, T., Adrian, B., Dengel, A. (2010) Case Acquisition from Text: Ontology-Based Information Extraction with SCOOBIE for myCBR. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 451–464. Springer, Heidelberg..

Samovar, L., Porter, R. (2004). Communication Between Cultures. Belmont (California), Wadsworth – Thomson Learning.

Saraiva, J., Barreiros, E., Almeida, A., Lima, F., Alencar, A., Lima, G., Soares, S., Castor, F. (2012). Aspect-Oriented Software Maintenance Metrics: A Systematic Mapping Study. In Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE) (Ciudad Real, Spain, May 14-15, 2012). DOI= http://dx.doi.org/10.1049/ic.2012.0033.

Scapini, I., Relações entre Itens Lexicais. In: Poersch, J., Wertheimer, A., Ouro, M., Ludwig, E., Scapini, I., Becker, B. (1995). Fundamentos de um Dicionário Remissivo. 1o Encontro do CELSUL, Florianópolis. Anais, V. 1, p.393-429.

Scatalon, L.; Garcia, R., Correia, R. (2010). Introdução a ontologias e suas aplicações. Omnia Exatas, v.3, n.2, p.19-28.

Smite, D., Borzovs, J. (2009). New Forms of Work in the Light of Globalization in Software Development. Infoconomic for Distributed Business and Decision-Making Enviroments: Creating Information System Ecology. Business Science Reference – Blekinge Intitute of Technology, p346.

Smith, Barry e Katherine Munn. (2008). Applied Ontology. An Introduction. Frankfurt/Paris/Lancaster/New Brunswick: Ontos Verlag. ISBN 978-3-938793-98-5.

Smith, M., Welty, C., McGuinness, D. (2003). Web Ontology Language (OWL) Guide Version 1.0. Disponível em: < http://www.w3.org/TR/2003/WD-owl-guide- 20030210/> .

Souza, M. (2007). Análise do Processo de Desenvolvimento de Software para o Desenvolvimento Distribuído de Software. Dissertação. Pelotas.

SPARQL. (2013). Query Language for RDF. Disponível em: <http://w3.org/TR/rdf-sparql-query>, Acesso em: jun de 2013.

SWEBOK. (2004). Guide to the Software Engineering Body of Knowledge. A project of the IEEE Computer Society Professional Practices Committee. Disponível em <http://www.swebok.org>. Acesso em: dez de 2012.

Teixeira, J. (2007). Modelagem de um Ambiente de Desenvolvimento Distribuído de Software baseado em Workflow. Graduação em Ciência da Computação. UFPEL.

The jQuery Foundation. (2013). Jquery. [S.1]: The jQuery Foundation, 2013. Disponível em: < http://jquery.com/>. Acesso em: 30 de ago. de 2013.

The Standish Group. (2013). Chaos Manifesto 2013. Disponível em: <http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf>. Acesso em: jan de 2014.

The Stanford Parser: A statistical parser. (2013). The Stanford Natural Language Processing Group. [S.1]: Stanford University, 2013. Disponível em: <http://nlp.stanford.edu/software/lex-parser.shtml>. Acesso em: 14 de jul. de 2013.

Thé, M. (2001). Raciocínio Baseado em Casos Uma Abordagem Fuzzy para Diagnóstico Nutricional. 170f. Tese de Doutorado. Universidade Federal de Santa Catarina, Florianópolis – SC.

Travassos, G., Biolchini J. (2007). Revisões Sistemáticas Aplicadas a Engenharia de Software. In: XXI SBES - Brazilian Symposium on Software Engineering, 2007, João Pessoa. SBES 2007 - XXI Simpósio Brasileiro de Engenharia de Software.

Urnau, Eduardo., Kipper, Liane Mahlmannn e Frozza, Rejane. (2014). Desenvolvimento de um sistema de apoio à decisão com a técnica de raciocínio baseado em casos. Perspectivas em Ciência da Informação, v.19, n.4, p.118-135, out./dez.

Uschold, M. e Gruninger, M. (1996). Ontologies: Principles, methods and applications. Knowledge Engineering Review, v. 11, p. 93–136.

Vieira, R. e Lima, V. (2001). Lingüística Computacional: Princípios e Aplicações. In: JAIA, SBC, Fortaleza, Brasil.

Wangenheim C. e Wangenheim, A. (2003). Raciocínio Baseado em Casos. Barueri: Editora Manole Ltda.

Wangenheim, A. (2013). Raciocínio Baseado em Casos – Visão Geral, In: The Cyclops Project. [S.1]. Disponível em: <http://www.inf.ufsc.br/~casos/CBR1-visaogeral-pg.pdf>. Acesso em: 16 de ago. de 2013.

Watson, I. (1997). Applying Case-Based Reasoning. Techniques for Enterprise Systems. Morgan Kaufmann, California.

Watson, I.; Marir, F. (1994). Case-based reasoning: A review. Knowledge Engineering Review, v. 9, n. 4, p. 327-354.

Webb G., Butler S. e Newlands. D. On detecting differences between groups. The ninth ACM SIGKDD international conference on Knowledge discovery and data mining. Pages 256-265. 2003.

Weber, L. e Klein, P.A.T. (2003). Aplicação da Lógica Fuzzy em software e hardware. 1. ed. Canoas: Ulbra.

Wilks, Y., Slator, B., Guthrie, L. (1996). Electric Words: Dictionaries, Computers and Meanings. Cambridge: The MIT Press. 289 p. Magazine.[retrieved: 06, 2013].

Wongthongtham, P., Chang, E., Dillon, T. S., Sommerville, I. (2007). Ontology-based Multi-site Software Development Methodology and Tools. J. of Systems Architecture. 52, 11 (Nov. 2006), p. 640–653. Disponível em: <http://dx.doi.org/10.1016/j.sysarc.2006.06.008>.

A

Casos Utilizados para Avaliação da Abordagem

Caso	Solução	Efetividade
Performing documentation and validation of requirements in project with national distance, high levels of temporal and geographical dispersion.	Making use of StoryBoards.	Allowed the sharing of understanding of the requirements among distributed teams. Showed decreased amount of duplicate requirements and not corresponding to the modules developed. Occurred mostly use cases ready to move to the design phase.
Choose of a process to distributed software development to an open source software development company.	Utilization of an agile method based on SCRUM.	It was possible to use agile method but with some problems of communication and organization, which were more or less resolved with the use of other tools such as forums.
Communication between product owner and team project in a distributed project using an adaptation of SCRUM.	Utilization of a forum, email or asynchronous communication tools.	
Simulating the Daily Scrum Meeting in distributed project using adaptation of the SCRUM.	Utilization of forum and mailing list.	
Performing synchronous meeting with the product owner to show the results obtained at the end of each Sprint in a distributed project using adaptation of the SCRUM.	Making use of MSN Messenger or Skype.	
Choose of process to distributed software development for academic project in an open-source software factory.	Implementation of the development process MORPHOS that unifies concepts of CSCW, DXP and RUP.	There were some problems and sometimes heroic efforts were needed to deadlines being met, but the biggest part of the problems came from the members in the project
Schedule and hold meetings with team members in different geographical locations.	To schedule meetings use Google Calendar and to make meetings use Skype or MSN Messenger or Google Talk with audio and video conversations.	Sometimes the communication was not very good and the members did not seem to work in groups, so sometimes it was necessary to make physical meetings.
Perception of distance between contributors in the same project, located in Brazil and India.	Using the PDI model to quantify the relative perceived distance between the teams.	
Choose of development software process for a small team using distributed software development.	Adaptation of the RUP development process.	It had many problems, but with adaptations this could be used.
Pair programming with geographical distance.	Utilization of E-mails and Messenger.	It showed a lot of problems
Communication among small team members in distributed project using a RUP adaptation.	Utilization of Messenger.	Presented several problems such as the impossibility of sharing code.
Finding an infrastructure to support distributed development projects based on components.	Adoption of a service of shared repository and distributed software components, the X-CORE.	It was very effective and easy to use, also removed the utilization necessity of another support tools as CVS and Bugzilla.
Establishment of inter organizational electronic contract negotiation electronic services and the establishment of electronic contracts, involving Web services and PN, based on concepts of LP software.	Utilization of the FeatureContract environment.	
Collaborative software development process for geographically distributed teams.	Utilization of the Contribution Processes, IDE for code development and documentation for collaborative project environments.	
Establishment of inter organizational electronic contract.	Making use of ontologies.	

Making allocation of globally distributed development teams for implementation activities of software modules.	Using recommendation based on non-technical characteristics of the teams and the dependencies between software modules.	
Software development process with large and distributed teams.	Using adapted concepts of SCRUM.	Despite the fact of presenting some difficulties with some adaptations, the result was good.
Making selection of distributed teams technically qualified to implement project software modules of software product lines.	Using fuzzy logic.	
Lack of communication, coordination and collaboration in software development teams separated geographically.	Using tools, methods and technologies to simulate proximity in the development environment. Ex: Video chat tools, Messenger, Skype audio conversations.	It showed effectiveness for teams with small differences in time zones.
Reduce loss of productivity and decreasing of teamwork because team members work in a distributed manner.	Scheduling of weekly in-person meetings.	Even after the scheduling of weekly meetings, some of the members did not attend the meetings and still maintained low productivity.
Development of a project in which members were in different cities and only one was in the same city of the customer.	Process control by those who were located in the city of the customer, use of a support tool to delegate tasks, use of software version control for programmers and designers, and individual meetings via google talk.	Everything occurred as expected and all the challenges were identical to those that could happen to a local team.
Communication regarding meetings and discussions between project members. The development team was composed by four developers who communicated remotely.	We used email, Google talk, Skype and, less frequently, phone.	In order to conduct meetings for planning, monitoring activities and definitions of what was being developed, these tools (google talk, skype etc) were efficient. We had a bit of trouble when it was necessary to carry day-do-day meetings, ask questions about the code, solve impasses, configure the system, etc.
There is a need to manage the schedule of activities performed by a team of developers distributed in different cities.	The usage of online and shared tools among team members, for example, google calendar and google docs. Furthermore, conducting online meetings, via Skype or Gtalk, and in-person monthly meetings.	The use of tools to assist the monitoring of project activities for all its members was essential. Since the tools are shared online, all team members were able to monitor what activities the members were executing and what tasks should be performed in the future. Nevertheless, monthly meetings were needed to complement the development and clarify any problems regarding the understanding of the project requirements.
During the execution of a project, there is a lack in communication between developers, analysts, managers, managers, clients and users; which may cause loss of information, lack of accountability and so on.	Installation and execution of a plan for managing communications.	The information exchanged began to be administered by managers, resulting in more reliable informations, compliance of the deadlines and assignment of responsibilities for each type of provided information.
The team members were located in different geographic regions (cities / countries) and were working on the same software component. In this context, there was the need to schedule and conduct meetings with team members that were located around the world.	Schedule meetings using the Google Calendar and conduct meetings via video conferencing HALO (HP proprietary) tools.	In general, the applied solution was effective. However, there were specific cases in which it was necessary to carry presencias meetings, mainly to optimize the process of resolving faults.
Failure to comply with deadlines, i.e., delays in most deliveries agreed between the development team and the company's business crew .	Adoption of Basecamp project management as the issue tracker tool.	More deadlines were met primarily because fewer tasks were required by the business crew as never before, once they were able to check that the development team was dealing with excessive activities.
When working in a distributed way, one of the most important things to consider is how to control the versions of source code. I noticed that, in the context of Websites and portals, what was developed by me was often undone by another developer when a	We established that all of the source code would be administered by a version control software, such as the Tortoise SVN, GIT or CVS. By using these tools with a proper configuration it became easier to control the versions and avoid	The source code is administered by version controllers such as GIT or SVN, making the process of merging changes of distributed developers do not conflict.

