Alexandre L'Erario

M3DS: um modelo de dinâmica de desenvolvimento distribuído de software

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

São Paulo

2009

Alexandre L'Erario

M3DS: um modelo de dinâmica de desenvolvimento distribuído de software

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

Área de Concentração: Engenharia de Produção

Orientador:

Prof Dr Marcelo Schneck de Paula Pessôa

São Paulo

2009

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 10 de dezembro de 2009

Assinatura do autor

Assinatura do orientador

FICHA CATALOGRÁFICA

L'Erario, Alexandre
M3DS: um modelo de dinâmica de desenvolvimento distri - buído de software / A. L'Erario. -- São Paulo, 2009.
175 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1. Desenvolvimento de software I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II. t.

DEDICATÓRIA

Dedico este trabalho aos meus pais Donato Bernardo L'Erario (*in memorian*) e Rita Minichiello L'Erario.

AGRADECIMENTOS

À meus amigos da Universidade Tecnológica Federal do Paraná, Alessandro Duarte, Rogério Pozza, Antônio Carlos e aos amigos da Fundação Educacional do Município de Assis, Almir, Guto, ao meu amigo da FATEC André Trindade e aos demais que me ajudaram nos momentos difíceis.

Aos professores Fernando Laurindo, Marly, Mauro Spinola do Departamento de Engenharia de Produção por me ajudarem e contribuírem com informações em suas disciplinas.

À secretária Ivelise por me ajudar na organização das orientações e por me auxiliar sobre as normas e políticas do departamento. A Tomás Kameyama, pelas conversas e companheirismo na Fábrica de Software.

Às empresas que se prontificaram para a realização desta pesquisa.

À minha namorada Flávia Regina Belote, sempre carinhosa e compreensiva.

Agradeço aos meus pais Donato Bernardo (*in memorian*) e Rita, por me incentivarem e me apoiaram em todas as decisões da minha vida. A minha irmã Ana Lucia, que tanto me incentivou a continuar nesta pesquisa.

Ao meu sábio orientador prof. Dr. Marcelo Pessoa, que me aceitou no programa de doutoramento e sempre me norteou nos momentos em que precisei.

Agradeço a Deus, por me dar forças e tranquilidade quando precisava.

“Os problemas importantes com que temos que lidar não podem ser resolvidos com o mesmo nível de pensamento que tínhamos quando os criamos.”

Albert Einstein

RESUMO

Este trabalho apresenta um modelo de dinâmica de desenvolvimento distribuído de software, cujo objetivo é representar a realidade e os aspectos de ambientes de DDS (Desenvolvimento distribuído de software), a fim de torná-los observáveis e descritíveis qualitativa e quantitativamente.

Um modelo preliminar foi elaborado a partir da revisão bibliográfica e de um caso de experimentação desenvolvido por L'Erario et al (2004). Para a construção e validação deste modelo, a metodologia de estudo de múltiplos casos foi aplicada em diversas organizações que desenvolvem software de maneira distribuída.

Ao modelo preliminar foram adicionados estados e transições significantes para a dinâmica do desenvolvimento distribuído de software, originando então o M3DS (Modelo de Dinâmica de Desenvolvimento Distribuído de Software).

Duas versões do M3DS são apresentadas. Uma versão construída sobre uma máquina de estados, cujo objetivo é representar apenas a transições entre os estados. Outra versão equivalente, porém mais formal, é apresentada no formato de redes de Petri, na qual é possível visualizar a dependência entre transições e mudanças de estado.

Com este modelo, é possível compreender o funcionamento de um projeto distribuído e auxiliar na eficácia da gestão da rede de produção, além de auxiliar as demais entidades e pessoas envolvidas a obterem um posicionamento na rede mais preciso. O M3DS pode, também, auxiliar a detecção proativa de problemas originados a partir do desenvolvimento a distância.

Os resultados apresentados neste trabalho respondem a questão de como as organizações desenvolvedoras de software produzem software de maneira distribuída. A originalidade da pesquisa centra-se na construção de um modelo de dinâmica do desenvolvimento distribuído elaborado com os dados levantados a partir de seis estudos de casos.

Palavras-chave: Desenvolvimento distribuído de software. Modelo de dinâmica de ambientes de desenvolvimento distribuído.

ABSTRACT

This work presents a dynamic model of distributed development of software, whose objective is to represent the reality and the aspects of DDS environments, in order to turn them qualitatively and quantitatively observable.

A preliminary model was elaborated from the bibliographical revision and an experimentation case developed by L'Erario et al (2004). The construction and validation of this model used the methodology multiple cases study in several organizations that develop software in a distributed way. After this, states and transitions were added in the dynamics model of the distributed development of software creating the M3DS. (Dynamics Model of Distributed Development of Software).

Two versions of M3DS are presented. A version built on a state machine whose objective is demonstrating the transitions among the states. Another version equivalent, however more formal, it is presented in the format of Petri nets. The second version makes possible to visualize the dependence between transitions and state changes.

With this model it is possible to understand the operation of a distributed project, aiding in the effectiveness of the manager of the network production and people can obtain a precise positioning in network. Besides, M3DS can also aid the proactive detection of problems originated from the development at the distance.

The results presented in this work answer the question: how the development software organizations produce software in a distributed way. The originality of the research is the construction of a model of dynamics of the distributed development elaborated from data of six cases studies.

Keywords: Distributed software development. Distributed software development dynamic model.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADSL	*Asymmetric Digital Subscriber Line* – Linha Digital Assíncrona do Assinante
CAD	*Computer-aided design* – Design ajudado pelo computador
CAE	*Computer Aided Engineering – Engenharia ajudada pelo computador*
CAM	*Computer Aided Manufacturing* – Manufatura ajudada pelo computador
CBSE	*Component-Based Software Engineering* - Engenharia de software baseada em componentes
CMMI	Capability Maturity Model® Integration
CRUD	*Create, Retrieve, Update, Delete* – Criar, recuperar, atualizar e excluir
CSCW	*Computer-Supported Cooperative Work* – Trabalho cooperativo auxiliado por computador
CSD	*Collaborative Software Development* – Desenvolvimento colaborativo de software
CSP	*Communicating sequential processes* – Processos de comunicação sequencial
CVE	*Collaborative Virtual Environment* – Ambiente virtual colaborativo
DDS	Desenvolvimento Distribuído de Software
DSD	*Distributed software development* – desenvolvimento distribuído de software
DSDE	*Distributed Software Development Environment* – Ambiente de desenvolvimento distribuído de software
ES	Engenharia Simultânea
GGPS	*General Group Problem Solving* – Grupo geral de resolução de problemas
GSD	*Global Software Development* – Desenvolvimento Global de Software
GVT	*Global Virtual Team*- Time virtual global
IDE	*Integrated Development Environment* – Ambiente de desenvolvimento integrado
IP	Internet Protocol – Protocolo de internet
M3DS	Modelo de dinâmica de desenvolvimento distribuído de software

PCSDE	*Process-Centered Software Development Environment*- Ambiente de desenvolvimento de software centrado em processos
PN / RP	Petri Network - Redes de Petri
RUP	Rational Unified Process – Processo unificado Rational
SLA	Service Level Agreement – Acordo no nível de serviço
SOA	*Service-oriented architecture* - Arquitetura Orientada a Serviços
VPN	*Virtual Private Network* – Rede virtual privativa
VTS	*Virtual Team System* – Sistema de equipe virtual

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

Em épocas remotas, **dividir para conquistar** foi uma política eficaz adotada por César que conduziu o Império Romano ao seu auge. Segundo Keegan (2006), essa política, naquele momento histórico, significava explorar individualmente cada inimigo em cada uma de suas terras. Dessa forma, a informação era detalhada e a estratégia era elaborada de acordo com a situação, posição e cultura de cada um dos oponentes.

Neste trabalho, dividir para conquistar não significa apenas separar por módulos. O conceito aqui empregado é tão amplo quanto o utilizado por César. Significa trabalhar com equipes dispersas, com costumes, línguas, processos e tecnologias diferentes. Significa que, assim como a estratégia de César descrita por Keegan (2006), temos que trabalhar com as informações originadas de diferentes fontes, chamadas nesta era, de parceiros. Utilizamos tais informações para alocar trabalhos e obter o máximo de performance em um ambiente distribuído.

Este capítulo apresenta uma visão geral sobre o tema abordado nesta tese. O trabalho é contextualizado, ressaltando os principais motivos que levaram o autor a trabalhar a questão. Efetua-se neste capítulo uma breve abordagem sobre DDS (Desenvolvimento Distribuído de Software). Os conceitos envoltos sobre desenvolvimento distribuído de software, juntamente com as demais teorias empregadas são desenvolvidos no capítulo 2.

O ponto-chave do desenvolvimento de software com qualidade é entregar um produto para o cliente que atenda suas expectativas dentro de custo e prazo aceitáveis. Nessas condições, apresentar uma solução com qualidade, visto que a complexidade de sistemas tende a aumentar, torna-se um desafio para as empresas desenvolvedoras.

Para vencer tal desafio, as empresas adotam procedimentos padronizados para a construção de software. O conjunto destes procedimentos, segundo Tyrrel (2002) é denominado processo de software, o qual engloba desde a análise dos requisitos até a entrega do produto final.

Um processo de software tem vários propósitos, dentre os quais se destacam:

- Qualidade: a aptidão do software para seu propósito;
- Eficácia: a solução correta para o usuário (conceitos ilustrados por Laurindo (2002) e Tyrrel (2002)).
- Predição: capacidade de planejar, estimar e seguir cronogramas;
- Reutilização: capacidade de reaproveitar programas já implementados;
- Manutenção: facilidade de manutenção nas soluções já implantadas e facilidade em adicionar novos módulos.

A qualidade do software depende de muitos fatores, dentre os quais, o processo de desenvolvimento de software. No desenvolvimento de software, muitos conceitos e tecnologias são empregados. Delen (1999) preconiza a existência de critérios como interoperabilidade, padronização (ambiente baseado em regras), repositórios, padrões de visualização, comunicação facilitada (*enterprise models*). Para Eischen (2002), os conceitos de engenharia são aplicados na concepção do projeto, na linguagem de programação utilizada, no banco de dados, nas camadas de software, na maturidade do processo, no conhecimento do usuário e na documentação.

No entanto, para o sucesso do projeto, qualidade e processo de software nem sempre são suficientes. Isso leva as organizações a expandirem seus mercados, criando filiais em várias regiões ou estabelecendo parcerias com organizações distantes para o desenvolvimento de produtos.

Uma maneira de estabelecer parceria entre as empresas que produzem software é constituir redes de produção. Segundo Alstyne (1997), uma rede pode ser definida a partir dos elementos envolvidos na sua estrutura, pelo seu processo e seu propósito.

As parcerias nem sempre são locais. São compostas por várias organizações distantes fisicamente umas das outras. Esse fato ocorre porque forças econômicas atualmente tendem a migrar para mercados globais, ultrapassando barreiras e desenvolvendo novas formas de competição e cooperação. O resultado, segundo Herbsleb e Grinter (1999), é a criação de organizações virtuais ou VTS que podem operar com sucesso sobre as barreiras geográficas e culturais.

O desenvolvimento global de software, abordado neste trabalho com um tipo de desenvolvimento distribuído de software, tem tomado grandes proporções,

pois as organizações das mais diversas categorias descobrem constantemente que o desenvolvimento distribuído de software pode aumentar a produtividade e reduzir o custo. Por isso, segundo Carmel e Agarwal (2001), na década de 1990 o número de entidades que se engajaram no GSD (*Global Software Development*) foi pequeno, porém o cenário está mudando rapidamente.

Muitos motivos levam as empresas de software a desenvolver de maneira distribuída. As empresas mais maduras podem controlar a qualidade da subcontratada, garantindo a qualidade do produto final.

Organizações com grau refinado de gerenciamento de controle de produção, conforme Slack (2002) ou que possuem certificação CMMI, por exemplo, tendem a ter um grau refinado nas atividades de modelagem e projeto. Por causa disso, com uma subcontratação na forma de DDS, esperam reduzir o custo e o tempo de produção.

Em alguns casos, a empresa terceiriza parte de um projeto, pois ainda não domina uma das tecnologias empregadas.

Outro motivo para as organizações desenvolverem software de forma distribuída é a redução da necessidade de grandes contratações imediatas de funcionários. Conforme identificado em um dos casos abordados no capítulo 5, a justificativa para manter equipes de desenvolvimentos distribuídas é que, no inicio de um novo projeto, não é necessária a contratação de um grande volume de profissionais em um curto período de tempo. A empresa reorganiza o projeto de forma que ele possa ser distribuído entre os sites (unidade independente de produção de software distante das demais) já existentes. Com isso, sempre há uma equipe experiente no desenvolvimento do projeto e sempre um volume constante de contratações de novos profissionais.

Há, também, casos em que a organização inicia um projeto distribuído com o objetivo de transferir conhecimento a uma filial. A situação é comum na abertura de uma filial fisicamente distante. Um dos motivos pelo qual uma organização resolve estabelecer uma filial é a necessidade de manter um time de desenvolvimento próximo aos clientes. Outro motivo, identificado em um dos casos, é a redução de custo baseado nas características econômicas regionais.

Uma organização pode instalar uma filial em uma determinada região, porque o custo de vida e/ou o custo do profissional é menor. Além do custo, o DDS pode ser justificado quando uma determinada região oferece condições econômicas e principalmente disponibilidade de mão de obra especializada. É o caso de algumas empresas que se instalaram na Índia, segundo Le e Zhang (2007).

Uma das entrevistas com gerentes, apresentados neste trabalho como os casos no capítulo 5, identificou uma vantagem um tanto incomum do DDS. Uma empresa menor, porém com grau relativo de maturidade, oferecia aos seus melhores estagiários um programa de treinamento. Após contratado, o estagiário passava a produzir muito mais, pois, além dos treinamentos, o funcionário, neste instante, já agregava experiência e conhecimento. Porém, outra empresa, maior, contratava o mesmo funcionário, fazendo com que a equipe da empresa menor ficasse desfalcada. O fato levou a empresa menor a criar um site remoto, fisicamente distante de seu concorrente, com o intuito de reduzir as perdas de funcionários e projetos para a empresa maior.

Também há necessidade de desenvolvimento distribuído quando o projeto é muito complexo. Nesse caso, a ideia é segmentar o projeto de forma que cada site desenvolva em particular uma parte, de acordo com sua função, necessidade e/ou especialidade.

A aplicação dos casos relatados no capítulo 5 também identificou que algumas organizações precisam atuar de maneira distribuída, como forma de encontrar mão de obra disponível, por não tê-la encontrado em um ponto apenas.

A utilização de computadores autônomos interligados em ambientes organizacionais favoreceu o surgimento de *groupware*, sistemas de trabalho em grupo, (ANTUNES, 2002) como. As mais diversas ferramentas de apoio a *groupware*, síncronas (serviço de mensagem instantânea) ou assíncronas (e-mail, fórum) passaram a ser utilizadas amplamente nas organizações.

As tecnologias de comunicação, segundo Leavitt (2007) têm ajudado a evolução do desenvolvimento terceirizado, que também pode ser uma forma de DDS. Tecnologias guiadas pela internet - videoconferência e telefonia IP, por exemplo - tornam a comunicação mais ágil entre os desenvolvedores e seus clientes.

Contudo Grudin (1994) dá provas de que o *groupware* pode limitar a trabalho cooperativo dos participantes por causa do fator social e motivacional e aspectos políticos do lugar de trabalho. Além disso, o desenvolvimento de atividades em grupo pode acarretar conflitos, discutidos no capítulo 2. Tais conflitos podem desencadear má execução das atividades que, consequentemente, compromete o projeto (prazo, custo) e o produto final (qualidade do software).

Considerando os problemas ocasionados pelo trabalho em grupos ao desenvolvimento distribuído de software, é necessário algum mecanismo que norteie a produtividade do grupo e garanta gerenciamento eficaz de projeto a fim de obter um produto final com a qualidade prevista.

1.1 Motivação

Um projeto de software piloto denominado ManWapp (L'Erario, 2004) foi desenvolvido no departamento de Engenharia de Produção da USP:

> O eLabSoft do departamento de Engenharia de Produção da Escola Politécnica da USP em parceria com a Fundação Educacional do Município de Assis, atualmente pesquisa o desenvolvimento e o processo de software. O projeto ManWapp, como um de seus projetos pioneiros, serviu de subsídio para aplicação de processos e tecnologias. Entre estes processos destacam-se o desenvolvimento distribuído e a implementação de um aplicativo capaz de conduzir a informação o mais próxima possível do usuário. O resultado foi o desenvolvimento de uma aplicação pervasiva chamada ManWapp. (L'ERARIO et al, 2004, p.166)

Durante o desenvolvimento do projeto ManWapp, foram detectados vários problemas, que induziram os participantes a procurar soluções. Elas serviram de subsídio para publicação e formação de conhecimento do grupo. Em um âmbito mais generalizado, os problemas detectados durante o ManWapp foram encontrados na literatura, na qual os mais diversos autores adotaram soluções específicas. Este problemas motivaram a elaboração desta pesquisa.

O interesse de abordar o tema GSD/DDS surgiu quando o projeto ManWapp foi desenvolvido porém, para este trabalho, uma abordagem mais genérica e teórica foi contemplada porque outro grande fator motivacional surgiu durante a revisão bibliográfica feita na época. Muitos artigos encontrados abordam o GSD. Na maioria

dos casos, são artigos publicados em revistas ou em congressos. A grande parte dos artigos encontrados foi referente a estudos de caso. No entanto Herbsleb e Grinter (1999), Battin (2001) et al, Ebert e Neve (2001), Carmel e Agarwall (2001) relatam soluções específicas em suas experiências.

Todos os casos encontrados têm componentes fundamentais para práticas de DDS. Porém, durante a revisão bibliográfica, surgiu a necessidade de estabelecer o modo de como os casos diferentes podem ser comparados.

Há numerosos casos que analisam grandes organizações, grandes softwares, pequenas empresas, pequenos softwares, poucos ou muitos sites, sites homogêneos ou heterogêneos. Na dificuldade de compreender o funcionamento da produção distribuída de software e de como agregar esta base em um conceito, surgiu a necessidade da elaboração de um modelo, chamado, neste trabalho, de M3DS: modelo de dinâmica de desenvolvimento distribuído de software.

1.2 Objetivos

O objetivo deste trabalho é elaborar um modelo da dinâmica de funcionamento de uma rede de produção de software. Tal modelo é identificado neste trabalho como um modelo denominado **M3DS: modelo de dinâmica do desenvolvimento distribuído de software**.

Esta seção apresenta uma contextualização geral sobre um plano de macro visão dos objetivos. No capítulo 3, é abordada a metodologia empregada e consequentemente há um esclarecimento mais formal e específico sobre os objetivos.

Os objetivos gerais explanados nesta seção estão atrelados a conceitos explorados metodicamente no capítulo 2, em que o desenvolvimento distribuído de software é abordado.

Acerca do esquema geral do trabalho, várias perguntas emergem como forma de posicionar precisamente o contexto do problema principal da tese.

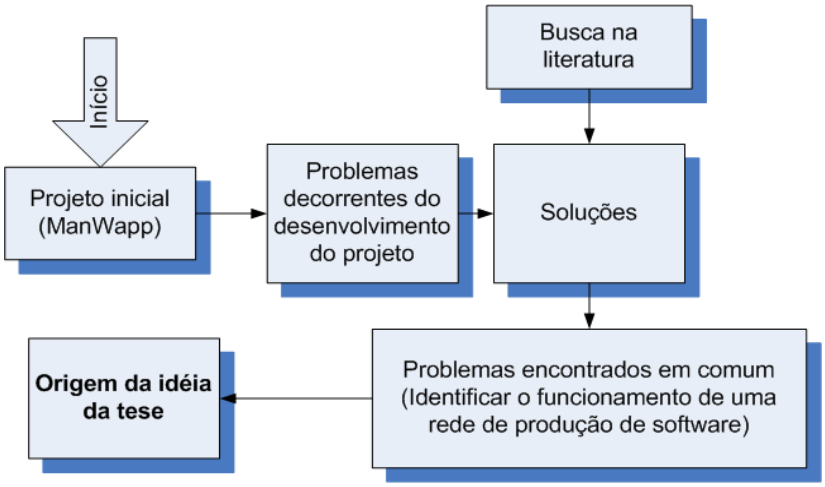

Figura 1 – Macrovisão da metodologia da tese

A figura 1 indica o roteiro que esta pesquisa percorreu. A metodologia utilizada neste trabalho foi o estudo de múltiplos casos. O estudo de caso visa a observar as organizações que utilizam DDS. Com essa metodologia, o trabalho pretende identificar variáveis fundamentais do ponto de vista do desenvolvimento, bem como abordar genericamente a forma como as organizações resolveram os impasses sobre o ambiente de produção distribuída.

1.2.1 Premissas

Baseado no projeto pioneiro ManWapp e em uma primeira revisão bibliográfica, este trabalho adota as seguintes premissas:

É possível desenvolver software em rede de organizações autônomas

Esta é a principal premissa adotada neste trabalho. Para que esta pesquisa fosse efetuada, foi comprovado a partir da literatura que as organizações podem desenvolver um projeto de software distribuído (DDS ou GSD). Esse desenvolvimento distribuído pode ser fruto de aglomeração de empresas autônomas cooperantes por apenas um período e/ou projeto, ou de fruto uma relação filial/matriz.

O impacto cultural compromete a qualidade do produto

Diversos autores – Carmel e Argawall (2001), Herbsleb e Mockus (2003) por exemplo - afirmam que as diferenças culturais comprometem a qualidade do produto. O fato ocorre devido à influência da cultura organizacional sobre o produto.

Não somente o idioma, mas processos e culturas regionais, como citados por L'Erario (2008), comprometem a coesão entre artefatos desenvolvidos distantemente. O resultado pode ser retrabalho a fim de compatibilizar estes componentes.

É possível sintetizar em um modelo a dinâmica do desenvolvimento distribuído

Há muitos casos na literatura identificados. Todos os casos sintetizam organizações de rede diferentes. Sintetizar em um modelo possibilita compreender melhor as diferenças entre redes de produções diferentes e alinhar redes semelhantes a fim de detectar proativamente impasses oriundos da distância física entre os *stakeholders* de um projeto distribuído.

1.3 Delimitação do escopo

Delimitar o escopo do trabalho significa estabelecer limites com relação ao universo da pesquisa de um trabalho. No contexto deste trabalho, inicialmente, são definidos os limites nos universos de empresas em que os estudos de casos foram empregados, nos tipos de projetos de software desenvolvidos pelas empresas e no foco da pesquisa.

Os casos foram selecionados baseados nos projetos de software que desenvolvem. O objetivo principal é analisar os projetos de desenvolvimento distribuído de software, globais ou não. Por isso, foram selecionadas organizações que já tinham experiência em DDS e o desenvolviam, sem impactos reestruturais na organização. Por esse motivo, as empresas escolhidas tinham um grau de maturidade de desenvolvimento de software e também desenvolviam software em um ambiente distribuído.

Entretanto, o modelo aqui proposto deve contemplar também as diferentes facetas do DDS, incluindo organizações de rede sem grau de maturidade.

Os projetos foram desenvolvidos distribuída e em alguns casos, globalmente. Na maioria, foram investigados projetos que demandaram grande volume de trabalho e tinham um volume considerável de linhas de código. Um projeto aberto (*open source*) foi explorado neste trabalho em particular e serviu de contraexemplo para os demais estudos de casos.

1.4 Processo de trabalho

O processo de trabalho empregado para o desenvolvimento deste projeto de pesquisa segue as seguintes fases:

Fase 1 - Preparação da pesquisa: Atividades referentes à delimitação da pesquisa, coleta e análise das referências bibliográficas, delimitação do tema e estabelecimento dos objetivos, questões e proposições.

Fase 2 - Estruturação da Pesquisa: Amadurecimento e análise do referencial teórico, definição do processo de investigação e o delineamento das conclusões. Seleção do método de pesquisa. Construção do roteiro de pesquisa e do protocolo de pesquisa.

Fase 3 - Execução da Pesquisa: Atividades de investigação, coleta de dados em campo por meio de entrevistas semiestruturadas nas empresas selecionadas, bem como a consulta de outras fontes referenciais de apoio, como documentos e fontes apresentados pelas empresas. A entrevista semistruturada é composta por um roteiro com o objetivo guiar o pesquisador na investigação das unidades de análise. A etapa engloba, ainda, a descrição individual dos casos.

Fase 4 - Análise dos Resultados: fase da análise dos dados de forma agregada, delineando o panorama do setor e extraindo as generalizações e conclusões.

Embora com as fases definidas sequencialmente, a pesquisa não foi desenvolvida de maneira sequencial e sim de maneira iterativa e incremental. Conforme ressalta Croom(2002), o processo de pesquisa é de natureza espiral e não

linear, com incursões pela literatura, sucedendo refinamento de proposições e, novamente, retornando à literatura e às experiências profissionais.

A figura 2 indica a relação entre o processo de pesquisa, os resultados esperados de cada fase e também quais os capítulos da tese gerados a partir dos resultados.

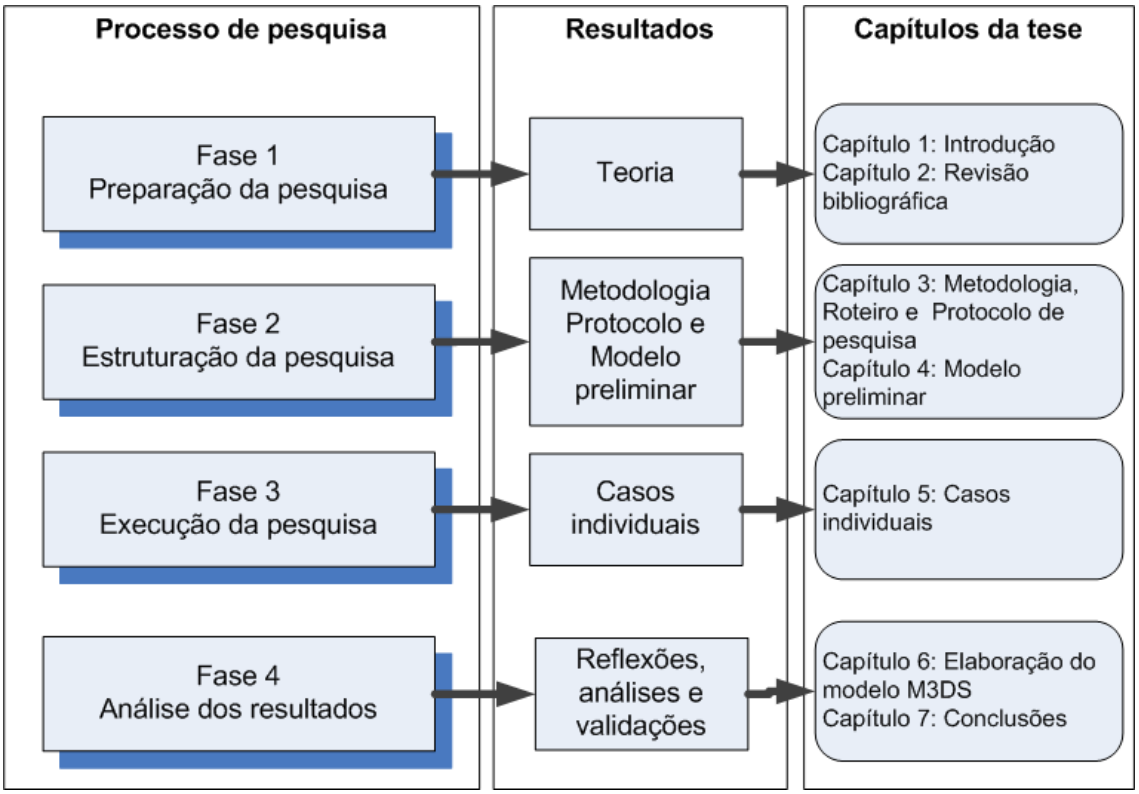

Figura 2 – Relação entre processo de pesquisa, resultados e capítulos da tese

1.5 Estrutura do trabalho

A figura 2 apresenta uma visão geral da estrutura da tese. Seguindo a figura, este trabalho está assim organizado:

- *Capítulo 1 - Introdução:* Visa a posicionar o leitor com relação ao projeto de pesquisa. Apresenta o contexto da área de aplicação desta pesquisa
- *Capítulo 2 - Revisão bibliográfica*: Apresenta uma reflexão sobre os principais autores da literatura. É exploração reflexiva dos textos encontrados.