"commit" overwrote the changes.	loss of work.	
The development of an online training platform for an educational institution.	The usage of online tools for project tracking and the agile methodology SCRUM.	Although it has helped in controlling and monitoring tasks, we still had problems regarding deadlines and communication among students.
We had problems related to the synchronization of working hours because part of the team works in a different time zone with 4 hours difference.	We define what we call "core hours", which is the period of time of intersection of the entire team. During this period, all team members try to stay dedicated to the project.	This increased interaction between all staff (local and remote), which helped not only to improve communication, but also in the formation of pairs (pair programming) remotely.
Difficulties in requirements elicitation of a project in the area of security involving four actors that were located at four different locations in the country.	In-person meetings in order to reduce the working hours as well as friction and rework in future software releases.	Average efficacy. Although it has reduced the required time, the cost was very high (8 persons * 3 days).
The biggest problem for project teams of various nationalities regards communication.	In this case, we established a common language (English). Although It seems a simple task, some cultures tend to not accept this change.	
Business processes are not mapped, which incurs a systemic solution that sometimes does not meet the requirements or even creates new problems.	Assignment of part of the staff to start the task of mapping processes using BPM tools.	A systemic solution is more aligned with reality, which aids the developers and all the people involved in the process to become aware of the workflow.
Lack of standardization of artifacts used in the context of the development cycle of geographically distributed teams.	Assess the need for standardization of common software development process between sites. In other words, evaluate the need to reach consensus on the processes and methods used by teams, which are common to all groups.	The solution aims to minimize the effort for integration of artifacts and communication between team members.
Difficulty in performing the "Daily Scrum" because part of the team works on different shifts or in different locations.	Perform the Daily Scrum during two times a day, the meetings in the morning and afternoon were used to cover the members working in different schedules; and perform Daily Scrum via SKYPE (using text only) for those who were at different locations.	
Difficulties for integrating the project team because of the geographical distance.	We started to use communication tools, which allowed us to conduct daily meetings and project monitoring.	The project progressed more effectively, the risks could be mitigated and the team has obtained a higher rate of productivity.
During the project, people needed to work in a distributed manner and carry meetings on specific dates to present the results.The definitions of activities and allocations were made in these meetings. However, other needs arose later. Hence, we faced the difficulty of organizing the new tasks.	It was used e-mails and tools such as Google Talk. For collaborative writing of documents, it was used email services. Thereby, when someone finishes a task, he alerts another team member in order to continue the tasks.	Once a tool for collaborative documentation was not used, there was a case that information was lost because one of the authors added new content, overwriting a previous content that should have been kept.
In the context of a project to develop an academic software, using the DDS environment, together with the agile project management model (SCRUM). The team was small (5 people) and each one had their roles well defined, i.e. a specialist for each function. This fact made it difficult the adoption of the SCRUM model, since the model requires a concise and coherent team, composed of multidisciplinary people. The tasks were divided equally for each member to try to equalize the job efforts. We had a chart of single units (tasks) to be Implemented and the team needed to complete it. However, in most of the times, the required time to complete the tasks had to be renewed, leaving the responsible for the task attached to it, since he was unable to	We tried create a kind of assembly line of software: each team member had a certain time to complete their task and, at the end of the term, if the task was not completed, the member would have to prepare a short report stating the main obstacles to complete delivery of the task. After that, the task was allocated to another team member.	We increased productivity on the development of the software, since we adopted a rotation of uncompleted tasks for other team members, i.e., if a member could not finish it or at least come close to completing it, another member was appointed to finish it.

reach a conclusion.		
During the development of a set of electronic games or digital games, the development teams were located in one state and the customers in another. The main problem was tackled during the validation of the game interface. The requirements of the games were well defined and did not led to further problems. On the other hand, the implementation of the game interface was difficult, even though we had well defined requirements. The problem was the following: how to define the game interface, since it was developed by a local team of interface designers that communicated with another loca team of developers. Validate and develop interfaces without changing the requirements set by customers (who were located in Pernambuco) was a problem.	Our solution was to introduce prototype interfaces to customers prior to the implementation of the interfaces. Since the prototypes did not alter the requirements, after the prototypes were validated the development and interface teams published interactive videos as each requirement was contemplated regarding the interface and thus the customers were able to validate each step at the time. If the customer did not like the interface of a given iteration, the new requirements were also used for the next iterations of the interface validation.The communication was carried using skype and youtube was used for the publication of prototypes. We also made available a particular requirement and its interface to be played by the customers. However, before customers experience each requirement and its interface, a local test was performed with local users and, after that, the validation of the client was carried.This idea saved development time and also mitigated risks to build a new interface for all developed modules for the games. The games were validated and the adopted solution has led to the success of the project as a whole.	The effectiveness was 100%. The risks of failure were mitigated, games were developed on schedule, the interface designers have worked without pressure, and functional and non-functional requirements are covered. Using skype, youtube and publishing the developed modules using web platforms for testing represents a viable solution in this type of distributed approach which involves national companies. The solution could also work for projects with different time zones.
Lack of knowledge of the company's board of development regarding the affairs and results of the development team.	Scheduling monthly in-person meetings with the management staff to ensure better understanding of the development team results.	
The project was developed with only one developer located in another workspace.	The local development team remained motivated during the project. However, the developer that worked in another location was unmotivated.	Daily meetings via skype and technological help via gtalk.
Absence of frequent communication.	Insertion of agile practices (daily meeting, review, retrospectives) using video conferencing tools.	It was possible to check the improvement in communication after the adoption of agile practices. However, we still see that the absence of in-person communication hindered the understanding of some activities.
Absence of an online System Version Control Software (CVS) that is used by distributed developers in different cities.	The usage of an online CVS system. A customized solution was used on a dedicated server designed for versioning the source code of the application.	The problems related to code sharing and code update were resolved.
Payment to employees occurred differently to conventional governed by CLT form. The value of the portfolio was much lower, for example R$ 850.00, while PPR was relatively fair, for example R$ 2,000.00, but paid for two months. As a consequence, there was a high turnover of employees, undermining the progress of projects.	We started hiring in accordance with the CLT standards.	The decrease in turnover enabled a faster accomplishment of the tasks along the execution of the projects and also reduced the expenses regarding training and qualification.
The distribution of tasks not observed geographical distribution and habits of the staff, So it caused problems for the time zone and decreased cooperation. Even in the same country members working in different shift.	Monitoring of the production schedules of the members (Day / Night) and redistribution of people between the teams to match schedules.	Very good. Increased teamwork and decrease problems of failures in software.
Bottlenecks in the production process of	The usage of the Jenkins (formerly	The bottleneck was eliminated because what was

some business systems of the companies.	Hudson) integration server.	once a task (build generation, deployment and execution of regression tests) of a developer became an automated process performed by the tool.
There was Incompatibility between files due to the character map. Some software and / or operating systems generate maps of different characters, which can cause error in files (ASCII, unicode, unicode not boom).	Establish a brief documentation of code specifications in order to standardize these problems.	A detailed documentation with technical specifications.
Implementation of a software prototype for control via RFID (wireless) for doctors and hospital instruments.	As the student was physically at the company, we established weekly or biweekly in-person meetings to establish goals. Outside the company, the student was free to schedule his own working hours.	Even though it generated satisfaction for the programmers, it also led to a certain level of complacency. Once we had only one programmer, the delivery deadline was increased.
Clear view of the remote team. Sometimes we needed to contact the remote people and did not know what was happening on the other side. It is normal for people to get up from the chair for any reason and get in touch using online chat tools sometimes is not enough.	We put a monitor in order to monitor both environments and we can easily see what the other teams are working on.	It became easier to keep in touch with the other teams because now it is easier to find someone and make fast contact using the microphone of the camera.
Difficulty of setting GUIs according to the country in which the product will be used.	Adjustments on the architecture to support intercionalização, and meetings via video conference to assess the GUI.	The results were excellent. All issues related to the culture of the users in the use of graphical interfaces were resolved.
The documentation of the use cases is outdated or nonexistent due to flaws in the initial project settings.	Identification of features that are not covered by the documentation and the process of decision making.	The development team has now a valid referential regarding what the system executes and the users may now solve their doubts by their own.
Failure to define the interface between the components that will be developed by different sites.	For the code components that have been developed in more than one site, it was necessary to perform activities to define the communication protocol between the modules of the component among the involved sites.	
Difficult to contact members who were distributed during working hours.	Usage of skype and cellphone during all working hours.	Sometimes the member still could not be found.
Difficulty of mitigating risks and completing activities.	We tried to minimize problems through daily meetings using communication tools.	Complex design, difficulties to communicate to clients and different levels of teams' knowledge. It was difficult to carry out the project.
The need for choosing a design methodology for development and monitoring of distributed projects.	The usage of a tool called XPlanner for sofware development based on agile teams (mainly using XP).	The tool itself helped in monitoring but did not contribute to the collaboration of the team, mainly because the tool functionalities were restricted.
Definition of activities and schedule for a distributed team.	Use of the planning poker on in-person meetings to define the activities and schedule.	
The project developers worked together but the designer was in another location.	Preparing a development environment and basic training on version control for the designer.	The communication was complicated. To overcome this problem, we performed meetings to define new sprints. Besides, the incorrect use of version control led to rework.
Management of activities. The project in question used the Scrum agile method and had something around 45 members, divided into five teams. One problem was to define the time required to perform certain tasks and use this information to make an adequate distribution of tasks between the teams.	To manage and distribute the activities Mantis tool was used. To set the sizes of the activities the poker tool was used. The scrum masters of each teams met to define the macro activities and, subsequently, each team met to divide the macro activities into smaller activities, also making use of the planning poker technique.	The activities began to be better distributed. Besides, we had greater control of the activities that were being carried out and what still needed to be done.
Absence of planning tools adopted during the DDS project.	Selection of Web 2.0 tools at the beginning of the project taking into account the experience of each distributed team.	It was noticed that there was a very good progress with respect to the use of these tools. The communication was clearer and more effective. However, there are some situations that can not be predicted at the beginning of the project and requires

		an in-person intervention by the manager or project leader.
The usage of a process for software development in the context of a project in which developer teams and stakeholders were located in different cities.	Adapting agile practices and Scrum methodologies based on the use of online project management tools.	Despite helping the management of the project, there was still the need for monthly in-person meetings. If there was an online tool that could manage the entire development process, from requirements elicitation until management of code, its use would facilitate the communication and understanding of the team members.
Standardization of code. Failure in monitoring coding standard and "vices" were appearing on each team. The problem was evident in the integration phase, generating rework due the adequacy of codes of different teams.	Using tools to monitor the standard of coding and use of an agile method based on SCRUM.	Very good. Increased productivity and reduced integration failures.
Increasing on the number of bugs of the system in production.	Code Review / Peer programming on a weekday.	Significantly decreasing on the occurrence of bugs since the adoption of the practice. It was discovered that the bugs were generated by the inexperience of some programmers that began to work in pairs and have their code reviewed by developers with greater expertise.
Difficulties to configure projects in heterogeneous development environment since part of the developers uses linux and the other part uses Windows.	We did not establish a solution.	Homogenize environments, regardless of Operating System.
Development of an interactive digital whiteboard with low cost.	Weekly meetings for checking and setting new goals for the scholarship holders (3 + 1 volunteer). On the other days, the monitoring took the online approach (msn, gtalk, email, facebook).	It was good, we managed to reach the goal before the stipulated deadline. The weekly monitoring forces the scholarship holder to develop something to present each week, providing a little more than he could in a monitoring process with greater freedom.
Integration between staff and environments. When a team is formed with members from different localities, we often do not personally know all the staff and different working environments. This phenomenon complicates the understanding of problems and solutions made by each part of the team and create a natural barrier between the local and remote people.	We make the rotation between the team where one member of each office changes place for 6 weeks. Thereby, a person who was working in the office of the country A, will stay in the office of country B for a period and vice versa. When the members returned to the office of origin, we rotate two more members until the whole team knows each other.	This approach increased the closeness between members, facilitating the exposure of problems and fluency of communication. This strategy also improved the knowledge of the remote desktop and dissimilar solutions taken related to the workplace.
Problems on the consent of stakeholders in other states regarding architectural characteristics of a project.	None. Since they were not convocated before decisions were taken, there was only one option, which was to consent the basic design.	None. That caused a lot of disorder and loss of sponsorship.
Another problem in the design of teams of various nationalities is the difference in culture that affects the understanding of business rules. Russians, for example, treated the registration of users differently from the way performed by Americans and Brazilians. It was a culture established of the software factories there.	Attempt to standardize the understanding across the internal trainings and better description of the use cases.	Description of well-defined business rules and the staff did not use vices related to development and test.
The functionalities were performing actions incorrectly due to noise communication between the user and the body of systemic solution (analysts and developers) working remotely.	Allocation of IT staff working to intermediate the communication between the client and the body of systemic solution (analysts and developers), translating customer expectations into a text that presents the required content and technical accuracy.	The best solutions are able to translate the needs from the perspective of the customers without losing technical accuracy.
Incomplete planning of material resources to facilitate communication between sites.	Outline what communication tools are needed for the development of the project to flow harmoniously without much interference from the manager.	