- *Capítulo 3 - Metodologia*: Apresenta a metodologia empregada no trabalho; os motivos da seleção do método e o protocolo de pesquisa.
- *Capitulo 4 – Modelo preliminar*: A partir de uma primeira experiência (L'ERARIO et al, 2004), juntamente com a revisão bibliográfica, foi criado um primeiro modelo de dinâmica de desenvolvimento distribuído de software.
- *Capítulo 5 - Estudo de Casos:* Apresenta os casos de forma individual e sistemática, seguindo o protocolo de pesquisa apresentado no capítulo 3
- *Capítulo 6 - M3DS:* Demonstra o modelo elaborado a partir dos estudos de casos apresentados no capítulo 4
- *Capítulo 7 - Conclusões:* Fechamento da tese e apresentações das conclusões finais do modelo M3DS.

1.6 Considerações finais

O objetivo desta introdução não foi explorar profundamente o tema de trabalho e a metodologia, explorados nos capítulos 2 e 3 respectivamente.

Nesta introdução foram apresentadas algumas referências básicas sobre desenvolvimento distribuído de software e também foi indicada a metodologia empregada.

O posicionamento sobre a origem da questão, a qual se originou a partir do projeto pioneiro ManWapp, serviu como elemento motivacional para este projeto de doutoramento.

As figuras que não possuem referências a autores foram criadas neste processo de pesquisa.

CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta a conceituação teórica sobre Desenvolvimento Distribuído de Software, engenharia simultânea e redes de Petri que foram utilizados na consolidação deste trabalho. Neste capítulo, os problemas do DDS são alinhados ao escopo da tese. Há revisão dos principais problemas emergentes no desenvolvimento distribuído de software encontrados na literatura.

2.1 Desenvolvimento distribuído de software

O desenvolvimento distribuído de software ocorre quando vários sites cooperam e/ou colaboram para desenvolver um mesmo produto ou parte dele. Neste cenário, a complexidade do processo de desenvolvimento se amplia.

A redução do tempo é a principal razão da divisão de tarefas na produção distribuída de software, porém o tempo de comunicação e de resposta entre os nós pode ser incomensurável. Segundo Herbsleb e Mockus (2003) e Suzuki e Yamamoto (1999), há muitas variáveis neste cenário, tais como a cultura, a língua, a capacidade de cada site. Becker et al (2001) afirmam que desenvolver software em uma rede de produção requer gerenciamento mais eficaz de tarefas. L'Erario e Pessôa (2007) afirmam que, além do gerenciamento de tarefas, é necessário que o mecanismo de coordenação dos sites seja eficaz. Martin (1996), assegurando que esse gerenciamento deve estender-se a ferramentas, capacidades e informação.

Em sua pesquisa, Sa e Maslova (2002), afirmam que o crescimento da internet e de ferramentas de CSCW influenciaram direta e principalmente na difusão do GSD. Porém, embora a internet ofereça inúmeras vantagens em relação à comunicação, um projeto de software é muito complexo e grande, por isso Suzuki (1999) afirma a necessidade de um framework, que deve esclarecer como a informação é gerenciada na rede e como as tarefas são delegadas, por exemplo.

O desenvolvimento distribuído de software agrega várias vantagens sobre o desenvolvimento centralizado, como ensinam Battin et al (2001), Carmel e Argawall (2001), L'Erario et al(2004) em seus estudos de caso.

Segundo Battin et al (2001), foi possível desenvolver 511000 linhas de código para o projeto e cada centro de desenvolvimento colaborou significantemente para a implementação.

L'Erario et al (2004), em seu estudo de caso, utilizaram o desenvolvimento distribuído para a implementação de uma aplicação pervasiva. Para tanto, foi elaborado um arcabouço de distribuição de tarefas a fim de coordenar a entrega de partes da aplicação pelos nós.

Carmel e Argawall (2001) afirmam que o número de organizações que aderem ao GSD é crescente. Tais organizações são empresas que distribuem filiais ou efetuam *offshore* com o objetivo de aumentar a produtividade, reduzir o custo e tornar a distribuição do software mais compatível com características locais.

Entretanto dividir a produção de um software entre diversos nós envolve uma série de problemas. Alguns problemas que, localmente, são insignificantes ganham proporção quando uma tarefa é dividida entre organizações fisicamente distantes.

Esses problemas são relacionados a atividades do desenvolvimento do software, como é o caso do levantamento de requisitos. Vários autores, como Edwards e Sridhar (2002) e Campbell e Van de Walle (2003) citam em seus trabalhos os problemas inerentes ao desenvolvimento distribuído, quando a atividade de levantamento de requisitos também é distribuída.

Campbell e Van de Walle utilizam comunicação síncrona de informações, com o uso de workspaces e videoconferência e também inclui o uso de uma ferramenta síncrona, sistema de informação capaz de gerenciar requisitos.

O levantamento de requisitos também é abordado na pesquisa de Edwards e Sridhar (2003), em que é explanada uma forma de mensurar eficácia de times virtuais em projetos de engenharia de software.

Em uma abordagem um pouco diferente, Tiwana (2004) utiliza a abordagem de caixa preta como requisito para transferir conhecimento entre o cliente e o desenvolvedor em ambientes de *outsourcing* (terceirização). O autor descreve uma série de variáveis a ser consideradas na prática de terceirização. Tais variáveis estão relacionadas principalmente com as novidades tecnológicas para uma das entidades

(consumidor ou desenvolvedor) e indicam o que deve ser mensurado. As medidas devem ser obtidas a partir do conhecimento na tecnologia, no conhecimento em processos de software e em processos de negócios de ambos desenvolvedor e consumidor. Nesta pesquisa o autor descreve um *guideline* (roteiro) para *outsoucing*.

Grande parte das referências encontradas na literatura – Clerc; Lago e Van Vliet (2007), Barcus e Montibeller (2008), Kotlarsky; Van Fenema eWillcocks (2008) entre outros – são estudos de caso. Esse método de delineamento científico indica que o pesquisador não consegue separar seu objeto de estudo do contexto em que ele se encontra. Além disso, o estudo de caso pode identificar variáveis não previstas no inicio da pesquisa. O delineamento estudo de caso é abordado no capítulo 3.

2.1.1 Dispersão física

Segundo Audy e Prikladnicki (2007), o desenvolvimento distribuído de software é caracterizado quando um dos atores no envolvidos projeto estiverem fisicamente distante dos demais. Os autores também classificam a dispersão geográfica em três escalas: nacional, continental e global.

A variação da dispersão física influencia fortemente a possibilidade de reuniões presenciais e principalmente o horário de trabalho das equipes. Quando a dispersão física for nacional ou regional, há possibilidade de reunião presencial ou videoconferência mais eficaz. Isso é possível por não haver diferença no idioma dos *stakeholders*. Além disso, o fuso horário é praticamente o mesmo para todos os desenvolvedores

Na dispersão continental, além do idioma que não mais necessariamente é o mesmo, o fuso horário tem uma pequena influência. Nesse cenário, efetuar reuniões presenciais pode representar aumento no custo do projeto, uma vez que o custo (monetário e tempo gasto) do deslocamento entre os *stakeholders* pode ser elevado.

Quando a dispersão é global, pode ser inviável reunir presencialmente todos os integrantes do projeto. Nesse caso, o idioma é diferente, a cultura e o fuso horário também. Dentre outros problemas, a religião, a economia local e o fuso

horário e outros, emergem como diferença entre os integrantes e agregam problemas ao projeto.

L'Erario e Pessoa (2007) definem que o desenvolvimento global de software é um tipo de desenvolvimento distribuído de software. Na abordagem de desenvolvimento global o modelo de negócio é denominado como *offshore*.

A distância é a principal característica do DDS. Entretanto, quando global, pode agregar ao projeto diferenças culturais entre as equipes de desenvolvimento. Consequentemente, o risco do projeto tende a aumentar.

2.1.2 Modelos de negócio do DDS

O desenvolvimento distribuído de software é uma das consequências da globalização. Nesta era, as empresas podem organizar-se de maneira distribuída e, por isso, podem também estudar melhor os meios para otimizar a produção distribuída de software. Dessa forma, um projeto de DDS pode ser resultado da distribuição de tarefas em um mesmo país/região ou até mesmo entre países diferentes, caracterizando o desenvolvimento global de software (CARMEL; ARGAWALL, 1999).

O relacionamento das empresas participantes de um projeto de DDS é uma característica que precisa ser definida. O resultado dessa relação é fortemente influenciado pela localização geográfica dos sites.

A soma da união entre distância física e a relação entre as empresas contribui para consolidação de um modelo de negócio dos projetos de DDS.

A relação entre as empresas tem ocorrido de três formas principais, segundo Robinson et. al. (2004) apud Prikladnicki (2006):

- Terceirização (*outsourcing*) ou aquisição: A empresa delega o controle sobre uma ou mais de suas atividades para uma empresa externa.
- *Joint-venture* ou colaboração: É um acordo entre duas ou mais empresas, que, com a união de recursos, executam um ou mais projetos por um determinado período de tempo.
- Departamentos / subsidiárias da empresa (*insourcing*): A empresa cria seu próprio centro de desenvolvimento de software.

Do ponto de vista de distância geográfica, a distribuição ocorre de duas formas, segundo Robinson et. al. (2004) apud Prikladnicki (2006):

- *Offshore*: Este cenário é caracterizado pelos diferentes países onde residem os *stakeholders* de um mesmo projeto. O cliente e até mesmo os desenvolvedores do software (codificadores, administradores de banco de dados, arquitetos e outros) podem residir em países distantes.
- *Onshore*: É considerado cenário de *onshore* quando todos os *stakeholders* de um mesmo projeto residem em um mesmo país. Podem ocorrer duas situações de desenvolvimento: *onsite* e *offsite.* Na primeira situação o cliente e desenvolvedor estão fisicamente no mesmo espaço físico. Na segunda, não.

Buscando uma visão unificada das formas de relacionamento entre as empresas e a distribuição geográfica, a figura 3 traz um modelo adaptado de Robinson et. al. (2004) apud Prikladnicki (2006).

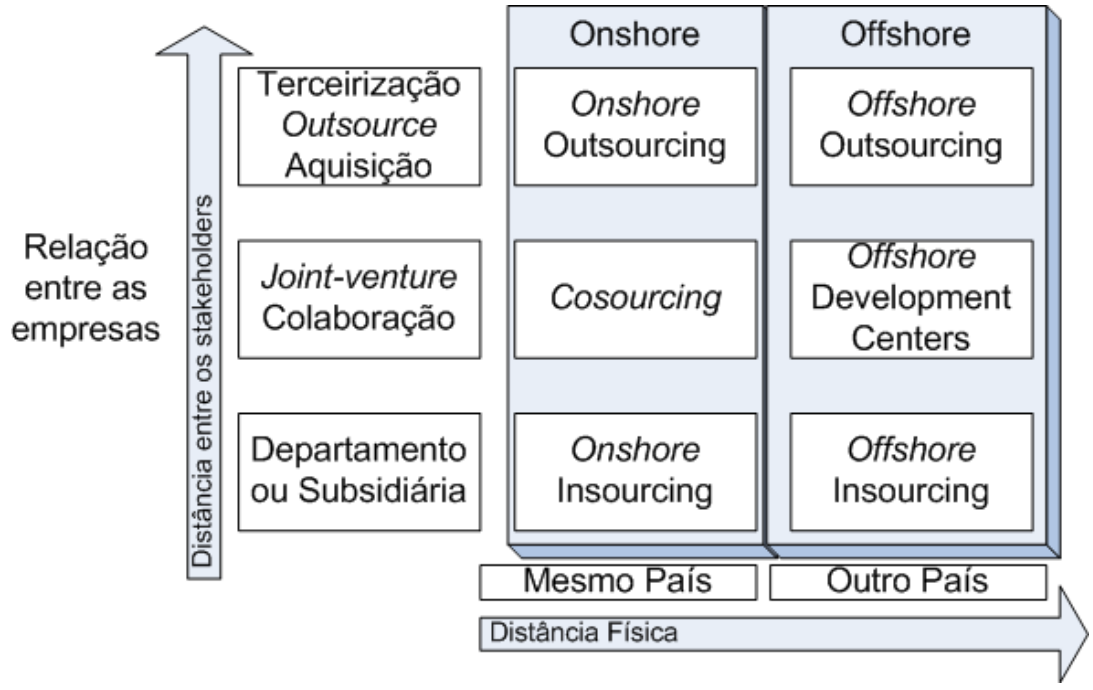

Figura 3 - Relação entre as empresas e distância física, adaptado de Robinson et. al. (2004)

Os modelos apresentados na figura 2, foco deste estudo, são assim definidos por Robinson (2004) e Carmel e Argawall (2005):

- *Onshore insourcing* ou demanda doméstica interna: esse modelo ocorre quando a empresa institui um departamento ou subsidiária interna atendendo às suas próprias demandas de software.
- *Onshore outsourcing* ou *outsourcing:* nesse modelo, a empresa terceiriza uma ou mais atividades para outra empresa, localizada no mesmo país.
- *Offshore outsourcing* ou *offshoring:* A empresa terceiriza uma ou mais atividades para outra empresa, localizada em outro país.
- *Offshore insourcing* ou *captive/internal offshoring:* criação de uma subsidiária da própria empresa para prover serviços de desenvolvimento de software (*insourcing*). Essa subsidiária está necessariamente localizada em um país diferente da matriz da empresa, ou empresa contratante (*offshore*).
- *Cosoursing:* parceria que temo objetivo de compartilhar a realização de atividades entre diversas empresas. É caracterizado pelo *joint-venture*.
- *Offshore development centers:* centros especializados em trabalhos de offshore. São organizações que oferecem serviços de desenvolvimento em um âmbito global.

Segundo Prikladnicki, Audy e Evaristo (2003) o termo *outsourcing* é definido como uma forma de uma empresa contratar outra para desenvolver software, em vez de desenvolver em seu próprio domínio (*in-house*). Para tanto, a empresa concentra seu núcleo de negócios (*core business*) em sua própria organização e repassa o desenvolvimento para outras e, portanto, reduz seu time de desenvolvimento.

O*ffshore outsourcing*, segundo os autores, ocorre quando a empresa desenvolvedora do software localiza-se em outro pais e oferece custos menores do que o desenvolvimento local. A escolha de *outsourcing* não implica em desenvolvimento distribuído, uma vez que o desenvolvimento pode ser efetuado nas instalações da própria organização que contrata o serviço.

Essa operação também é comumente denominada de *offshore*. A definição de *offshore*, segundo Dutta (2005), em termos simples é mover parcial ou totalmente o trabalho para outro país com trabalho mais barato (mais viável).

Dutta e Roy(2005) analisam o modelo de *offshore* e a suas consequências de uma maneira geral. Utiliza um diagrama denominado *loop casual* para ilustrar o modelo. Nesse *loop* há uma relação de causa consequência. Exemplo: a taxa de *offshore* em um país tende a aumentar se há um custo menor de produção em outro país e se há pressão dos lideres para fazer *offshore*. A partir dessas relações e de informações reais, os autores elaboraram equações e efetuaram um experimento computacional de simulação.

Matloff (2005) afirma que a uniformidade de ferramentas auxilia a produtividade em sistemas de *offshore*. Além disso, o autor cita alguns pontos fundamentais em *offshore*. Tais pontos incluem:

- Não estimar economia elevada de custos.
- Manter o trabalho criativo *in-house*, ou seja, não repassar a essência do trabalho, o que efetivamente representa agregação para outra organização desenvolver.
- Detalhar extremamente as especificações.
- O uso de *expertise*, no início, pode reduzir e advertir sobre os problemas de idiomas e culturais.
- Ter em detalhes os currículos de todos os trabalhadores da empresa que presta o serviço.
- Averiguar a questão da propriedade intelectual.

Sobre uma linha similar, o conceito de *nearshore* segundo Sanchez (2006), é caracterizado pela proximidade geográfica entre os mercados fornecedores e consumidor. Nesse cenário, as empresas brasileiras de TI se deparam com boas oportunidades de negócios, já que o Brasil se enquadra como polo potencial de investidores estrangeiros em tal tipo de serviço.

Uma abordagem sobre o ponto de vista de resolução de problemas impõe ao grupo um modelo de GGPS – *General Group Problem Solving*. O modelo, segundo Turner e Fuller (2000), envolve uma série de fatores antecedentes à tomada de

decisão, que levam a um processo de decisão baseada nas características emergentes dos grupos.

2.1.3 Coordenação

Segundo Cataldo (2007), a habilidade de uma organização executar prosperamente suas tarefas depende da combinação apropriada da estrutura organizacional, processos, comunicação e mecanismos de coordenação.

Recentemente, razões empresariais, segundo Sinha e Sengupta (2007), levaram a projetos de software crescente e distribuído. Para Sengupta (2006), desenvolver software em ambientes distribuídos agrega um conjunto de desafios. Além disso, Sinha e Sengupta (2007), afirmam que a inabilidade de comunicar e coordenar dá origem a outros problemas em um ambiente de desenvolvimento distribuído.

Cataldo et al (2007) afirmam que a abordagem de modularização do software é uma ferramenta muito utilizada para dividir o desenvolvimento de um projeto complexo em unidades de gerenciáveis. Entretanto, algumas dependências técnicas permanecem, criando dependência entre tarefas e tornando difícil o gerenciamento.

A visão mais generalista de Burton e Obel (1998) afirmam que uma abordagem de coordenação é o exame da relação entre estrutura organizacional, processos e mecanismos de coordenação. Além disso, a literatura de teoria organizacional sugere que a habilidade de uma organização realizar suas tarefas prosperamente depende da combinação apropriada da estrutura organizacional, processos, comunicação e mecanismos de coordenação.

A coordenação depende da forma na qual uma organização está estruturada. O design organizacional, criado a partir das maneiras nas quais os elementos da organização estão relacionados pode revelar os mecanismos de coordenação.

O design organizacional estratégico identifica, ainda segundo Burton e Obel (1998), o que a organização faz na sua estrutura e não nas tarefas, identifica ainda que

as tarefas específicas dependem de estrutura organizacional. Além disso, especifica as unidades de agrupamentos organizacionais.

Agregadas a esse design, informações como a relação entre as unidades organizacionais, controle, sistemas de incentivo e fluxo de informações, são identificadas e definidas.

Burton e Obel (1998) asseguram que sete elementos constituem o *framework* para a elaboração consistente de um design organizacional. São elas: configuração, complexidade, complacência, centralização, coordenação, comunicação e compensação.

Em um ambiente de Desenvolvimento Distribuído de Software, o mecanismo de gerenciamento é a ferramenta que coordena a produção. Há diversas abordagens sobre o assunto. Em uma visão mais gerencial, autores como Burton e Obel(1998) e Mintzberg(2003) enfatizam o design organizacional como a principal estrutura de controle das organizações. Entretanto, autores como Cataldo et al(2007), Borden; Nett e Wulf(2007) e Sinha; Sengupta e Ghosal (2007) enfatizam as ferramentas técnicas como mecanismos de coordenação.

Essa segunda abordagem, mais técnica, envolve, por exemplo, o gerenciamento da documentação do software, o gerenciamento de configurações e mudanças, mecanismos de atualização de repositórios, ferramentas de comunicação e periodicidade de reuniões.

Ambas as abordagens, técnica e gerencial, exercem forte fluência sobre a produção distribuída de software. Enquanto a primeira estabelece uma estrutura organizacional entre empresas ou departamentos, a segunda abordagem é complementar à primeira, sendo utilizada como uma ferramenta de produtividade.

A coordenação técnica pode ser compreendida como a maneira de interação organizada entre os *stakeholders* de um projeto de DDS, enquanto a coordenação em uma abordagem gerencial é a forma como os *stakeholders* estão relacionados.

A figura 4 é uma representação abstrata desses dois tipos de coordenação detectados na literatura. A coordenação gerencial está relacionada com o design

organizacional, abrangendo-se por todos seus departamentos. A coordenação técnica centra-se na interação entre os sites com a utilização de ferramentas.

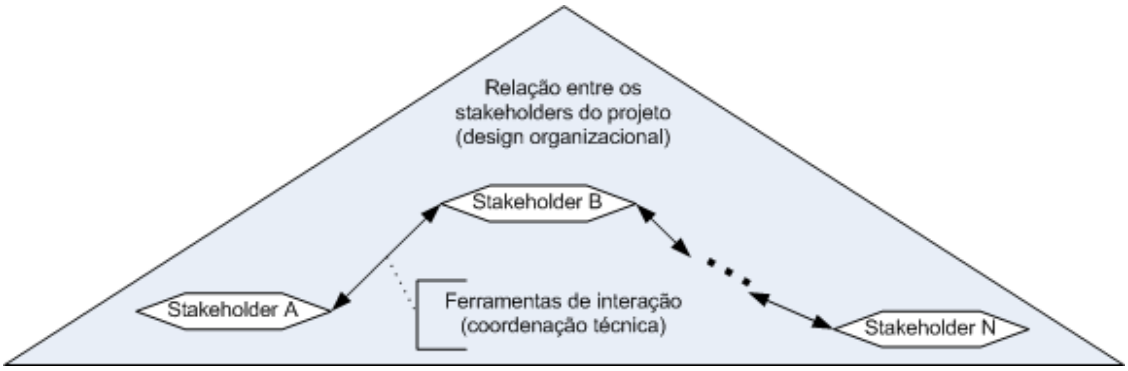

Figura 4 - Relação entre Coordenação Técnica e Coordenação organizacional

2.2 Engenharia simultânea

O objetivo principal da engenharia simultânea é aumentar a lucratividade, diminuindo o tempo de lançamento de novos produtos por meio da construção de um processo de desenvolvimento enxuto e eficaz, segundo Maliniak (1991) apud Facin(1998).

O desenvolvimento distribuído, analisado em perfil mais amplo, abrange outras áreas incluindo o projeto simultâneo de automóveis e outros produtos. Sobre esse perfil, o desenvolvimento distribuído de software absorve os conceitos de engenharia simultânea.

Para Cleetus (1992):

> A engenharia simultânea é uma abordagem sistemática para o desenvolvimento integrado do produto que enfatiza a resposta às expectativas do consumidor e que incorpora valores de equipe tais como: cooperação, confiança e compartilhamento, de tal maneira que a tomada de decisão prossegue com intervalos prolongados de trabalho paralelo, por todas as perspectivas do ciclo de vida a partir do início do processo, sincronizados por trocas comparativamente breves, para produzir consenso.

Para o autor, essa é a melhor definição porque considera o que está de fato envolvido na prática da engenharia simultânea. Outros autores a mencionam: Maliniak

(1991), Schneider (1994) e Zhang H. e Zhang D. (1995). Outras definições de Engenharia Simultânea (ES) são apresentadas por Zhang (1995):

• Desenvolvimento concorrente das funções de projeto do produto, com comunicação aberta e interativa acontecendo entre todos os membros da equipe. O propósito é reduzir o tempo de projeto que vai da concepção até o lançamento do produto.

• É uma abordagem sistemática para gerar todo o potencial dos benefícios esperados pela aplicação de tecnologias avançadas em indústrias. É baseada no desenvolvimento de uma nova ciência e tecnologia de produção. Aplicá-la requer ajuste na organização das empresas, para que se possa usufruir dos benefícios da utilização das tecnologias avançadas. Esse ajuste não somente incorpora o processo desde o projeto de concepção até a produção, mas também envolve a totalidade das atividades de produção da área de manufatura através do ciclo de vida do produto e inclui relacionamentos prolongados com seus parceiros — consumidores e fornecedores.

Segundo Facin (1998), o potencial e as vantagens de um processo de desenvolvimento mais eficaz estão na oportunidade de redução do custo em cada fase de desenvolvimento, reforçados pela possibilidade de extinção dos custos esperados e pela expectativa de que não haja necessidade de um reprojeto.

O esforço de desenvolvimento eficaz durante as primeiras fases do projeto, além de gerar melhores oportunidades de redução de custos no projeto do produto, causa enorme impacto no tempo de comercialização de novos produtos e na qualidade, no custo dos ciclos de produção e distribuição.

Schneider (1994) aponta, entretanto que, no caso da engenharia simultânea, os custos até crescem mais rapidamente devido ao envolvimento mais cedo do pessoal das diversas áreas funcionais, comparando-se com a engenharia tradicional (modelo serial) em que atuam primeiro os profissionais da engenharia de produtos e os demais profissionais são envolvidos somente em fases posteriores. Um exemplo clássico é o dos engenheiros de processo, que, no ciclo tradicional, são envolvidos somente após a liberação do projeto elaborado pelos engenheiros de produto.

Para Facin (1998) muitos afirmam que, embora os custos de desenvolvimento sejam similares em ambos os casos, a adoção da engenharia simultânea traz uma redução dos custos, porque são evitadas as alterações de engenharia e retrabalhos, experimentados no início da produção quando o produto é desenvolvido da maneira tradicional.

De acordo com Adachi; Shih e Enkawa (1994), no aspecto organizacional, muitas das publicações a respeito da redução do tempo de desenvolvimento do produto recomendam, indistintamente, a organização de equipes multifuncionais isoladas da hierarquia das empresas, suprida preferencialmente com mais generalistas do que especialistas, como a chave para redução dos ciclos de tempo de comercialização.

Foreman (1989) afirma que o conceito da Engenharia Simultânea não é um conceito inteiramente novo, porque, por muitos anos, algumas companhias têm obtido vantagem dos benefícios do aumento da comunicação entre as diversas áreas da companhia, desde o início do processo de projeto de um novo produto.