	Communication tools should provide an overview of the progress of activities for everyone, show the list of participants, the expertise of each team member and other functionalities.	
Lack of commitment from some members of the team and how to manage it.	Creation of measures for monitoring production. The developers that could not reach the minimum production received notifications. If this fact was repeated, then the developer was fired from the project if there were not good reasons.	Some cases have lasted until the resignation of the members.
Distributed teams in two different states (EP / DF), hindering daily monitoring of the project, risk identification and solution of negotiating disputes.	Use of tools that control encountered problems and used solutions, communication tools and meetings at head office via internet with other members located in another state.	Time reduction to solve disputes.
Participants had to perform activities at home and the code was the same. How to ensure versioning.	Use of version control tools such as SVN.	It is possible to version, but the collaboration is hampered because there is no how to collaborate (exchange of ideas, etc.) in the tools traditionally used.
In a project with a 9 members team working in 3 different cities, team members were not meeting the deadlines.	Deadline estimations were assigned to the member responsible for each activity.	
In a project with a 9 members team working in 3 different cities, lack of knowledge on the technologies used in the project was reported.	Face-to-face training sections and workshops at the begining of the project.	It was difficult to draw up a schedule fit for all members. Discussed material was sent for absent members.
In a project with a 9 members team working in 3 different cities, there existed no structure for performing tests.	Performance of only unitary tests in emulators.	Only functional requirements were tested.
In a project with a 9 members team working in 3 different cities, lack of communication with the client for clarification about features was reported.	A team member, with close ties to the client, strenghtned the communication between the developer and the client	
In a project with a 9 members team working in three different cities, a reduction in the number of members happened.	Reassignment of roles.	Problems reported by the client were fixed first, which led to an early release of some features and the postponement of some others.
In a project with a 10 members team working in the same city but in distant offices, there existed communication problems between team members.	New communication channels adopted: Internet forums, eletronic mailing lists and chat softwares.	
In a project with a 10 members team working in the same city but in distant offices, gathering all members in the same location was difficult.	Asynchronous communication replaced face-to-face meetings.	
In a project with 50 members working in 5 different cities, there were incompatibility problems with time to do simple and small meetings	Alocating the members in smaller teams.	
In a project with 50 members working in 5 different cities, the members were not prepared to use the development tools of the project.	Face-to-face training sessions and workshops, for all members to be at the same knowledge level.	
In a project with 50 members working in 5 different cities, there were difficulties in managing and validating the tasks performed by the teams.	Weekly virtual meetings for progress evaluation.	
In a project with 50 members working in 5 different cities, there was the need for a tool that enabled efficient communication among team members.	New communication channels adopted: Internet forums, eletronic mailing lists and chat softwares.	
	New scheduling method: two deadlines	

In a project with 24 members from 11 different countries deadlines were not met.	per project phase.	
In a project with 4 members from the same country time was wasted with unecessary tasks.	Determine which tasks were unecessary and the time spent with them, for improving the project management.	
In a project with 202 members working in two different countries, not all members were familiar with concepts from Requirements Engineering, Development Processes, Software Configuration and Management.	Training sessions so the knowledge required was attained by all members.	
In a project with 202 members working in two different countries, there existed problems in the use of different Process Models.	Implement the same Process Model for all teams in the project.	
In a project with an unknown number of members that work in 3 different countries, difficulties in performing functional tests of the system were reported, since the members were not familiar of the businnes model being tested.	Visual modeling of the project using UML, U2TP and TTCN-3 for a better comprehension of the business model.	
In a project with an unknown number of members that work in 3 different countries, the businnes plan to be followed was not well understood.	Continuing traning sessions with members for a complete comprehension of the business model.	
In a project with an unknown number of members that work in 3 different countries, there was a lack of clarity in regards to how tests were to be performed.	Study the modeling (documentation of the system) so the tests may be perfomed precisely.	
In a project in which nor the number of members nor the cities involved were reported, problems related to cultural differences among members happened.	Training sessions on cultural differences with all members.	
In a project in which nor the number of members nor the cities involved were reported, some members did not have the required level of knowledge.	Implementation of techniques for knowledge transfer among members.	
In a project in which nor the number of members nor the cities involved were reported, syncronization problems happened with tasks performed in a distributed manner.	Scheduling of face-to-face meetings in times reasonably compatible with all members aiming a better integration.	
In a project with an unknown number of members working in one country, there existed problems in the identification of project risks.	Analyse the hierarchy of problems, so as to determine if one problem is led by another and from that identify risks.	
In a project with an unknown number of members working in one country, problems with risk mitigation were reported.	Perform risk management individually and analyse the impact over the hierarchy.	Creation of periodic risk mitigation reports, for determining whether the solution is recommended.
In a project with an unknown number of members working in 4 different countries there existed problems with the delegation of tasks by the projects managers.	Evaluate each team member so a precise picture of their skills may be obtained.	Improve communication among managers and members, through interviews.
In a project in which nor the number of members nor the cities involved were reported, there existed problems related to lack of technical knowledge of members of the team.	Delegation of tasks according to each member's knowledge and interests, whenever possible.	
In a project with 355 members working in 2 different countries, problems in the alocation of tasks were reported, due to national and cultural differences.	Interview best members of each team, so as to delegate the rights tasks according to their culture and skills.	
In a project with 355 members working in 2 different countries, feedback problems from	Analyse each team's time zone, so as to keep the project running 24/7.	While a team is not active, another takes charge in order to continue the development.

clients and teams happened due to the distance and time zones.		
In a project with an unknown number of members working in one country, geographical distance led to communication problems among the teams.	Keep documentation of each team up-to-date and available to all members of the project in order to minimize the number of travels and meetings.	
In a project in which nor the number of members nor the cities involved were reported, problems happened due to the addition of new requirements (both functional and non-functional) during the architecture and modeling steps of the project.	Use an UML extension model.	Any project changes during the architecture or modeling stages must be known by all members of the team.
In a project with an unknown number of members working in one country, communication problems among the teams existed.	Assign a leader for each team, who are to be responsible for the communication among all teams.	

B

Formulários utilizados para coleta de dados do Mapeamento

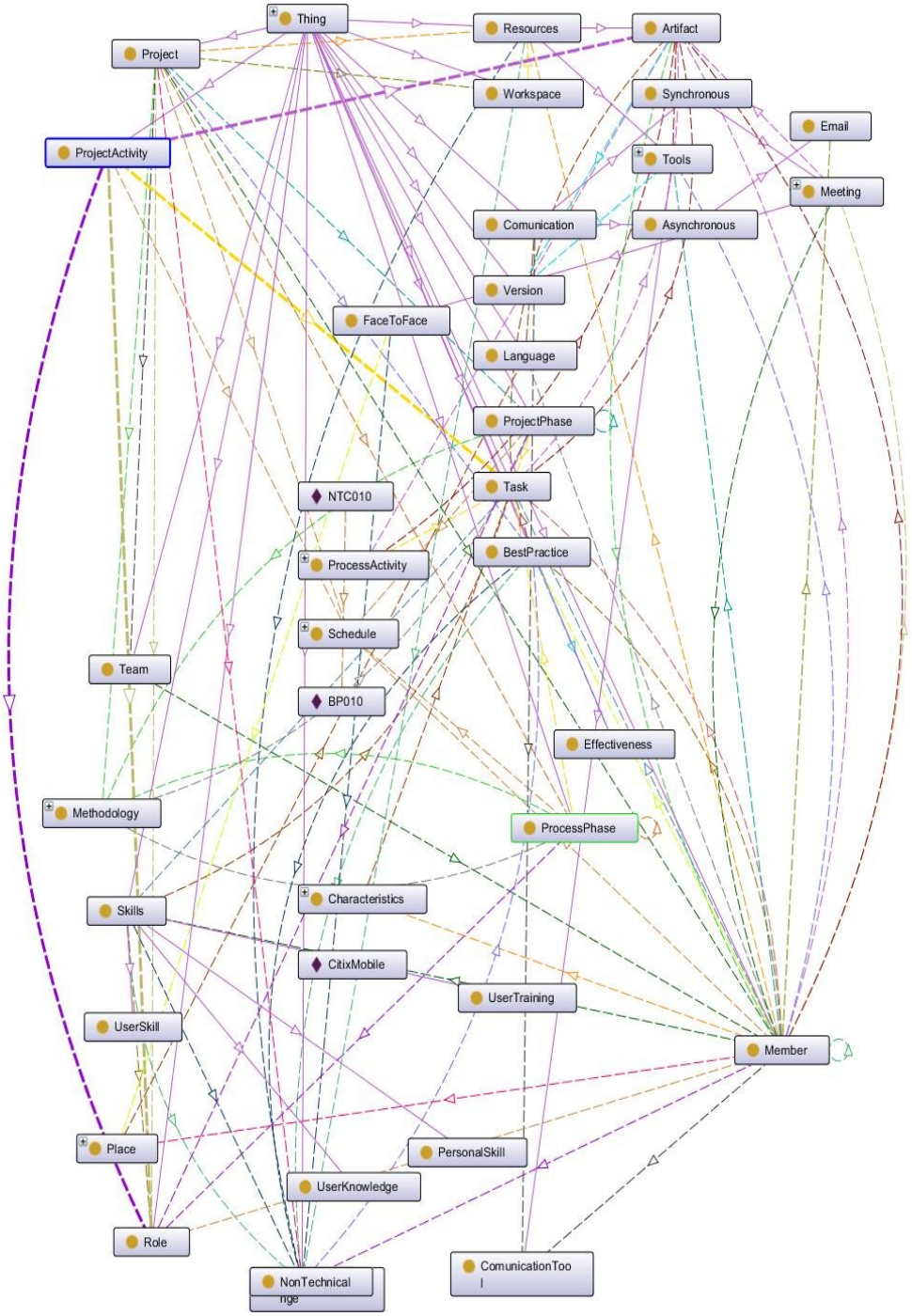
FORMULÁRIO DE COLETA DE DADOS Revisão Sistemática na Utilização de Ontologias no DDS	
ID: EPXXX	
Pesquisador:	**Data da Avaliação:**
Título do Trabalho:	
Autores:	
Fonte de Pesquisa:	
Local de Publicação:	
Tipo do Local de Publicação:	**Ano:**
Tipo de Estudo:	
Foco:	
Método de Pesquisa:	
Tipo de Contribuição:	
País do Estudo:	
Pontuação de Qualidade:	
Resumo do Trabalho:	
Comentários Adicionais:	
QUESTÕES DE PESQUISA **Como o trabalho responde as seguintes questões de investigação**	
Q1: Quais as técnicas que utilizam Ontologias no desenvolvimento distribuído de software?	
Q2: Quais as ferramentas que utilizam Ontologias no desenvolvimento distribuído de software?	
Q3: Quais os modelos que utilizam Ontologias no desenvolvimento distribuído de software?	
Q4: Quais as boas práticas que utilizam Ontologias no desenvolvimento distribuído de software?	
Q5: Quais as Ontologias formalizadas no contexto de desenvolvimento distribuído de software?	

FORMULÁRIO DE AVALIAÇÃO DA QUALIDADE Revisão Sistemática na Utilização de Ontologias no DDS	
CRITÉRIO	**AVALIAÇÃO**
1. Os objetivos ou questões do estudo são claramente definidos?	
2. Existe uma clara descrição do contexto na qual a pesquisa foi realizada?	
3. O estudo é coerente e bem estruturado, permitindo avaliação?	
4. Os resultados encontrados estão relatados de forma clara?	
5. A pesquisa apresenta comparação de seus resultados?	
6. O estudo apresenta técnicas, ferramentas, modelos ou boas práticas que utilizam ontologias no apoio ao DDS?	
7. O estudo apresenta ontologias formalizadas no contexto de DDS?	
TOTAL DE PONTOS	

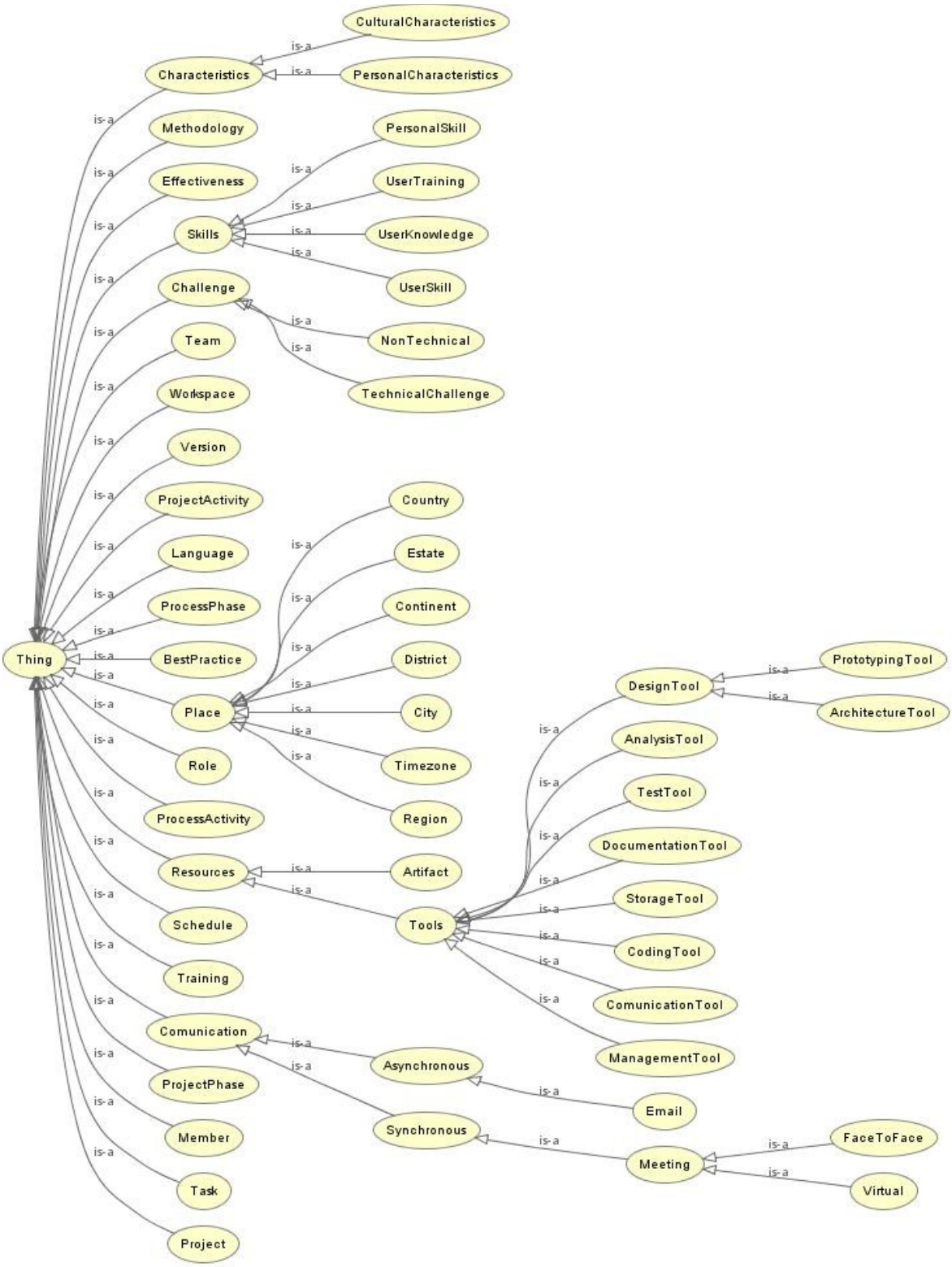
C

Documentação da DKDOnto

Relacionamento de Classes



Hierarquia de Classes

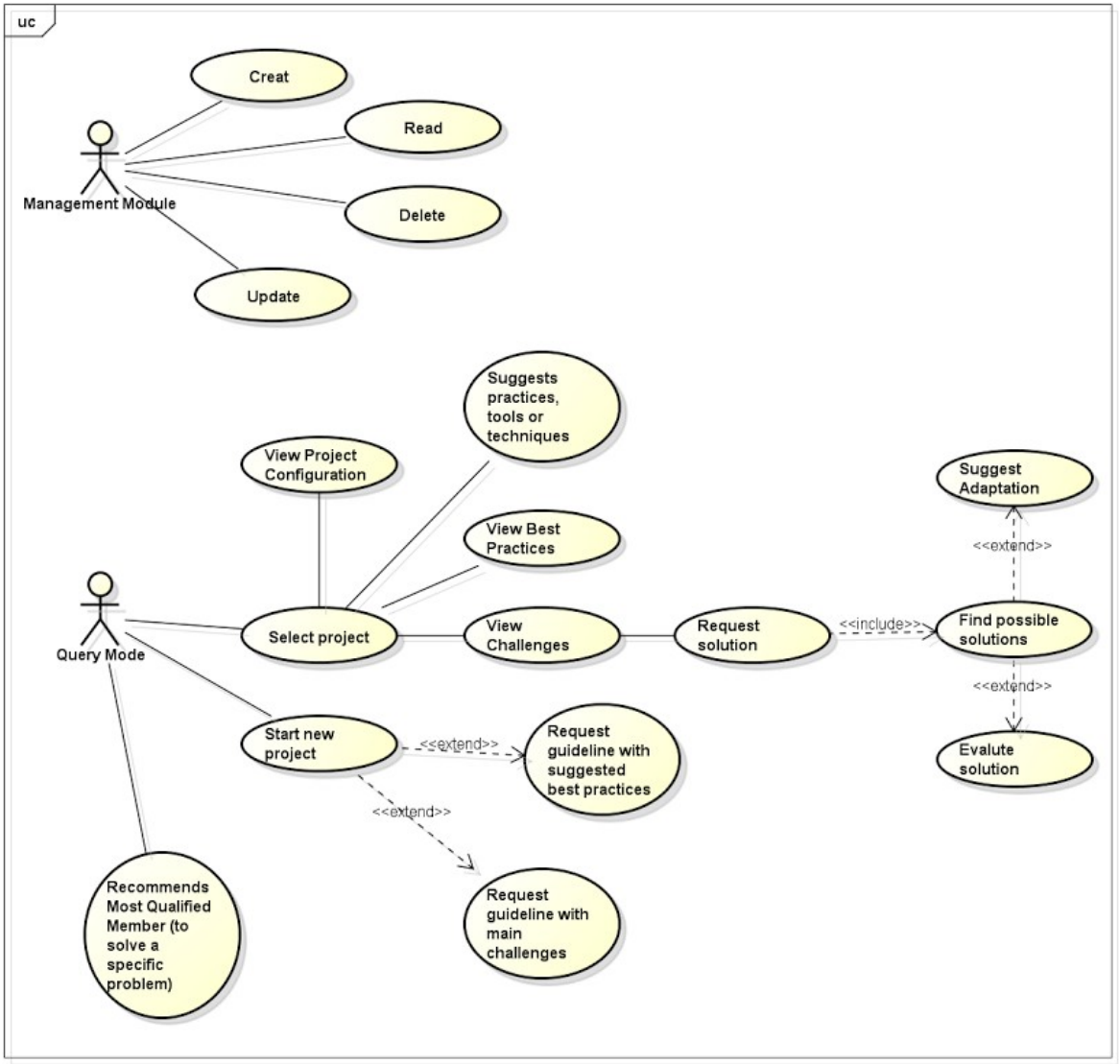


Alguns axiomas da DKDOnto

- project ⊑ ¬ team ⊓ ∃has.(bestPractice ⊓ challenge ⊓ team ⊓ language ⊓ workspace) ⊓ (∃distributed.severalPlace)
- role ⊑ ¬ member ⊓ (∃requires.skill ⊓ ∃composes.team)
- task ⊑ ¬phase ⊓ (∃generate.artifact ⊓ ∃composes.phase)
- member ⊑ roleOfmember ⊓ (¬ (∃authorizes.(bestPractice ⊓ challenge))) ⊓ (∃composes.(team ⊓ project)) ⊓ (∃communicates.member) ⊓ (∃has.(culturalCharacteristic ⊓ personalCharacteristic ⊓ skill ⊓ role ⊓ characteristic)) ⊓ (∃speaks.language) ⊓ ∃generates.artifact ⊓ ∃uses.tool ⊓ ∃performs.task ⊓ ∃located.place
- culturalCharacteristics ⊑ (∃generates.nontecnnicalChallenge ⊓ ∃helpstosolve.nontecnnicalChallenge)
- personalCharacteristics ⊑ ∃helpstosolve.nontecnnicalChallenge
- manager ⊑ (∃authorizes.(bestPractice ⊓ challenge ⊓ bestPractice)) ⊓ (∃suggests.(bestPractice ⊔ challenge))
- challenge ⊑ ¬ (methodology ⊔ phase ⊔ resource)
- bestPractice ⊑ ¬challenge ⊓ (∃solves.challenge) ⊓ (∃uses.resource) ⊓ (∃generates.case)
- nontechnicalChallenge ⊑ challenge ⊓ ∃affects.comunication
- comunication ⊑ ¬ language ⊓ (∃generates.bestPractice) ⊓ (∃helpstosolve.challenge)
- faceToFace ⊑ meeting ⊓ ∃avoids.challenge
- team ⊑ ¬ (project ⊔ member) ⊓ (∃use.(workspace ⊓ methodology))
- resources ⊑ ∃located.workspace
- meeting ⊑ synchronous

D

Diagrama de Casos de Uso – DKDs



E

Documentação RBC Approach

Modelagem Lógica do Banco de Dados.

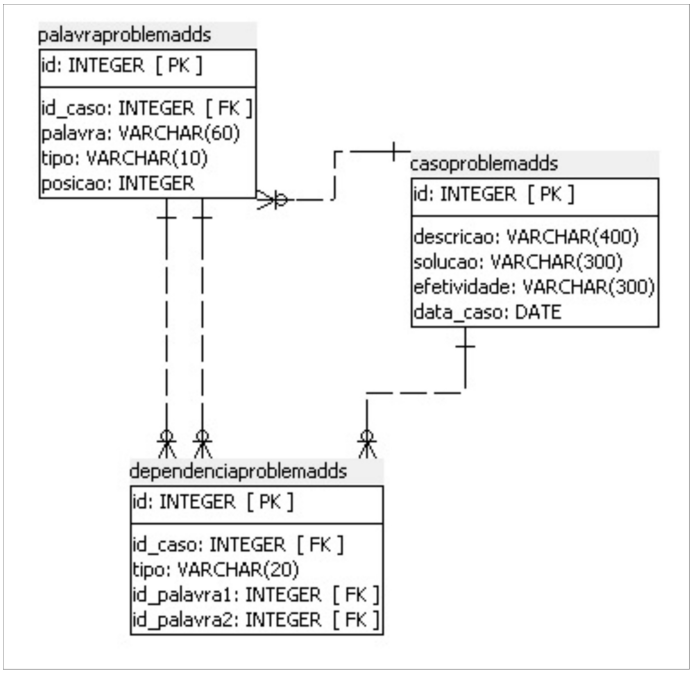


Diagrama de Atividades da Busca de Casos Semelhantes.