Essa comunicação deve ocorrer entre todos os vários grupos funcionais componentes da organização de uma companhia envolvida com o novo produto e também com o processo de produção que será utilizado.

A mensagem implícita no conceito da engenharia simultânea é que ela é muito benéfica para projetar os processos que serão utilizados para produzir um produto, ao mesmo tempo em que ele está sendo projetado.

Segundo Facin (1998) ensina que desenvolvimentos tanto em hardware como em software são somados à coleção de ferramentas disponíveis para aumentar as capacidades de muitas companhias em utilizar sistemas de engenharia simultânea.

Nesses sistemas, todo trabalho de engenharia desde o projeto até a produção e testes são desenvolvidos eletronicamente em estações de trabalho conectadas. No entanto, para obter a vantagem completa das ferramentas disponíveis, algumas mudanças na organização das companhias e na maneira como as pessoas fazem seus trabalhos são necessárias, como veremos mais tarde.

Logicamente que para a prática da engenharia simultânea, não é essencial que a empresa realize pesados investimentos em softwares e equipamentos de última

geração. Todavia desde que tenha recursos, ela poderá se beneficiar bastante se puder dispor de ferramentas que facilitem e agilizem o processo decisório, principalmente se for uma grande indústria de fato. O avanço tecnológico está cada vez mais favorecendo a integração dos membros da equipe do projeto, mesmo que estejam geograficamente distribuídos.

Um aspecto importante do conceito da engenharia simultânea envolve a integração dos recursos da companhia que desempenham atividades de projeto muito mais cedo no processo de desenvolvimento – para fazer a tarefa certa da primeira vez – do que no processo tradicional. Para implantar a engenharia simultânea efetivamente, Foreman (1989) assegura que é necessário endereçar-se às ferramentas que fazem essa cultura corporativa funcionar. As ferramentas podem ser divididas nas seguintes categorias:

- TQC - *Total Quality Control* (Controle Total da Qualidade)
- CIM - *Computer Integrated Manufacturing* (Produção Integrada por Computador)
- Melhoria da produtividade (*Just-in-Time*)
- Sistemas Humanos

A implementação efetiva de algumas ferramentas fundamentais em cada uma dessas áreas proporcionam à companhia vantagem competitiva, porque diminui custos de projeto e produção, encurta o tempo de projeto e desenvolvimento, reduz o retrabalho e o número de alterações de engenharia.

O autor ainda destaca que existem diversos benefícios que uma companhia pode obter a partir do uso da abordagem da ES:

• O processo usado para fabricar o produto pode ser otimizado desde o início, em vez de ser excessivamente restringido por requisitos de projeto que não levam em conta o processo.

• O produto pode ser projetado para permitir o uso dos níveis e tipos apropriados de técnicas de produção: automatizada ou integrada.

• É possível considerar os planos de investimento, vendas e negócios mais cedo no processo de projeto do produto. Dessa maneira, verifica-se a viabilidade de iniciar o projeto.

• O ambiente de trabalho promove criatividade, proporcionada pelas trocas entre as disciplinas. Tipicamente, pessoas com tipos de especialidade diferentes são colocadas em grande proximidade.

• Administradores têm informações mais completas e um melhor controle das atividades entre as disciplinas. O império dos administradores têm se expandido incluindo todas as atividades relacionadas a um dado produto, ao invés de uma função em particular, isso acontece porque as equipes passaram a trabalhar por projeto. Isso elimina muitas barreiras políticas que costumavam existir.

Womack; Jones e Ross (1991) apontam que todas as grandes empresas automobilísticas sempre têm o mesmo problema ao desenvolverem um novo produto. O problema é como fazer com que os diversos departamentos funcionais – marketing, engenharia de motores, engenharia de carroceria, engenharia de chassis, engenharia de processos e operações fabris – colaborem, por um extenso período de tempo, para desenvolver um novo veículo com sucesso. Eis aqui uma área que a engenharia simultânea pode ser usada.

Facin (1998) faz uma abordagem sobre como os sistemas computacionais auxiliam a engenharia simultânea. A abordagem indica sistemas de informações como ferramenta fundamental de apoio ao processo do produto. A autora afirma que, segundo Reddy et al (1993), uma arquitetura em níveis para os diferentes tipos de ferramenta computacionais (banco de dados, sistema operacional, rede de comunicação, multimídia, aplicativos específicos etc.) pode ser implementada para integrar todas as ferramentas e proporcionar um ambiente verdadeiramente colaborativo e controlável. Estes níveis são cinco:

1. Nível da atividade:

Representa as atividades empreendidas pelas equipes multidisciplinares. Uma equipe está envolvida no ciclo contínuo de planejar, implementar, monitorar e melhorar o conjunto de atividades vitais para o sucesso de um dado projeto. Nesse nível, a preocupação é com o gerenciamento do projeto.

2. Nível da transação:

Para compreender como são os trabalhos individuais em um ambiente integrado por computador em grandes organizações, consideram-se seis atividades

fundamentais desempenhadas por membros de uma equipe nesse nível. Elas são: observar, computar, comunicar, negociar, decidir e arquivar. Essas simples transações tornam-se poderosas quando desempenhadas no contexto de um ambiente heterogeneamente distribuído. Os tipos de ferramentas encontradas neste nível são as de CAD/CAM/CAE, por exemplo.

3. Nível dos serviços de colaboração:

Envolve uma variedade de serviços para dar suporte às transações citadas anteriormente e também às atividades do dia a dia dos membros da equipe. Esses serviços podem ter as seguintes categorias: comunicação, coordenação, troca de informações, histórico e integração.

a. Comunicação:

A falta de comunicação entre membros da equipe geograficamente dispersos é o maior obstáculo para o trabalho cooperativo. A comunicação eficaz entre os membros pode ser facilitada pelo uso de aplicações como equipamentos multimídia para conferências usando textos, gráficos, áudio e vídeo compartilhados por todos.

b. Coordenação:

A coordenação é crítica para o funcionamento eficaz das equipes de desenvolvimento multidisciplinares. Os membros das equipes devem influenciar uns aos outros para que seja feito um produto com alta qualidade. A coordenação proporciona um suporte computacional a fim de auxiliar a tomada de decisão e as negociações das soluções em uma rede dispersa geograficamente. À categoria estão associadas a visão comum das atividades e dados, as atividades de planejamento e programação, a notificação de alterações aos membros da equipe e a administração das restrições nas múltiplas perspectivas.

c. Troca de informações:

A informação gerada é armazenada em formatos de dados heterogêneos e em várias bases de dados espalhadas na empresa. A troca de informação implica desenvolvimento de representações comuns de dados e acesso transparente em um sistema heterogeneamente distribuído. Porque problemas surgem na administração de dados repetidos, versões e controle simultâneo, a administração das alterações é muito importante neste nível.

d. Histórico:

Em um ambiente de engenharia simultânea, é desejável registrar eletronicamente as intenções do projeto e a evolução de um produto desde o projeto conceitual até a obsolescência. 0 histórico do projeto é útil para futuros projetos e para documentação dos já existentes. Indexar, correlacionar e armazenar os vários tipos de documentos (de projeto, de produção, especificações etc.), e arquivar decisões conseguidas em reuniões entre os projetistas são alguns dos problemas que necessitam ser avaliados nesse contexto.

e. Integração:

Esse serviço une todos os demais, facilita o acesso às ferramentas de engenharia e serviços de uma maneira transparente na empresa, proporciona mecanismos para descrever quais as ferramentas disponíveis para os usuários e de que forma, além de oferecer mecanismos para explorar a infraestrutura de comunicação.

4. Nível do modelo de informação da empresa:

Os serviços apresentados tratam da viabilidade da informação na empresa. O acesso a tal informação é facilitado pela padronização, que caracteriza a maneira como um produto é construído, o processo adotado para produzi-lo e os recursos disponíveis na organização.

5. Nível da rede de comunicação:

Uma rede de computadores é a base para implementar um ambiente para aplicação da engenharia simultânea com equipes de trabalho reconhecidas como virtuais por causa da separação física. Esse nível abrange a tecnologia de comunicações e a computação distribuída.

2.3 Redes de Petri

Um sistema pode ser especificado com a utilização de alguns conceitos matemáticos como a teoria dos conjuntos, a lógica, a álgebra abstrata, a teoria das categorias, a teoria dos domínios etc. Segundo Martins (1988), utilizando essas ferramentas é possível construir sistemas formais e estudar certas propriedades desses sistemas: consistência, coerência e completeza.

Segundo Ribeiro (2000), os métodos formais de especificação originaram-se para apoiar a construção de máquinas abstratas que representam o comportamento do sistema. Desse modo, a semântica (comportamento) de um sistema era dada pela definição dessa máquina abstrata com estados discretos e por sequências de operações computacionais que modificavam o estado da máquina. Exemplos dessa abordagem, chamada operacional, são as máquinas de estados finitos e a VDL (Vienna Definition Language), explicada por Berg em 1982.

Os conceitos de Redes de Petri (PN – Petri Network) foram introduzidos por Carl Adam Petri em sua tese de doutorado (1962), intitulada ***Kommunikation mit Automaten*** na faculdade de Matemática e Física da Universidade de *Darmstadt*, Alemanha, como ferramenta para descrever relações entre condições e eventos no estudo de protocolos de comunicação entre componentes assíncronos.

Embora ocorresse uma ampla divulgação acadêmica ao longo de três décadas, o seu potencial só foi reconhecido na metade da década de 1980, quando essa teoria foi usada para implementações práticas nas áreas de informática e manufatura devido à disponibilidade de novos recursos de hardware e software.

Segundo Raposo (2000), a PN é uma ferramenta de modelagem aplicável a uma série de sistemas, especialmente àqueles com eventos concorrentes. É ferramenta matemática e gráfica que oferece ambiente uniforme para modelagem, análise e projeto de sistemas a eventos discretos. Sua aplicação tem se estendido a uma grande quantidade e variedade de sistemas. Os principais sistemas em que é aplicada são: sistemas de comunicações, sistemas de software, sistemas de processamento de informação, além das aplicações em modelagem, simulação e sequenciação de sistemas flexíveis de manufatura.

O principal objetivo das PN é modelar o comportamento de um sistema a partir de seus estados e mudanças. Redes de Petri sãos formadas por lugares, transições e arcos. Hasegawa (1996) destaca as seguintes características e vantagens da técnica:

- Representa a dinâmica e a estrutura do sistema segundo o nível de detalhamento desejado.
- ldentifica estados e ações de modo claro e explícito, facilitando com isso a monitoração do sistema em tempo real.
- É capaz de representar de forma natural as características de sincronização, assincronismo, concorrência, causalidade, conflito e compartilhamento de recursos.
- Constitui-se de uma teoria muito bem fundamentada para a verificação de propriedades qualitativas.
- Possui uma semântica formal e precisa que permite ao mesmo modelo ser utilizado tanto para a análise de propriedades comportamentais (análise quantitativa e/ou qualitativa) e avaliação do desempenho, quanto para a construção de simuladores a eventos discretos e controladores (para implementar ou gerar códigos para controle de sistemas). Verificar, ainda, comportamentos indesejáveis como bloqueio, limitação etc.
- Incorpora conceitos de modelagem do tipo refinamento (*top-down)* e do tipo composição modular (*botom-up*) através de técnicas como: modularização, reutilização, refinamento etc.

Como ferramenta matemática, um modelo em rede de Petri pode ser descrito por um sistema de equações lineares, ou outros modelos matemáticos que refletem o comportamento do sistema, o qual possibilita sua análise formal. Tal característica permite realizar a verificação formal das propriedades comportamentais do sistema. Há várias extensões do modelo clássico das redes de Petri. Entre elas, segundo Maciel e Cunha (1996):

- Redes de Petri coloridas – *tokens* individualizados por meio de cores atribuídas a eles;
- Redes de Petri hierarquizadas – permitem o agrupamento ou refinamento de partes do modelo;
- Redes de Petri temporizada determinística – adicionam o conceito de tempo ao modelo.

As transições podem ser disparadas a partir do momento em que estão habilitadas, quando o evento associado a ela ocorrer. Uma transição está habilitada quando cada um de seus lugares de entrada possuir a quantidade de *tokens* correspondente ao peso do arco a ele ligado. Por definição, um arco não marcado possui peso igual a 1.

A figura 5 mostra um exemplo simples de uma rede de Petri e seus componentes. Uma PN é composta por lugares e transições. Graficamente, os lugares são representados com círculos e as transições com barras ou retângulos. Os arcos são rotulados com seus pesos (exceto se o peso for igual a um), e um arco de peso k pode ser interpretado como k arcos paralelos. Uma transição possui certo número de lugares de entrada e saída, o que representa as pré e pós-condições do evento observado.

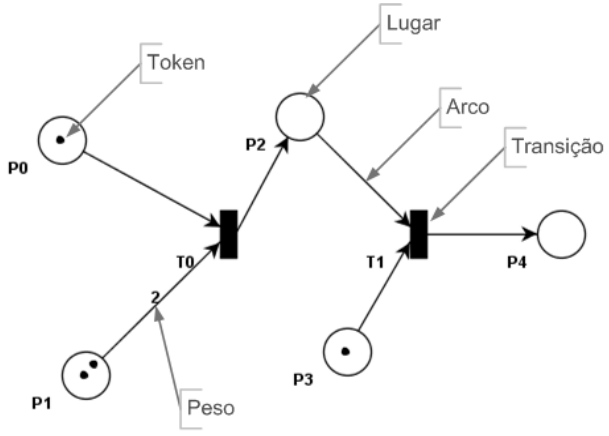

Figura 5 - Exemplo de uma rede de Petri

As PNs oferecem ainda algumas características de análise do sistema modelado. De maneira geral, há PNs simples com maior poder de modelagem e extensões de PNs com maior poder de decisão (RAPOSO, 2000).