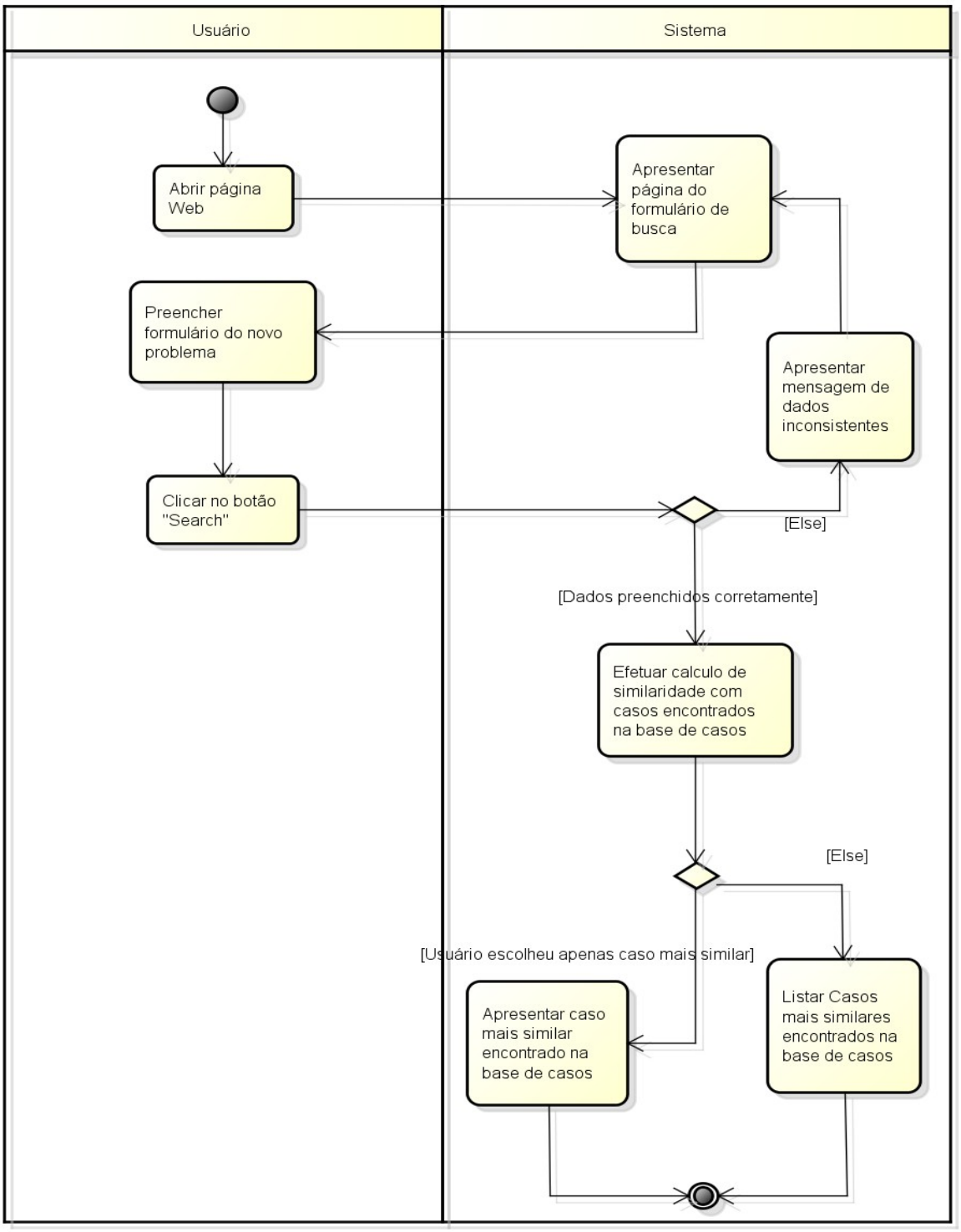


Diagrama de Atividades da Retenção de Novo Caso

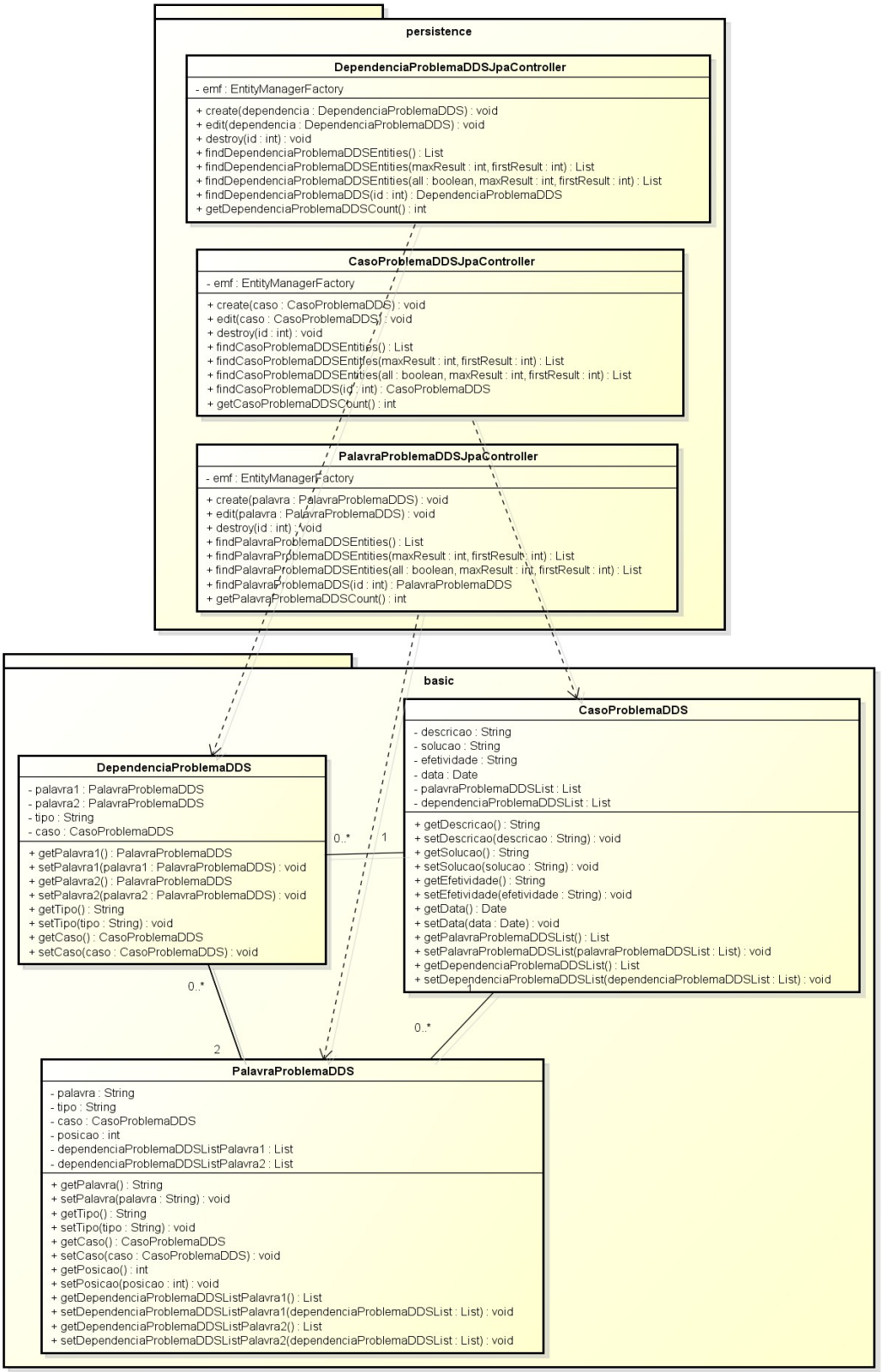


Diagrama de Classes – Relacionamento entre pacotes “*basic*” e “*persistence*”

Diagrama de Classes – Relacionamento entre pacotes “*view*” e “*reasoning*” e “*pln*”

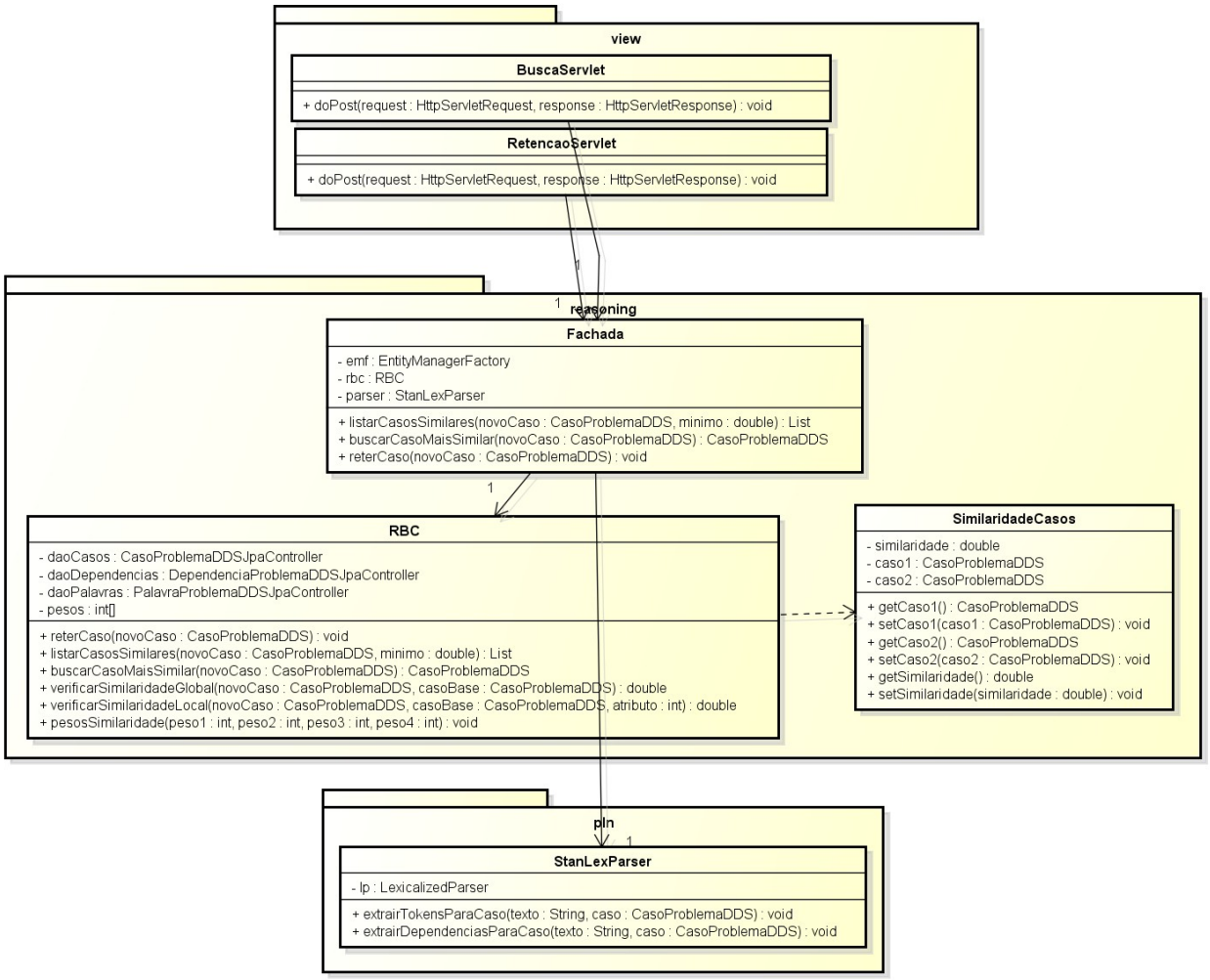
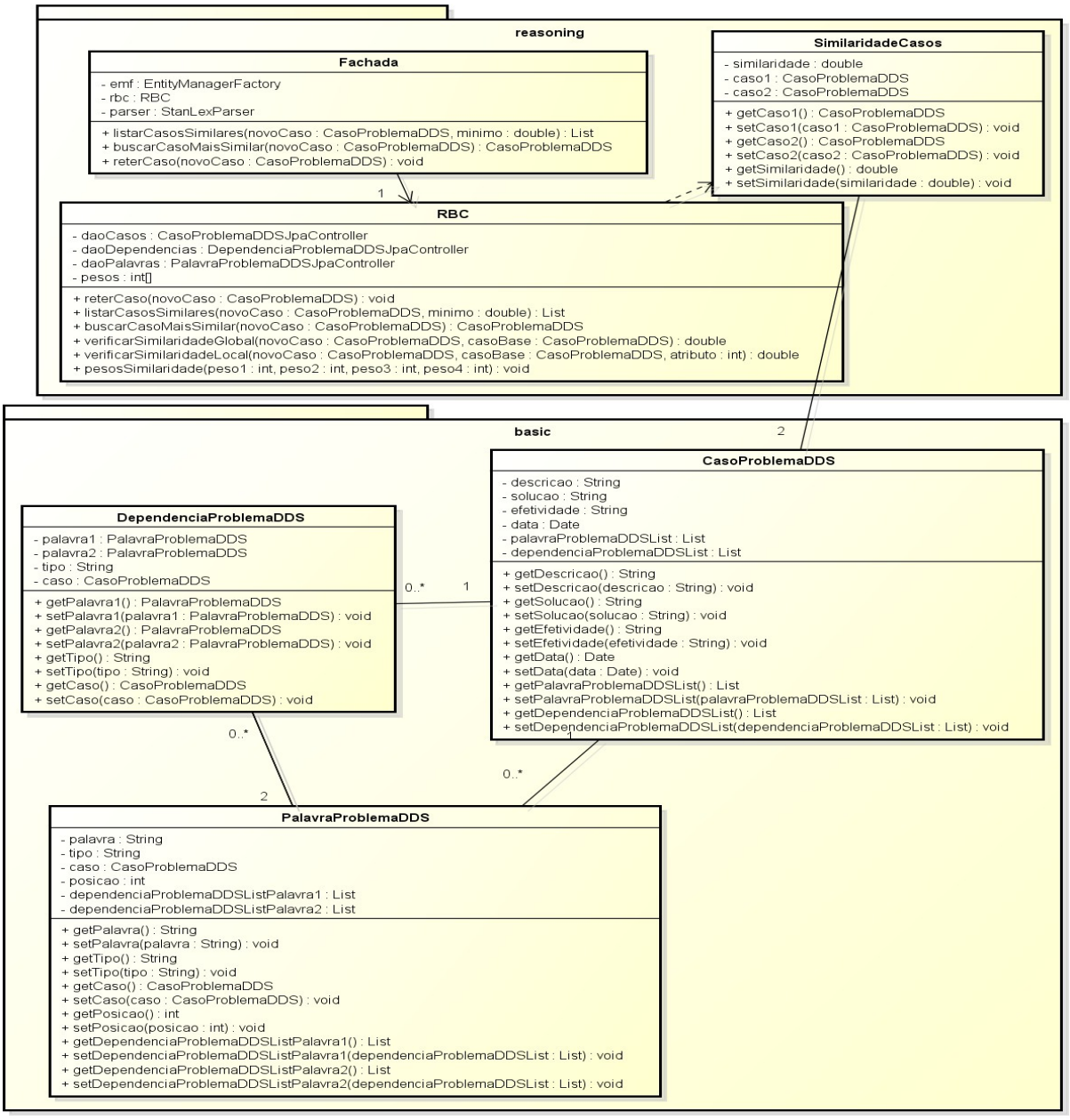


Diagrama de Classes – Relacionamento classe SimilaridadeCasos e CasoProblemaDDS


reasoning
Fachada
- emf : EntityManagerFactory
- rbc : RBC
- parser : StanLexParser
+ listarCasosSimilares(novoCaso : CasoProblemaDDS, minimo : double) : List
+ buscarCasoMaisSimilar(novoCaso : CasoProblemaDDS) : CasoProblemaDDS
+ reterCaso(novoCaso : CasoProblemaDDS) : void
SimilaridadeCasos
- similaridade : double
- caso1 : CasoProblemaDDS
- caso2 : CasoProblemaDDS
+ getCaso1() : CasoProblemaDDS
+ setCaso1(caso1 : CasoProblemaDDS) : void
+ getCaso2() : CasoProblemaDDS
+ setCaso2(caso2 : CasoProblemaDDS) : void
+ getSimilaridade() : double
+ setSimilaridade(similaridade : double) : void
1
RBC
- daoCasos : CasoProblemaDDSJpaController
- daoDependencias : DependenciaProblemaDDSJpaController
- daoPalavras : PalavraProblemaDDSJpaController
- pesos : int[]
+ reterCaso(novoCaso : CasoProblemaDDS) : void
+ listarCasosSimilares(novoCaso : CasoProblemaDDS, minimo : double) : List
+ buscarCasoMaisSimilar(novoCaso : CasoProblemaDDS) : CasoProblemaDDS
+ verificarSimilaridadeGlobal(novoCaso : CasoProblemaDDS, casoBase : CasoProblemaDDS) : double
+ verificarSimilaridadeLocal(novoCaso : CasoProblemaDDS, casoBase : CasoProblemaDDS, atributo : int) : double
+ pesosSimilaridade(peso1 : int, peso2 : int, peso3 : int, peso4 : int) : void
basic
2
CasoProblemaDDS
- descricao : String
- solucao : String
- efetividade : String
- data : Date
- palavraProblemaDDSList : List
- dependenciaProblemaDDSList : List
+ getDescricao() : String
+ setDescricao(descricao : String) : void
+ getSolucao() : String
+ setSolucao(solucao : String) : void
+ getEfetividade() : String
+ setEfetividade(efetividade : String) : void
+ getData() : Date
+ setData(data : Date) : void
+ getPalavraProblemaDDSList() : List
+ setPalavraProblemaDDSList(palavraProblemaDDSList : List) : void
+ getDependenciaProblemaDDSList() : List
+ setDependenciaProblemaDDSList(dependenciaProblemaDDSList : List) : void
DependenciaProblemaDDS
- palavra1 : PalavraProblemaDDS
- palavra2 : PalavraProblemaDDS
- tipo : String
- caso : CasoProblemaDDS
+ getPalavra1() : PalavraProblemaDDS
+ setPalavra1(palavra1 : PalavraProblemaDDS) : void
+ getPalavra2() : PalavraProblemaDDS
+ setPalavra2(palavra2 : PalavraProblemaDDS) : void
+ getTipo() : String
+ setTipo(tipo : String) : void
+ getCaso() : CasoProblemaDDS
+ setCaso(caso : CasoProblemaDDS) : void
0..*
1
0..*
1
0..*
2
PalavraProblemaDDS
- palavra : String
- tipo : String
- caso : CasoProblemaDDS
- posicao : int
- dependenciaProblemaDDSListPalavra1 : List
- dependenciaProblemaDDSListPalavra2 : List
+ getPalavra() : String
+ setPalavra(palavra : String) : void
+ getTipo() : String
+ setTipo(tipo : String) : void
+ getCaso() : CasoProblemaDDS
+ setCaso(caso : CasoProblemaDDS) : void
+ getPosicao() : int
+ setPosicao(posicao : int) : void
+ getDependenciaProblemaDDSListPalavra1() : List
+ setDependenciaProblemaDDSListPalavra1(dependenciaProblemaDDSList : List) : void
+ getDependenciaProblemaDDSListPalavra2() : List
+ setDependenciaProblemaDDSListPalavra2(dependenciaProblemaDDSList : List) : void

F

Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study

Systematic Mapping Protocol

1. Background

This document present a research protocol used to guide the execution of systematic mapping to identify tools, techniques, best practices, and models that use ontologies to support the Distributed Software Development (DSD). The protocol used to guide the execution of systematic mapping was based on the guidelines defined by Kitchenham [1] and Travassos and Biolchini [2].

2. Research Questions

The research questions were defined based on the scope of this mapping. An important issue in this process was to search for reviews and accurate analyses on the field, looking for current researches and open challenges related to the use of ontological resources in Distributed Software Development processes.

2.1 Research Questions in Free Format

(Q1) Which tools, techniques, models and best practices that using ontologies have been proposed to support the DSD?

(Q2) Which ontologies have been proposed or adopted in the context of DSD?

2.2 Research Questions in Structured Format

Q1: tools, techniques, models and best practices used in DSD that using ontologies.

Q2: ontologies formalized in the DSD context.

The terms used to identify works to answer the research questions was based in specific words to Ontology, DSD and specific words, like techniques, methods, tools, models, etc.

3. Search Strategy of the Primary Studies

This section describes the search strategy of the primary studies. The first step was to build a search string. Based on the research questions, we identified the main search terms and its synonyms. These definitions were based on other reviews and systematic mappings that involved the search terms: DSD and Ontology. Besides, the definitions were developed under the guidance of experts and researchers.

The search string definition involved a testing phase, aiming to refine and obtaining the most appropriate string to the research objectives. The test phase was conducted using different versions of the string and performing automated searches in a few selected digital libraries, such as IEEE Digital Library and Elsevier Scopus. The first step of the test used a most comprehensive search string,

composed of several terms related to ontologies and knowledge, such as: knowledge representation, knowledge management, knowledge transfer, conceptualization and concepts formalization. These searches returned several papers, but just a few were related to the ontology topic. Hence all the research terms related with Ontologies have been removed. Using a reduced version of the string, the amount of works returned decreased, however, all the relevant works to the research found in the first search have also been collected. Consequently, the tests performed with the reduced version of the string showed more efficient results. The resulting search string from this process is presented in Table 1.

Table 1. Search Terms

Relative	Terms
Ontologies	(Ontology or Ontologies)
DSD	AND (Distributed Software Development OR Global Software Development OR Collaborative Software Development OR Global Software Engineering OR Globally Distributed Work OR Collaborative Software Engineering OR Distributed Development OR Distributed Team OR Distributed Teams OR Global Software Team OR Global Software Teams OR Globally Distributed Development OR Geographically Distributed Software Development OR Offshore Software Development OR Offshoring OR Offshore Outsourcing OR Dispersed Team OR Dispersed Teams OR Distributed Software Project OR Multi-site Software Development OR Distributed Environment of Software OR Outsourced Software Project OR Virtual Team OR Virtual Teams)
Especific	AND (Technique OR Techniques OR Method OR Methods OR Tool OR Tools OR Software OR Program OR Programs OR System OR Systems OR Model OR Models OR Framework OR Frameworks OR Methodology OR Methodologies OR Good Practice OR Good Practices OR Best Practice OR Best Practices OR Lesson OR Lessons OR Learned OR Success Factor OR Success Factors)

The search strategy used to map the primary studies involves automated searches through well-known digital library search engines. They were chosen based on the relevance for the computer science community, and the availability of papers on this field on the Internet or with libraries, which have partnership with the Federal University of Pernambuco. The search has been performed in the following digital libraries: i) ACM Digital Library (http://dl.acm.org); ii) EI Compendex (http://www.engineeringvillage2.org); iii) IEEE Digital Library (http://ieeexplore.ieee.org); iv) Science Direct (http://www.sciencedirect.com); and, v) Scopus (http://www.scopus.com/home.url).

4. Primary Study Selection Strategy

This section describes the inclusion criteria of the search and selection process of the primary studies. A team composed of three researchers was assembled to execute this process: a master's student and two PHD students. In addition, two professors supervised the whole process.

The inclusion of the work is done accordingly to their relevance to the research questions. The inclusion criteria of this research are: (1) the study must provide techniques, tools, models, and / or best practices using ontologies to support DSD or ontologies formalized in this context; (2) only full papers will be considered (extended abstracts and presentations will be disregarded); (3) repeated papers will

not be allowed, in this case, oldest and incomplete versions will be disregarded; and, (4) duplicated papers will not be permitted, the first instance found by the research will be considered. It is worth it to point out that these criteria were discussed with all the researchers involved in this research.