O objetivo de modelar um sistema com redes de Petri é analisar as propriedades do sistema e os problemas nele contidos. A análise é importante para verificar, por exemplo, se o sistema possui *deadlock*[1], se é possível reverter estados, se há algum estado que nunca será alcançado ou até mesmo análise de desempenho, com a utilização de PNs associadas ao tempo.

2.3.1 Formalismo das redes de Petri

Redes de Petri são grafos direcionados e bipartidos que podem ou não possuir um estado inicial. Cardoso e Valette (1997) apresentam a PN como um modelo formal, de três maneiras diferentes: como um grafo, como um conjunto de matrizes e como um sistema de regras.

A diferença entre essas representações restringem a apenas forma de apresentação, sendo que em qualquer um dos modos é possível verificar se as transições são paralelas ou possuem conflito, se uma transição está ou não habilitada e, também, efetuar o disparo de uma transição habilitada, fazendo a rede evoluir.

A representação gráfica pode ser a mais vantajosa, entretanto sua integridade é provida pelo formalismo que mapeia o comportamento do sistema modelado.

2.3.2 Definição formal

Redes de Petri é definida formalmente por Murata (1989), como uma quíntupla **PN = (P, T, F, W, Mo)**, em que:

[1] Deadlock é definido por Silverschatz (2008) como impasse causado pela dependência e alocação de recursos.

P é um conjunto finito de lugares,
T é um conjunto finito de transições,
F é um conjunto de arcos,
W é uma função de peso,
M é a marcação inicial e,
$P \cap T = \varnothing$ e $P \cup T \neq \varnothing$.

A estrutura de PN, N = (P, T, F, W) sem um estado inicial específico é denotada por N. A PN com estado inicial determinado é denotada por (N, Mo).

O comportamento de muitos sistemas pode ser descrito como estados e suas mudanças. A simulação do comportamento dinâmico de um sistema acontece com o disparo das transições, quando os estados ou marcações da PN correspondente são modificados. Uma ou mais marcações **k** podem estar associadas a um estado **p.**

O estado de um sistema, segundo Cardoso e Valette (1997), é dado pelas quantidades de marcas nos lugares da PN. Cada lugar representa um estado parcial do sistema e suas transições são associadas a eventos que ocorrem no sistema real. Murata (1989) explica que as transições são acionadas quando o evento ocorre, os eventos anteriores, juntamente com as transições habilitam seu disparo, repassando seus devidos pesos para o próximo estado.

2.3.3 Propriedades das PNs

Redes de Petri podem fornecer excelente suporte à análise de muitas propriedades e problemas associados a sistemas concorrentes. As propriedades dependentes de um estado inicial são mais comumente chamadas de Propriedades Comportamentais. Algumas delas são alcançabilidade, vivacidade, reversibilidade, limitação, abrangência e persistência.

A alcançabilidade *(Reachability)* é a propriedade é fundamental para compreender o dinamismo de qualquer sistema. A marcação em uma PN é considerada alcançável se há uma sequencia finita de disparos que conduza a ela.

Se a rede de Petri permite um número limitado de marcações, então é considerada k-limitada. Se é uma PN 1-limitada então, é denominada de segura.

A vivacidade (*Liveness)* indica segundo Cardoso e Valette (1997), se, e somente se, todas as transições de uma PN são ativas. Uma rede que possui essa propriedade garante que nenhum bloqueio pode ser provocado por sua própria estrutura. A vivacidade de uma rede garante a ausência de partes que nunca serão atingidas.

Uma PN é reversível *(Reversibility)* se a marcação inicial é alcançável de toda marcação alcançada. Na maioria das vezes, não é necessário voltar ao estado inicial do sistema, desde que se possa voltar a algum outro estado.

Uma rede de Petri é dita coberta (*Coverability*) se seus estados são potencialmente dísparáveis.

Uma PN é chamada persistente se, o disparo de uma transição não desabilita outra.

2.4 Considerações finais

As teorias explanadas neste capítulo serviram de anteparo para que este projeto fosse elaborado. Cada teoria em particular, colaborou com este projeto em algum aspecto. Tais aspectos foram essenciais para a consolidação da metodologia de pesquisa (Capítulo 3), principalmente com relação à realimentação das teorias utilizadas no protocolo de pesquisa. Essas teorias realimentaram também a consolidação e validação dos estudos casos que compõem o capítulo 4.

A contribuição cientifica deste capítulo, para este trabalho de pesquisa está elucidada na tabela 1, que explica quais as principais colaborações de cada teoria para o Modelo de dinâmica de desenvolvimento distribuído de software (M3DS) proposto no capítulo 5.

Tabela 1 - Colaboração de cada teoria para o M3DS

Teoria	Colaboração para este projeto
Desenvolvimento distribuído de software	Teoria base para a compreensão dos estudos de caso apresentados na literatura e o que cada um deles explorou metodicamente. Compreensão do funcionamento geral das estruturas de DDS. Caracterização do desenvolvimento distribuído de software de acordo com a relação entre as empresas/organizações e a distancia entre elas. Coordenação em ambientes de desenvolvimento distribuído de software, do ponto de vista organizacional e técnico. Compreensão dos problemas oriundos do desenvolvimento distribuído. Elaboração de um modelo de dinâmica de desenvolvimento distribuído de software inicial.
Engenharia simultânea	Compreensão da codificação e projeto simultâneo e distribuído de software. Como os ajustes oriundos da colaboração são abordados e como são constituídos os processos de engenharia simultânea de software. Colaboração e requisitos necessários para efetuação de ambientes de engenharia simultânea. Compreender como funciona o fluxo de informação em projetos simultâneos e distribuídos
Especificações formais e Redes de Petri	Especificação do M3DS. Compreensão do modelo M3DS como um modelo em redes de Petri. Verificação e especificação do modelo de dinâmica de desenvolvimento distribuído de software.

CAPÍTULO 3 – METODOLOGIA

A metodologia determina qual o modelo de solução empregado sobre um problema que se apresenta sob um determinado âmbito com um escopo definido e uma profundidade estabelecida. Este capítulo apresenta a metodologia adotada para este trabalho. As próximas seções descrevem o posicionamento da tese perante o Departamento de Engenharia de Produção, uma visão geral de metodologia de pesquisa, a justificativa da metodologia adotada e o protocolo de pesquisa.

3.1 Posicionamento da tese

Estabelecer o posicionamento da tese fundamentalmente determina área em que o trabalho é enquadrado. Não é intuito desta seção determinar a abordagem alcançada por este trabalho e sim apresentar uma visão do ambiente na qual a pesquisa está em andamento, contextualizando seu posicionamento perante o departamento de Engenharia de Produção da Escola Politécnica da Universidade de São Paulo.

Este trabalho foi desenvolvido dentro do GTI, uma das áreas de pesquisa do departamento. Dentro do departamento de Engenharia de Produção, encontram-se as seguintes áreas de pesquisa:

- EPEF: Economia da Produção e Engenharia Financeira
- GOL: Gestão de Operações e Logística
- TTO: Trabalho Tecnologia e Organização
- QEP: Qualidade e Engenharia do Produto
- GTI: Gestão da Tecnologia da Informação

3.1.1 O projeto dentro da engenharia de produção

O esquema proposto por Lovejoy, citada por Nakano (1996), se constitui de uma pirâmide cuja base é composta pelas disciplinas de Física, Psicologia social e Filosofia, é empregado neste documento como referência para localizar o projeto na Engenharia de Produção.

Na pirâmide, as três disciplinas são consideradas a base do ensino de Gestão de Operações. A figura 6 ilustra onde está localizado o projeto Rede de Produção de Software

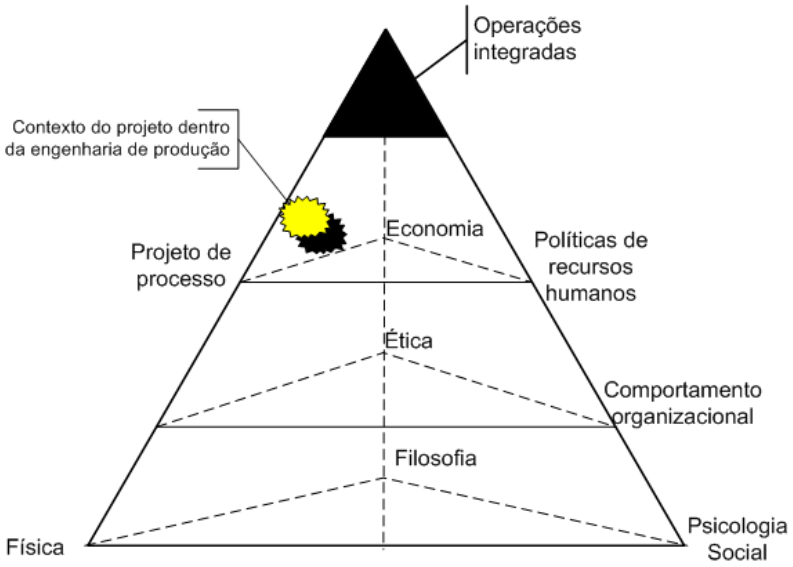

Figura 6- Posicionamento da pesquisa dentro da Eng. de Produção, ajustado para este trabalho e adaptado de Nakano (1996)

Este projeto se enquadra na Engenharia de Produção, principalmente pelo fato de utilizar processos e políticas de relacionamento entre corporações. Dentro de um aspecto mais amplo, abrange ambientes de desenvolvimento distribuído de software.

3.2 Metodologia científica

A pesquisa científica é o modo de construir o conhecimento científico. Para tanto, seu objetivo é compreender, encontrar, selecionar, estruturar e resolver problemas que interessam ao ser humano. Consequentemente, é necessário que o pesquisador elabore hipóteses, modelos e teorias.

O produto essencial de uma pesquisa científica é o conhecimento que pretende ser seguro e certo, apto a operar soluções para um problema humanamente relevante.

A pesquisa científica deve ser conduzida de acordo com um método. Segundo Galliano (1979) método é um conjunto de etapas ordenadamente dispostas a

serem vencidas na investigação da verdade, no estudo de uma ciência ou para alcançar um determinado fim. O autor afirma que são necessárias técnicas para executar as atividades definidas no método de pesquisa.

A figura 7 indica qual a relação entre o problema, o método científico e a solução.

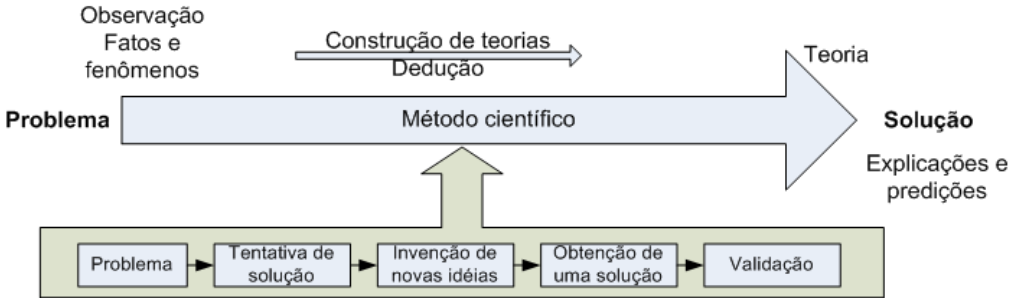

Figura 7 - Pesquisa científica, adaptado de Galliano (1979)

Para Lakatos e Marconi (2005), o método científico é o modo de entender logicamente como o processo da pesquisa científica deve ser conduzido. Lakatos afirma que um método científico deve cumprir um conjunto de etapas que engloba desde a definição do problema, procura pelo conhecimento, tentativa de solução e validação da solução.

Do ponto de vista dos artefatos redigidos, a essência do método científico é um problema verbalizado, que inicia o processo da pesquisa. A próxima etapa é a revisão da literatura, definindo o referencial teórico para a pesquisa. Esse referencial é adicionado a todo o ser do cientista, o qual parte, então, para o processo da pesquisa.

Nesse momento, o pesquisador estabelece uma estratégia, um projeto, um delineamento para o processo da pesquisa. Finalmente, o resultado da investigação é apresentado em uma obra escrita para mostrar a solução científica do problema.

Do ponto de vista das etapas de trabalho, Yin (2005) apresenta esta essência como: definição do problema, design ou delineamento de pesquisa, obtenção dos dados, análise dos dados e redação.

Yin ainda define delineamento de pesquisa como a lógica que liga os dados, informações e fatos coletados (e as conclusões construídas) às questões iniciais, ao problema da pesquisa. Delineamento de pesquisa é o método científico particularizado

para uma pesquisa científica em especial. Todo estudo científico tem um projeto de pesquisa, explícito ou não.

Um projeto de pesquisa é o procedimento lógico de coletar, analisar e interpretar observações. Permite ao pesquisador extrair inferências sobre as relações causais entre os fatores e/ou as variáveis sob investigação. O delineamento de pesquisa também define o domínio de generalização, isto é, para quais situações, ou para qual população de eventos a teoria permanece válida. Um projeto de pesquisa responde, pelo menos, as seguintes questões: quais questões e problemas estudar, quais dados são relevantes, como coletar os dados e como analisar e interpretar os resultados.

O modo de conduzir a pesquisa científica é denominado delineamento de pesquisa. Há, portanto, diferentes tipos em função do contexto em que a pesquisa está, e em função do entendimento epistemológico do pesquisador. Vários autores têm procurado classificar os tipos de processos de pesquisa possíveis ou utilizados. Os principais delineamentos encontrados na literatura serão elencados nos próximos tópicos.