The primary studies selection process was performed in three steps:

- **Selection of primary studies potentially relevant to the research**: primary studies collected after the search process were assessed by the first author of this paper, considering only the title, abstract and keywords. Only papers clearly irrelevant to the study were excluded. Whether or not there was uncertainty regarding the relevance of the study, it was included. By the end of this stage, all potentially relevant studies for the research were selected for further analysis.
- **Selection of primary studies through the inclusion / exclusion criteria**: two researchers evaluated all papers approved in the previous selection, considering the title, abstract, keywords, introduction, and conclusion. In some cases, a meticulous reading of the paper was necessary. Each researcher was responsible for reading and evaluating the papers individually, including whether or not each one was relevant to the research through the inclusion criteria adopted. Whenever a paper did not fit in at least one inclusion criteria, it was excluded from the mapping. By the end of this step, each researcher prepared a report with their decisions on the inclusion or exclusion of each paper on primary research with their justifications.
- **Conflict resolution and final assessment of the primary studies selected**: in the final stage of selection, the two reports - one from each researcher - with the decisions of inclusion or exclusion of each primary study were compared. In cases of disagreement in the inclusion or exclusion decisions, a third reviewer was responsible to perform an even more detailed reading and opine on its relevance. At last, the two researchers held a meeting to discuss and decide on the inclusion or exclusion of papers that were in disagreement. The result of this process was the final set of primary studies included in this mapping study

5. Extraction and Synthesis of Data

After the selection of primary studies, it was performed the extraction and analysis of the relevant data to the research. To organize the data extraction, analysis and synthesis of this research, we adopted the JabRef [3] tool, which is enabled to make customizations, and eases the literature review overall process. All primary studies received a unique identifier to identify them easily during the process. For instance, PS01 means the first primary study selected. For each primary study, the following information was extracted and registered: **Title and Author(s)**; **Publish Location**: conference or journal; **Publication Date**; **Research Context**: academic, government or industrial; **Research Focus**: people, organizations or projects; **Research method**: experimental, survey, case study, theoretical study and experience report; **Contribution type**: validated research, ongoing research, proposed solution, theoretical work, opinion work, or work experience; **Answers to Research Questions**: evidence to answer the research questions investigated by this mapping; and, **Additional Information**: evidence to allow this mapping performs a secondary analysis, such as: (1) which stages of development these resources support; (2) which problems / goals leaded to the proposal / adoption of ontologies as formalism for representing data in the DSD; and (3) which are the advantages and disadvantages of using ontologies to support DSD.

6. Criteria of Quality Evaluation

The quality evaluation of each primary study has also been conducted in order to provide an overview of the primary studies quality and focus the efforts on the most important studies in the process of result analyses. Therefore, the quality evaluation criteria used were: (1) clear, well-defined research objectives; (2) description of the research context; (3) coherent and well-structured studies; (4) presentation of results; and, (5) comparison of these results with other works.

In this process, each primary study received: 0 (primary study did not meet the criterion), 0.5 (primary study partially met the criterion) or 1 (primary study completely met the criterion) point for each criterion defined. Therefore, primary studies could be classified as: Bad (0, 0.5 or 1 point), Regular (1.5 or 2 points), Good (2.5 or 3 points), Very Good (3.5 or 4 points) or Excellent (4.5 or 5 points).

7. References

[1] Kitchenham, B., “Guidelines for performing Systematic Literature Reviews in Software Engineering”, EBSE Technical Report (EBSE’2007), Department of Computer Science, University of Durham, UK, 2007.

[2] Travassos, G. and Biolchini, J., “Revisões Sistemáticas Aplicadas à Engenharia de Software”, XXI Brazilian Symposium on Software Engineering (SBES), Brasil, 2007.

[3] JABREF. Available in http://jabref.sourceforge.net/

G

Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study

Results of the Quality Evaluation

The following table presents the complete results of the quality evaluation process realized in the systematic mapping study conducted in this research.

Quality Classification	Code of the Primary Study
Bad	
Regular	PS19, PS23, PS25, PS43
Good	PS6, PS7, PS8, PS13, PS14, PS21, PS22, PS26, PS33, PS34, PS40, PS41, PS44, PS46, PS51
Very Good	PS4, PS9, PS11, PS12, PS18, PS24, PS27, PS29, PS30, PS32 PS36, PS37, PS39, PS42, PS47, PS48, PS49, PS50
Excellent	PS1, PS2, PS3, PS5, PS10, PS15, PS16, PS17, PS20, PS28, PS31, PS35, PS38

H

List of Evidences

The following table presents the description of all evidencesidentified in the systematic mapping study conducted in this research. In general, are 57 resources that use ontologies in support to software engineering process in distributed development context.

Code of the Primary Study	Code of the Resource	Resource Type	ResourceDescription
T01	**PS01**	**Tool**	This tool is able to access, search and change software component ontology's data. The ontology defines concepts such as descriptions, attributes operations, features and relationships between the components. It allows the definition, standardization and sharing of components information, thus facilitates its reuse and integration.
T02	**PS02**	**Tool**	This is an ontology-based system that is able to assist the access and sharing of information between distributed teams. It is composed by clever software agents that make use of a knowledge repository which involves 4 ontologies: project domain concepts, Project information, Software Engineering concepts and project management concepts. Based on this knowledge repository, agents are capable of classifying problems inserted by software engineers and provide them with probable solutions, besides enabling the ability to search for information and managing the access and changes to the project data.
T03	**PS03**	**Tool**	This tool accompanies the collaboration among distributed teams during the process of software development. Ontology defines the concepts related to aspects of collaboration such as the team members, the relationship between them, time aspects, shared information resources and attributes. This tool provides a collaboration platform which enables a real-time capture and analysis of the interactions of distributed teams. Thus, it allows the evaluation of the evolution of collaboration and identification of collaboration patterns.
T04	**PS04**	**Tool**	An ontology-based approach for the development of a monitoring interactions system to be used in applications supporting the process of collaboration (groupware). This system allows collecting the actions of users of a groupware application during the process of collaboration, and stores the information in a document, facilitating analysis and evolution of the interactions among users. It was used an ontology to model the concepts involved in the collaborative work of the users, as an organization, actor, role, task, action, workgroup, and activity information.
T05	**PS06**	**Tool**	A multi-agent intelligent system to support communication and knowledge sharing in DSD projects. These agents have a knowledge base (ontology repository) which consists an ontology with concepts of software engineering and an ontology with project data.
T06	**PS08**	**Tool**	A tool for integrating an ontology engineering software and services for communities of engineers working on projects of DSD. The tool uses software agents to access the underlying ontologies and provide services such as: handling project information, provide recommendations for solutions to issues and allow the evolution of ontology software engineering.
T07	**PS09**	**Tool**	A framework that uses ontologies to facilitate the development and sharing of knowledge related to the process of system requirements. This framework facilitates communication and suggests a set of strategies to improve the requirements process in DSD projects, considering not only the characteristics of the project, but also the resources involved and the characteristics of the people involved
T08	**PS11**	**Tool**	A platform consists of multiple software agents and ontologies to support distributed collaborative work. It serves as the basis for the development of tools to support the process of collaboration between distributed teams. Through a generic model of intelligent agents which use ontologies as knowledge base, is possible to develop applications

			to exchange messages between users, sharing of knowledge, troubleshooting, task allocation, plan and build tasks.
T09	**PS12**	**Tool**	A system to support collaboration between virtual teams, focusing on the design process. Aiming to support collaboration among distributed teams, improve the allocation of resources and minimize the major problems in distributed design process, the system has an architecture based on software agents and ontologies, which are used to integrate and share knowledge.
T10	**PS13**	**Tool**	A system to support knowledge management and process execution in DDS. The system provides support for project coordination: allowing assign tasks, manage teams, set deadlines and monitor the progress of project. The tool allows developers to access task lists and information relevant to those tasks, and provide a better communication between the teams.
T11	**PS14**	**Tool**	A system to support members of Open Source software communities during the resolution process of "bugs". The system uses an ontology-based semantic interface to help members to exchange messages during the resolution of "bugs" and offer recommendations messages related to the artifacts. Ontologies are used to describe and store information about those artifacts and about the community structure.
T12	**PS15**	**Tool**	This tool assists distributed, agile processes that consist of knowledge related to a specific task. Ontologies are used to describe and relate concepts of tasks and information resources. Its main function is to match each set of information with a corresponding project activity, assign these activities to the developers and enable each and every one of them to analyze recurring information. Consequently, it organizes the information and facilitates the access to the most relevant piece of knowledge regarding a certain activity.
T13	**PS16**	**Tool**	This is a recommendation tool used in the requisite specification process into agile and distributed environments, at which the requisite analysts have a knowledge basis associated to the project's domain, which can be used to make decisions by the time of the elicitation of requisites. The system's knowledge basis integrates 4 ontologies: business domain ontology, ontology for agile requisites, a general requisite ontology and an environment structure one. These ontologies are to compose a project's knowledge basis with semantically related concepts, so it's possible to make inferences and real-time recommendations based on them.
T14	**PS18**	**Tool**	This is an open-source framework which offers access support, organization and knowledge sharing on the project among distributed software development teams. It utilizes ontologies to permit semantic annotations of the concepts involved and to unify the vocabulary used among the teams Ontologies define environment structures concepts, artifact contents, raised problems and their respective presented solutions, interaction elements among the developers, business rules and miscellaneous project data. This toll is also accessible on the Web and the Eclipse IDE and its main functions are: information search, knowledge sharing and automatic enlargement of the knowledge basis through the capture and inferences on user interaction data.
T15	**PS19**	**Tool**	A tool activities management for distributed environment that uses ontology to standardize the data of the project and involves the concept of collaborative work in spheres, ie, individual units of work. The tool allows users to manage and obtain information related to their activities.
T16	**PS20**	**Tool**	A tool for semantic integration of data from distributed sources. The tool allows for the collection, integration and distribution of data from

			different project repositories. The data collected is integrated into a common data model, through ontologies.
T17	**PS23**	**Tool**	A tool for visualization of structural and behavioral aspects of virtual collaboration, which was developed to perform in conjunction with the corporate chatsystem CVE LiveNet. The tool uses ontology with the underlying concepts of a virtual collaboration system, allowing an overview of the information being shared on the project.
T18	**PS24**	**Tool**	A tool to support those involved in the process of requirements elicitation, allowing collecting, organize, classify and relate the software requirements. The tool uses an ontology that defines the main concepts and relationships of the requirements. It also uses wikis, allowing a semantic collaboration in the requirements process.
T19	**PS25**	**Tool**	An agent-based system which use ontologies to support simultaneous, combination and coordination of various support services to software process, such as user login, user profile management, information retrieval of meetings, messaging, data integration, internet management and access control, and data recovery services. Ontologies are used to obtain a common semantic understanding of the information, services, their features and their relationships with other services.
T20	**PS26**	**Tool**	Uma ferramenta para gerenciar as informações relacionadas com cenários de testes de software que utiliza uma ontologia para definir e compartilhar informações relacionadas com o domínio e as atividades de um processo de testes. A ferramenta permite realizar raciocínios sobre a base de conhecimento desenvolvida.
T21	**PS27**	**Tool**	A system to support collaboration among distributed teams and heterogeneous in the design process through the management and unification of domain knowledge. Furthermore, it allows the integration of other tools and resources dispersed through a unified knowledge base and defined by shared ontologies.
T22	**PS29**	**Tool**	A system to automatically generate traceability of requirements in respect of DSD projects. Ontology with concepts and relations of the requirements is used to standardize the requirements artifacts. A set of requirements represented in natural language is syntactically and semantically analyzed through the Stanford Analyzer and its elements are placed in the ontology extracted.
T23	**PS30**	**Tool**	A web-based system to perform information gathering, knowledge extraction and ontology maintenance in software engineering. This ontology defines the concepts and relationships of engineering software, related software processes, requirements, design, implementation, testing and validation.
T24	**PS33**	**Tool**	A tool to support collaboration among distributed teams. The main function of the system is to integrate data and collaboration services from different sources. The system involves a collaborative development environment of Eclipse-based software and a collaborative web browser. Ontology is used as a knowledge base representing the relationships between concepts involved in the project context.
T25	**PS34**	**Tool**	A system for supporting the process of requirements elicitation in distributed environments, focusing on the interaction between staff and customers. The system has functions of graphical modeling, to customers and engineers can define requirements specifications using UML modeling. The system also involves language translation functions through software agents and a web translation service called Language Grids. The system uses two ontologies, one with concepts related to requirements and other with concepts for language services.

T26	**PS36**	**Tool**	A tool to facilitate the description of details and the reuse of concepts in the context of negotiations and establishment of electronic contracts in the context of DSD. The model uses ontologies as a mechanism to enable the reuse of information and artifacts in the process that must be repeated several times during the execution of various development projects involving DSD.
T27	**PS37**	**Tool**	A system to allow indexing of documents related to architecture and software requirements through an ontology aiming to facilitate access, search and traceability of knowledge involved in the project.
M01	**PS05**	**Model**	This model assists the formation of virtual teams that involve a quantitative method, ontologies with the project's data and professional abilities, as well as a search engine to search for information on experts in specific Web sources.
M02	**PS07**	**Model**	Model for managing knowledge across teams DDS, involving a knowledge base consisting of ontologies of Software Engineering and specific knowledge about project. The model involves three types of software agents: Safety Officer, which handles authentication and access levels to the system; Agent Ontologies, which manipulates and maintains the ontology repository, and finally, Agent Decision Making, which is responsible for controlling data updates of ontologies, which evolve to reflect the development of the project.
M03	**PS17**	**Model**	This is an enterprise software engineering model for integration of information related to the artifacts of the project. The model involves ontologies to define terminologies and relations between the artifacts, besides tools to access and share the artifacts' information.
M04	**PS21**	**Model**	A proposed methodology for software maintenance using semantic web technologies and involves the collection and recording of metadata of software, such as, requirements documents, information of software components, metrics and test data. Ontologies are used to represent the concepts of software engineering. Further, the RDF language is used to define structures of the metadata of software, and SPARQL, is used to perform queries about data and extract knowledge.
M05	**PS22**	**Model**	Model for knowledge management based on topics map (local knowledge bases) and semantic wikis (popular knowledge bases). The wiki pages use ontology-based metadata to provide advanced semantic search and navigation based on metadata, to support the development of software collaboratively in different contexts.
M06	**PS28**	**Model**	This model automatizes the selection and combination of services and software engineering backup tools, based on web semantics techniques and ontologies to define concepts and relations between the tools.
M07	**PS31**	**Model**	A Requisites Engineering model that uses techniques for knowledge management and ontologies as support to software requisites teams in DSD. Ontologies are used to formalize the knowledge related to the requisites of the project and the organizational knowledge.
M08	**PS32**	**Model**	A ontology-based model of software development for the development of Open Source softwares which allows categorizing information, improving communication and allowing the development of sophisticated search agents. The model involves: i) a semantics database through ontologies; ii) semantic queries, facilitated by use ontologies iii) generating RDF iv) the semantic navigation database; and, v) collecting and analyzing new data to be added in the semantic knowledge base.
M09	**PS35**	**Model**	This model disseminates the textual information in DSD environments which involves collaboration tools and ontology to describe the concepts and shared elements.

BP01	**PS01, PS21**	**Best Practice**	Using mechanisms based on ontologies to standardize and share information related software components across distributed teams facilitates the integration of components developed remotely, as well as facilitate understanding of its characteristics and, consequently, the reuse of these components in other projects.
BP02	**PS07**	**Best Practice**	Using ontology-based mechanisms to assist discussions and decision making between distributed teams can reduce the impacts of low synchronous communication and improve the setting changes during the process of software development in distributed scenarios.
BP03	**PS02, PS06**	**Best Practice**	Using intelligent systems capable of answering questions independently, identifying and managing problems during the project, and classifying attributes and roles of design concepts. These types of systems typically use intelligent software agents in conjunction with ontologies, and may assist developers in resolving doubts and reduce the impacts caused by remote communication in DSD projects..
BP04	**PS03, PS23**	**Best Practice**	Collaboration is seen as a major challenge in DSD, so it is important to use collaborative mechanisms that enable assessments of the activities of communication and information exchange among distributed teams. These engines use ontologies to identify patterns of communication, manage collaboration activities and capture behaviors among teams.
BP05	**PS05, PS13**	**Best Practice**	It is important to have a knowledge base of competencies and professional skills, whether internal or external to the organization. A good practice is to use ontologies to define data professionals, being used as a knowledge base for mechanisms that work in formation of distributed teams appropriate to the needs of specific projects.
BP06	**PS02, PS06 BP07, BP09**	**Best Practice**	Using ontologies to standardize and share knowledge allows a uniform understanding of the concepts involved in design and facilitates communication among distributed teams.
BP07	**PS16, PS29**	**Best Practice**	Using ontologies to define knowledge associated with the domain of the project, the environmental context and requirements can assist the definition and evolution of requirements in distributed environments. These ontologies can serve as a knowledge base for tools to improve the agility, accuracy and consistency in the requirements specification, and as tools for information sharing and recommendations for requirements analysts.
BP08	**PS17**	**Best Practice**	Using ontologies to define terminology and relationships of artifacts, enables a unique understanding of knowledge, improving quality and accuracy of documents and artifacts generated. Furthermore, all of the artifacts related to a repository support traceability and awareness of the artifacts that are related.
BP09	**PS20**	**Best Practice**	Using ontologies enables the creation of a common data model and resource development to automate the gathering, integration and management of project data. This makes it possible semantic integration of data from different data sources that can assist in managing and sharing project information.
TE01	**PS07**	**Technique**	Technique that uses ontologies in the process of decision making between distributed teams based on voting and reputation of engineers software.Envolve a system that uses software agents and ontologies,

			and is used when a project team raises an important question and thus a reasoned decision about performing or not any changes in the knowledge base of the project. Based on the vote of the teams and the reputation of each member, which is controlled by the Markov model, the system performs or not the change in knowledge repository project.
TE02	**PS29**	**Technique**	Combining ontologies to automatically generate traceability of relationships from requirements. The Ontology with concepts and relations of the requirements is used to standardize the requirements artifacts.
M11	**PS45**	**Model**	ODEP-DPS, um processo holístico e colaborativo para o desenvolvimento de DPSs (*Data Providing Services).*
M10	**PS39**	**Model**	Este trabalho apresenta um framework para integração de componentes de software. O framework se baseia no conceito de Software as a Service (SaaS) e utilizam uma ontologia para anotações de componentes de software com especificações formais. O objetivo desta abordagem é fornecer uma boa integração, gerenciamento e escalabilidade de componentes de software em ambientes colaborativos e distribuídos.
TE03	**PS41**	**Technique**	O objetivo deste paper é estabelecer uma plataforma de sistema multi-agente para companhias individuais foramerem VEs ecológicas baseado em teoria ontológica e agentes inteligentes. As abordagens ontológicas incluem construção de ontologia compartilhada, correspondência de ontologias, integração de ontologias, armazenamento de ontologias e raciocínio ontológico. No caso geral em que o criador da VE é um fabricante e os parceiros colaboradores são fornecedores, o sistema multi-agente abrange três tipo de agentes inteligentes: agente gerenciador de conhecimento (KMrA), agente fabricante (MA) e aagente fornecedor (SA). MA e SA representam as capacidade e interesses do criador da VE e os parceiros da VE, respectivamente. KMrA fica encarregado do funcionamento de sub-tarefas da abordagem ontológica. Para selecionar parceiros para a VE ecológica, o seu criador também considerará os critérios ambientais, além dos critérios gerais de fornecedores, como preço, quantidade, qualidade e tempo de entrega.
TE04	**PS43**	**Technique**	Aquisição e verificação colaborativa de requerimentos de stakeholders distribuídos
T29	**PS43**	**Tool**	Sistema colaborativo em rede de aquisição de requerimentos semânticos de software baseado em inteligência social, com a contribuição de stakeholders e experts.

T30	**PS44**	**Tool**	FADYRCOS: Framework for Dynamic Reconfiguration of networked Collaborative Systems. Framework genérico que permite gerenciamento de sessão, adaptação de semântica multi-nível, implementação automática de componentes para atividades colaborativas.
T31	**PS46**	**Tool**	Mapas de conceito. Mapas de conceito baseados em ontologias pode ser muito útil como um sistema de aprendizado e comunicação entre stakeholders. Com o uso de mapas de conceito na Engenharia de software, há uma melhora na habilidade de stakeholders compreenderem o projeto, entender o relacionamento entre elementos e interações e facilmente expressar ideias e pontos-de-vista numa força facilmente compreensível.
T33	**PS49**	**Tool**	It is the first open ource system that seamlessly integrates a wiki-based platform with semantic knowledge representation for formal queries and reasoning with NLP web services for text analysis. These NLP services embody artificial intelligence (AI) ‘assistants’ that work collaboratively with the human users on a especification, within a single, cohesive interface. Several assistants are available by default, such as the detection of a number of common SRS defects, and custom NLP pipelines can be easily deployed through a service-oriented architecture (SOA).
T32	**PS48**	**Tool**	DiSen CollaborAR, software que provê uma estrutura para observar o contexto de artefatos ao longo do projeto. O software utiliza uma abordagem ontológica baseada em recuperação semântica, exibindo os artefatos em uma ontologia formal.
T28	**PS42**	**Tool**	TeamWeaver framework, que instrumenta a IDE Eclipse e sensoriza ações dos desenvolvedores como mudanças em código Java, invocações de um método no código-fonte, pesquisas na Web, ou ocorrências de erros de compilador.Cada ocorrência de uma ação do desenvolvedor é armazenada como uma instanciação de um conceito de uma ontologia descrevendo o domínio da engenharia de software.

I

Formulário para Extração de Informações e Formação de Casos

Evento	Ano	Tipo do Projeto (indústria / academia)	Breve Descrição Projeto	Num. de Participantes	Locais e Membros Envolvidos	Tecnologias Envolvidas	Artefatos Gerados	Papeis utilizados (gerente, desenvolvedor, etc)	Meios de comunicação empregados	Problemas conhecidos previamente	Problemas no decorrer	Adequações / Soluções	Certificações ?	Instituições Envolvidas (empresas, universidades, etc)

J

Questões de Competência

Quais os problemas de comunicação entre idiomas nos emails, workspaces dos artefatos e conversas por telefone ou VOIP, em um projeto que é distribuído entre 5 países diferentes e utiliza a língua inglesa como idioma principal?

Quais as fases, os papéis e os artefatos de uma metodologia específica de desenvolvimento de software para um projeto que está distribuído entre cidades distintas em um mesmo país, porém com o cliente localizado em outro país?

Em um projeto no qual os participantes estão distribuídos entre 4 países diferentes, como seria possível formar as equipes e distribuir as atividades com base nos conhecimentos com o cuidado nos respectivos fusos horários?

Quais os riscos enfrentados por um gerente em um projeto distribuído sem o conhecimento dos membros que compõe os times do projeto em cidades e regiões distintas, porém em um mesmo país?

Em um projeto cujo cliente está localizado em outro país, qual melhor forma de comunicação deve ser utilizada?

Quais diferenças culturais entre participantes de um projeto que tem seus membros distribuídos geograficamente em países diferentes?

Quais os artefatos devem ser gerados para um projeto que deseja utilizar scrum e que tem aproximadamente 20 membros e está distribuído em 3 cidades de um mesmo país?

Qual a melhor forma de comunicação síncrona para equipes distribuídas com diferenças grandes de horário?

Em um projeto com mais de 100 envolvidos, distribuídos em 4 países do mesmo continente, quais as melhores ferramentas para trabalhar na fase de requisitos?

Como definir papeis para os membros dos times de um projeto onde parte da equipe é proveniente de outro país e totalmente nova?

Quais maiores desafios enfrentados por membros que nunca tiveram experiência com projetos distribuídos?

Quais melhores práticas para adotar por membros que não possuem determinadas experiências com as tecnologias propostas porém são impostas por parte da equipe de outro lugar?

Quais habilidades os membros devem possuir para desempenhar atividades que envolvem participação de diversos membros provenientes de outros continentes?

Quais habilidades um gerente deve ter para gerenciar um projeto que possui 4 países diferentes com idiomas nativos diferentes do inglês.

Quais frases de processo devem ser utilizadas numa metodologia ágil num projeto com 14 membros e distribuídos em regiões diferentes de um mesmo país?

Quais fases de projeto devem existir para um projeto que utiliza o RUP e tem equipes de 40 pessoas em 4 continentes diferentes?

Como um membro do projeto pode recomendar uma boa prática de forma que outros projetos da mesma empresa possam ser reutilizados?

Quais os maiores desafios para projetos com times que utilizam RUP em suas grandes equipes distribuídas?

Quais características certas tarefas requerem e são necessárias por membros que nunca participaram de projetos distribuídos?

Quais habilidades de um membro são mais necessárias para tais tarefas quando o projeto tem equipes que falam dois idiomas por padrão no projeto?