3.2.1 Pesquisa bibliográfica e investigação histórica

Segundo Fachin (2001) esses tipos de delineamento são empregados quando o objetivo é explicar um problema, utilizando o conhecimento disponível a partir das teorias publicadas em livros, periódicos e obras congêneres. É comum o pesquisador utilizar esse método, antes mesmo de sua própria pesquisa. Nesse caso, o objetivo é encontrar na literatura disponível uma forma de familiarização com o tema de pesquisa e com os problemas relacionados na sua área de atuação. Quando esse delineamento de pesquisa é alocado antes da pesquisa em si, é denominada revisão de literatura ou revisão bibliográfica

Na revisão bibliográfica, o pesquisador constrói seu conhecimento acerca de seu trabalho de pesquisa. Tornando suas referências explícitas, o pesquisador ilustra o caminho e as bases teóricas percorridas até concluir sua teoria. Isso favorece a continuação de seu trabalho por outros pois, não somente a teoria/solução final é

apresentada, mas também todo o trajeto, todos os problemas e bases científicas extraídas em publicações, que o pesquisador considera como essencial e correto para seu contexto, são demonstrados nesta etapa.

3.2.2 Pesquisa Descritiva, Não experimental e Exploratória.

São delineamentos que não se prestam a inferências analíticas ou estatísticas, servindo apenas para registrar impressões ou avaliações de fenômenos ou processos. Em rigor, tais delineamentos não seriam delineamentos de pesquisa, já que não podem ser utilizados nem para construção de teorias e modelos, nem para sua verificação. Seriam meros repositórios de dados, com algum texto explicativo.

A pesquisa descritiva descreve as características das relações entre variáveis, de uma determinada população ou fenômeno. Para tanto, emprega técnicas padronizadas de coleta de dados como o questionário e observação sistemática. O tipo de pesquisa descritiva busca a resolução de problemas, melhorando as práticas por meio da observação, análise e descrições objetivas, utiliza entrevistas com peritos para a padronização de técnicas e validação de conteúdo.

3.2.3 Levantamento amostral ou Survey

A tradução da palavra *Survey* de origem inglesa para o português é amostragem, entretanto neste contexto, a tradução mais adequada seria levantamento amostral.

O levantamento amostral, segundo Czaja e Blair(2005) é o processo de coletar informações sobre alguns dos individuais de uma dada população real. A legitimidade das análises e interpretações dos dados provindos do *survey* pode ser conferida pela estatística. A coleta dessas informações no levantamento amostral é feita com questionários, planilhas, folhas de verificação, entre outros instrumentos possíveis.

Caso adotado,há algumas recomendações para elaboração de um questionário adequado, tais como elaborar questões de múltipla escolha em questões não constrangedoras, de múltipla escolha e em número considerável.

3.2.4 Experimentação

Também chamada de pesquisa experimental, a ideia básica da experimentação é o controle e mensuração. O pesquisador deve controlar determinística e/ou probabilisticamente todas as causas sobre os efeitos de interesse. Fachin (2001) afirma que as variáveis são manipuladas de maneira pré-estabelecida e seus efeitos são suficientemente controlados e conhecidos pelo pesquisador.

Se a experimentação é empregada em um contexto onde há fatores aleatórios atuantes, é necessário que a estatística seja utilizada para a interpretação. Consiste essencialmente em estabelecer um domínio de pesquisa, em que variáveis serão postas a se realizarem de modo controlado e serão avaliadas objetivamente.

A ideia de experimento também abrange o conceito de simulação. Simulação é um experimento realizado num modelo físico representativo de processo real. A qualidade das inferências retiradas da simulação dependerá, essencialmente, da qualidade do modelo físico e de quanto ele se aproxima da realidade.

3.2.5 Estudo de caso

Esse delineamento é de caráter exploratório. Seu objetivo é estabelecer uma plataforma para ajudar o discernimento na recepção de intuições sobre a teorização de um dado problema. Um estudo de caso, segundo Fachin (2001), objetiva principalmente ser a experiência prática refinando a maneira de pensar empírico-indutivo para a construção de teorias.

Nesse delineamento, o pesquisador precisa mergulhar em uma situação real, acompanhando-a, sentindo-a, pensando-a, analisando-a, e finalmente apresentar uma teoria que a descreva e a explique. O método não é adequado para generalizações estatísticas, sendo empregado apenas para generalizações analíticas

Yin (2005) define estudo de caso como uma lógica de planejamento de pesquisa empírica que investiga um fenômeno contemporâneo em seu contexto real, especialmente quando os limites entre fenômeno e contexto não são claramente evidentes.

Demais delineamentos semelhantes ao estudo de caso separam o fenômeno do contexto. É o que ocorre com os delineamentos *survey* e experimentação. Em um experimento, há uma separação nítida entre o fenômeno e o contexto enquanto o *survey* não investiga profundamente o contexto do fenômeno. Segundo Yin, estudo de caso, aborda pesquisas em que há mais variáveis de interesse que dados.

3.2.6 Pesquisa-ação

Segundo Westbrook (1995), a pesquisa-ação tem o duplo propósito de auxiliar a reflexão, formulação ou implementação da ação e de desenvolver, enriquecer ou testar quadros referenciais teóricos ou modelos relevantes ao fenômeno em estudo.

Há uma relação ativa e explicita entre os pesquisadores e os responsáveis pela ação em uma área específica. É um método de pesquisa científica em que o pesquisador é atuante e até provável causador dos seus fenômenos de estudo dentro de um contexto definido. Consequentemente, o pesquisador tem influência sobre seu ambiente de pesquisa. Essa característica é o que diferencia principalmente a pesquisa-ação do estudo de caso.

Elden e Chisholn (1993) afirmam que o método da pesquisa-ação objetiva o atendimento a construção ou teste de um referencial teórico voltado à compreensão e ao entendimento de um aspecto da vida humana. A pesquisa ação também tem o objetivo de solucionar um problema prático com uma ação implementada simultaneamente à contemplação teórica.

3.2.7 Pesquisa participante

A pesquisa participante, segundo Brandão (1987), é uma investigação social por meio do qual se busca plena participação da comunidade na análise de sua própria realidade, com objetivo de promover a participação

Essa modalidade de estratégia investigativa científica pode ser considerada, como uma alternativa intermediária entre o estudo de caso e a pesquisa-ação. Em um estudo de caso, o investigador, apesar de envolver na situação, não exerce influência sobre ela, permanecendo como um elemento estranho e oculto ao ambiente de pesquisa. Em uma pesquisa-ação, espera-se que o investigador participe e intervenha de um modo explícito no sistema que ele se propõe a conhecer cientificamente.

Na pesquisa participante, o pesquisador faz parte de sua própria população amostral de pesquisa. É membro ativo dos elementos envolvidos na pesquisa, influenciando e sofrendo influência em seu objeto de pesquisa.

A tabela 2 é uma comparação entre as diversas estratégias de delineamento da pesquisa apresentadas neste capítulo. A coluna intitulada como "controla eventos comportamentais", refere-se à necessidade da estratégia em controlar eventos de seu ambiente de pesquisa.

Tabela 2 - Comparação entre as diversas estratégias de pesquisa, adaptado de Yin (2005)

Estratégia de delineamento	Forma de questão da pesquisa	Abordagem principal	Exige controle sobre eventos comportamentais	Focaliza acontecimentos contemporâneos	Construção ou descoberta	Confirmação ou justificação
Experimento	**Como, por que**	**Quantitativo**	**Sim**	**Sim**	**Não**	**Sim**
Survey	**Quem, o que, onde, quantos, quanto**	**Quantitativo**	**Não**	**Sim**	**Não**	**Sim**
Estudo de caso	**Como, por que**	**Qualitativo**	**Não**	**Sim**	**Sim**	**Não**
Pesquisa participante	**Como, por que**	**Qualitativo**	**Não**	**Não**	**Sim**	**Não**
Pesquisa-ação	**Como**	**Qualitativo**	**Não**	**Não**	**Sim**	**Não**
Pesquisa descritiva	**Como, por que, quem**	**Qualitativo/Quantitativo**	**Não**	**Sim/Não**	**Não / Sim**	**Sim**

A coluna da tabela 2 intitulada como "construção ou descoberta" refere-se à capacidade da estratégia em construir teoria ou descobrir eventos não previstos no início das atividades de pesquisa. O estudo de caso, por exemplo, pode identificar durante a pesquisa um fato novo, gerando uma nova teoria. A última coluna desta mesma tabela refere-se à capacidade da estratégia para confirmar ou justificar uma

dada teoria. O experimento pode ser empregado perfeitamente neste caso, sendo utilizado para isolar o fenômeno de seu contexto e confirmar ou justificar uma teoria. Esta tabela é baseada em Berto e Nakano (2001) e Yin (2005).

3.3 Metodologia adotada

A metodologia empregada neste trabalho é o estudo de múltiplos casos que consiste na aplicação do protocolo do delineamento estudo de caso em várias organizações que praticam o desenvolvimento distribuído de software.

O estudo de caso pode tratar mais variáveis de interesse do que dados e, por isso, é necessário apresentar pressupostos teóricos, visões de mundo e vida particulares para completar ou preencher a lacuna de informações.

Santos (2001) indica a metodologia para selecionar um objeto de pesquisa restrito, com o objetivo de aprofundar-lhe os aspectos característicos cujo objeto pode ser qualquer fato/fenômeno individual ou um de seus aspectos. Exige do pesquisador grande equilíbrio intelectual e capacidade de observação, além de parcimônia quanto à generalização dos resultados.

Uma definição semelhante é dada por Robson (2002), para o qual um estudo de caso é uma estratégia para pesquisar que envolve investigação empírica de um fenômeno particular, contemporâneo, dentro de seu contexto real de vida utilizando múltiplas fontes de evidência.

Um experimento ou um levantamento, segundo Robson não é somente um experimento ou um levantamento, mas também, necessariamente, um "caso" de pesquisa.

Por fim, Robson propõe algumas técnicas para coleta de dados sobre um estudo de caso. Essas são agrupadas em categorias: observação, participante, sistemática, simples, entrevista, aberta, enfocada, estruturada, uso de documentos e registros.

O estudo de caso trabalha com o **como** e o **porquê** dos fenômenos, sem exigir controle sobre os eventos comportamentais. Avalia eventos contemporâneos e

possibilita a replicação literal do estudo de caso, sobre o aspecto de resultados semelhantes.

Os casos abordados neste projeto de pesquisa englobam grandes empresas e/ou grandes projetos distribuídos de software. Separar o contexto do projeto de software sem compreender a razão pela qual a empresa desenvolve de forma distribuída pode fazer com que a veracidade do modelo final seja reduzida. Por isso, o estudo de caso foi o método de pesquisa que mais se alinhou com o problema deste trabalho.

Os demais métodos apresentam empecilhos críticos de modo que não são recomendados para o perfil deste projeto.

O experimento não foi utilizado, é difícil construir um modelo próximo ao real e simular as mesmas situações cotidianas que ocorrem nas organizações de DDS. Os projetos analisados são reais e grandes, envolvendo um grande número de desenvolvedores. Além disso, os casos analisados são exemplos de GSD. Por isso, é difícil recrutar e treinar um grande número de pessoas devido ao custo e ao prazo de pesquisa.

Embora um viés deste trabalho tenha empregado pesquisa-ação (L'ERARIO; PESSOA, 2008), imergir em um projeto/empresa de desenvolvimento distribuído de software trata particularmente de uma organização. Isso pode inviabilizar a identificação de variáveis importantes, relativa a demais casos, que contribuem para a elaboração de um modelo de desenvolvimento distribuído mais consolidado e generalista. Além disso, a participação efetiva em um ambiente desse tipo é complexa.

Apesar de ser possível, segundo L'Erario e Pessoa (2008), criar um ambiente de DDS utilizando Universidades fisicamente distantes, é improvável que um ambiente criado nesta natureza seja tão fiel quanto os das grandes organizações de desenvolvimento de software.

O *survey* não revela relações que podem ser identificadas no processo de pesquisa. Não foi adotado neste trabalho, pois não é um método adequado para trabalhar analítica e detalhadamente um ambiente de pesquisa e procurar novas variáveis não detectadas no início da pesquisa.

Na pesquisa participante, seria necessário o pesquisador imergir na organização, exercendo influência e manipulando os processos de desenvolvimento, porém as organizações avaliadas já têm um conjunto de processos padronizados, empregados em um âmbito global. Além disso, a influência do pesquisador sobre a empresa pode comprometer os processos que a empresa julga adequados, isso as levaria a rejeitar interações externas.

A imersão do pesquisador em uma organização de produção de software também exigiria treinamento nos processos da organização, tornando o pesquisador papel ativo na produção do software. Tal imersão não abordaria uma visão macro, generalizando a própria organização, mas uma abordagem sistemática dos problemas que uma empresa particular enfrenta cotidianamente. Dessa maneira, o escopo desta pesquisa seria modificado.

A figura 8 representa o roteiro da pesquisa a ser efetivada. As setas representam o fluxo de informações decorrentes durante o trabalho de pesquisa.

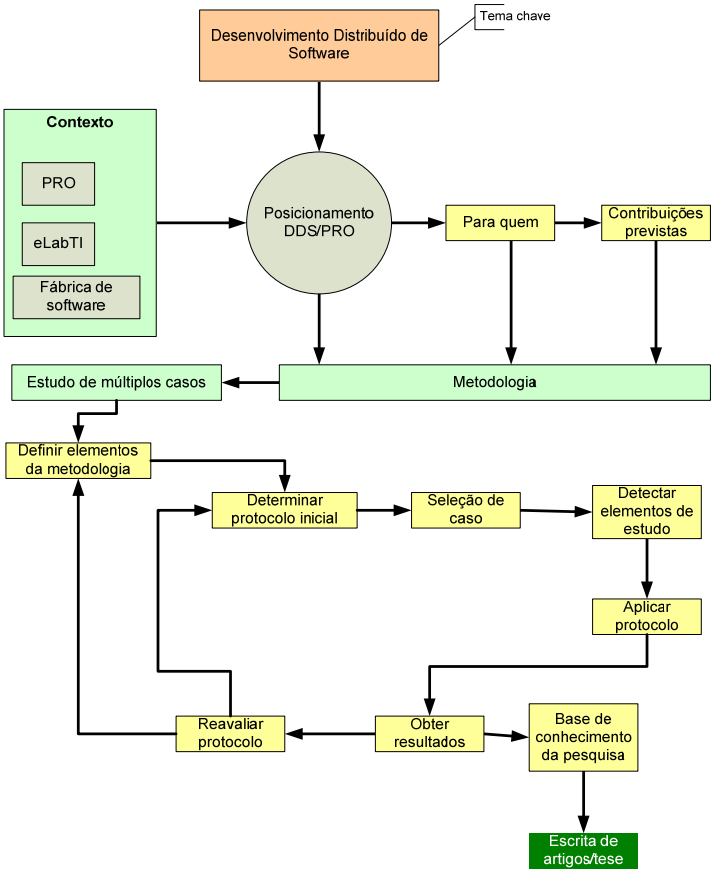

Figura 8 - Metodologia da tese

O círculo indicado como *Posicionamento DDS/PRO* na figura 8 indica o alinhamento do tema base, o desenvolvimento distribuído de software dentro do contexto da engenharia de produção.

A pesquisa não abordará temas técnicos como aspectos de codificação, mas somente os aspectos de distribuição e controle de atividades.

Segundo Umberto Eco (2003), a pesquisa deve identificar o público-alvo e as contribuições previstas. Nesse caso, o público desta pesquisa são as organizações que produzem software de forma distribuída ou têm a intenção de se iniciar em ambientes de DDS. Basicamente, este trabalho é direcionado a toda organização que produz software distribuidamente.

Este trabalho é instrumento que pode ser utilizado na prática de ensino/aprendizagem de desenvolvimento distribuído de software. O M3DS pode ser empregado sistematicamente por docentes a fim de ensinar os principais conceitos empregados no desenvolvimento distribuído de software.

As contribuições previstas neste trabalho englobam a compreensão do funcionamento dos ambientes de desenvolvimento distribuído de software, bem como uma abordagem sistemática sobre um ponto de vista escolhido neste ambiente.

A próxima seção descreve os elementos da metodologia que foram indicados na figura 8. Após a definição dos elementos, um protocolo inicial de pesquisa é elaborado e aplicado recursivamente com realimentação, exibida na figura 8 em formato de ciclo. O resultado do ciclo servirá para fomentar o corpo de conhecimento da pesquisa que, consequentemente, será o subsídio para elaboração de artigos e a escrita desta tese.

3.4 Elementos da metodologia adotada

Os elementos do estudo de caso descritos nesta seção são componentes do modelo de estudo de caso proposto por Yin (2005). Aqui, os componentes de pesquisa, questões do estudo, unidades de pesquisa e outros são discutidos detalhadamente.

A estratégia da pesquisa está ilustrada na figura 9. Baseado na literatura, foi construída a questão primordial, que conduziu a um conjunto finito de proposições. Tais proposições nortearam a elaboração das questões do protocolo de pesquisa que, posteriormente, conduziram o estudo em campo.

Figura 9 - Estratégia da pesquisa

A questão primordial desta tese é: **Como as organizações desenvolvem software de maneira distribuída?**

3.4.1 Proposições do estudo

As proposições do estudo são constituídas de expressões verbais ou simbólicas, que, ao final do projeto, devem ser classificadas como verdadeiras ou falsas. As seguintes proposições são aferidas neste trabalho:

P1: A heterogeneidade dos processos e/ou dos artefatos tornam o desenvolvimento mais complexo.

P2: O impacto cultural compromete o desenvolvimento.

P3: Artefatos não padronizados podem gerar impasses.

P4: O histórico da comunicação síncrona/assíncrona pode auxiliar novos sites.

P5: O distribuidor de tarefas é composto por vários níveis, de acordo com o grau de coordenação nele exercido e conforme a granularidade do repasse que deve ser entregue.

P6: O distribuidor de tarefas adota, de acordo com a necessidade do projeto, uma série de parâmetros para distribuir atividades, informal ou formal, que indica a produtividade da rede.

3.4.2 Questões do estudo

As questões de estudo concentram-se em identificar a discussão principal da tese. Neste trabalho, a metodologia de estudo de caso implica que a pergunta está atrelada a uma resposta de como e por quê. A questão da tese pode identificar também sob qual ponto de vista o trabalho foi seccionado e qual linha de pensamento lógico inicia o processo de solução de um problema.

A abordagem deste trabalho é a intersecção de dois pontos de vistas. O primeiro é a organização que entra em um ambiente de produção distribuída de software. O segundo é o projeto desenvolvido de maneira distribuída. Por isso, foi elaborado um modelo preliminar, apresentado no próximo capítulo. O modelo foi conduzido juntamente com o protocolo para validação.

O modelo proposto nesta tese abrange a interseção desses dois pontos, os principais ativos em um ambiente de DDS.

As seguintes questões são tratadas neste trabalho:

Q1: Como os diversos sites de desenvolvimento interagem durante a produção do software?

Q2: Como a organização mantém a produtividade em um ambiente de DDS?

Q3: Por que o uso de artefatos padronizados minimiza o impacto em ambientes de DDS?

Q4: Como as tarefas são distribuídas para a produção?

Q5: Como é o processo inicial de desenvolvimento distribuído nas organizações?

Q6: Por que e como as tarefas são rearranjadas de acordo com a demanda/necessidade?

Q7: Por que as organizações optaram por desenvolver em DDS e não em um modelo centralizado?

A figura 10, figura 11, figura 12, figura 13, figura 14 e a figura 15 representam o mapeamento das proposições em questões. Não há relação de apenas uma questão por proposição, mas há uma indução que leva as proposições a alimentar

várias questões. Do mesmo modo que uma questão pode ser alimentada por várias proposições.

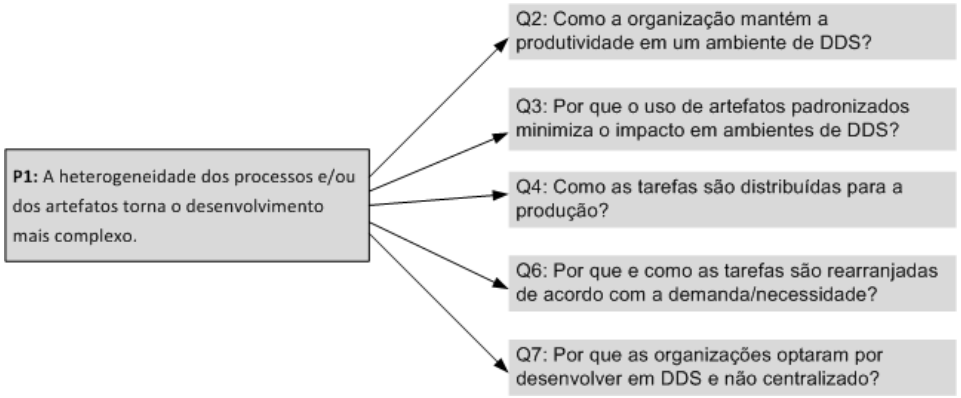

Figura 10 - Mapeamento da proposição P1 em questões

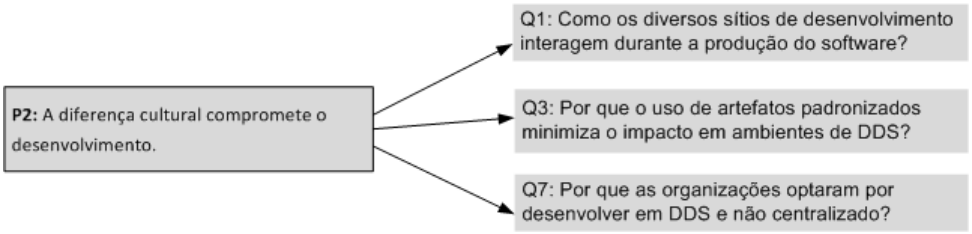

Figura 11- Mapeamento da proposição P2 em questões

Figura 12- Mapeamento da proposição P3 em questões

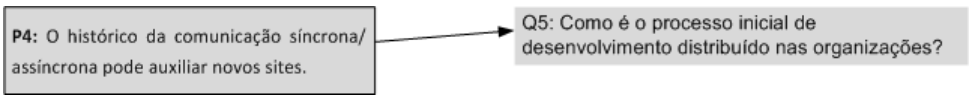

Figura 13- Mapeamento da proposição P4 em questões

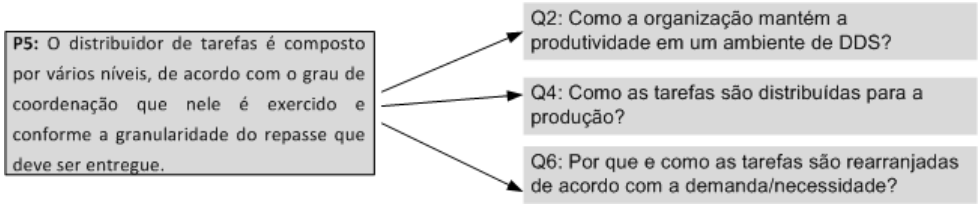

Figura 14- Mapeamento da proposição P5 em questões

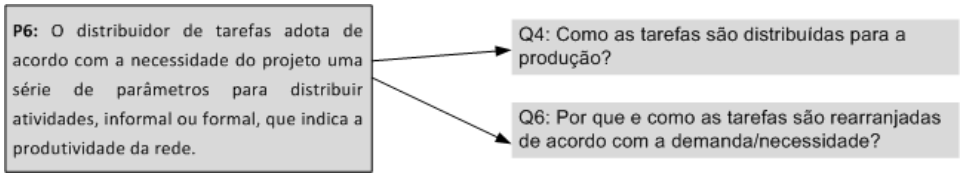

Figura 15 - Mapeamento da proposição P6 em questões

As questões foram mapeadas em blocos do protocolo de pesquisa, detalhado na seção intitulada *3.4.5 Protocolo de pesquisa*. Cada bloco foi norteado por uma ou mais questão e cada questão norteou um ou mais blocos.

A figura 16, figura 17, figura 18, figura 19, figura 20, figura 21 e a figura 22 representam o mapeamento efetuado entre as questões da tese e os blocos do protocolo de pesquisa. As bases teóricas (citações) que geraram os blocos são identificadas na seção *3.4.5 Protocolo de pesquisa*

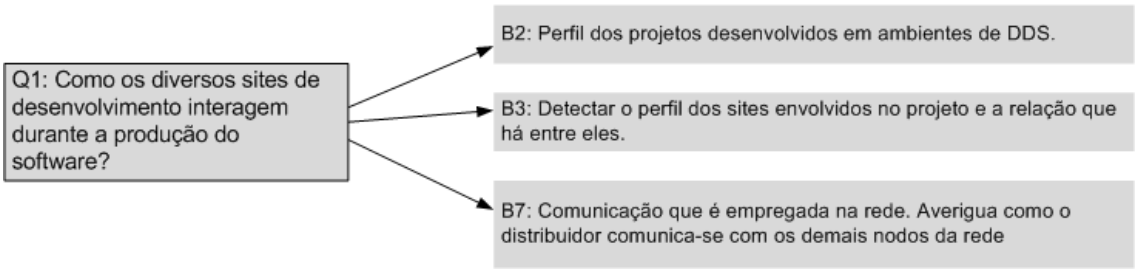

Figura 16 - Mapeamento da questão Q1 em blocos do protocolo

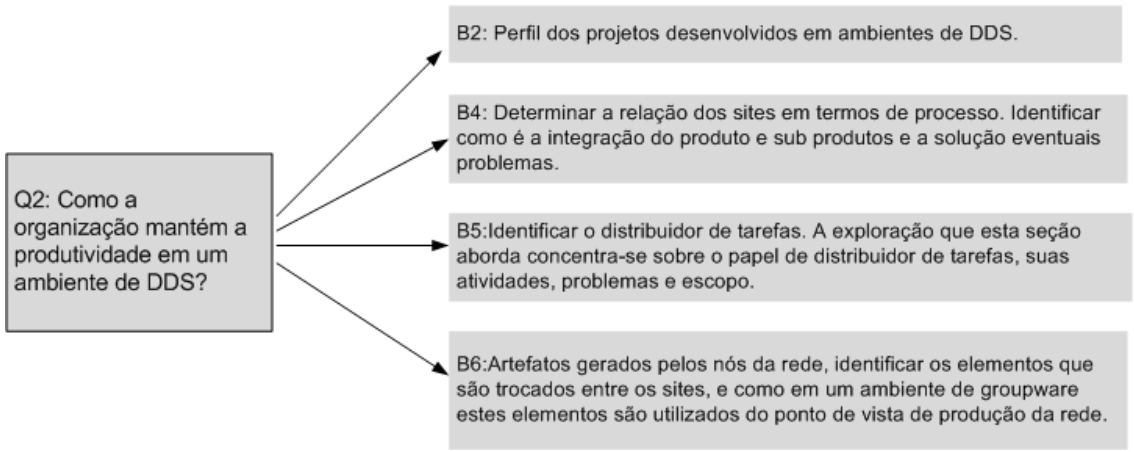

Figura 17 - Mapeamento da questão Q2 em blocos do protocolo

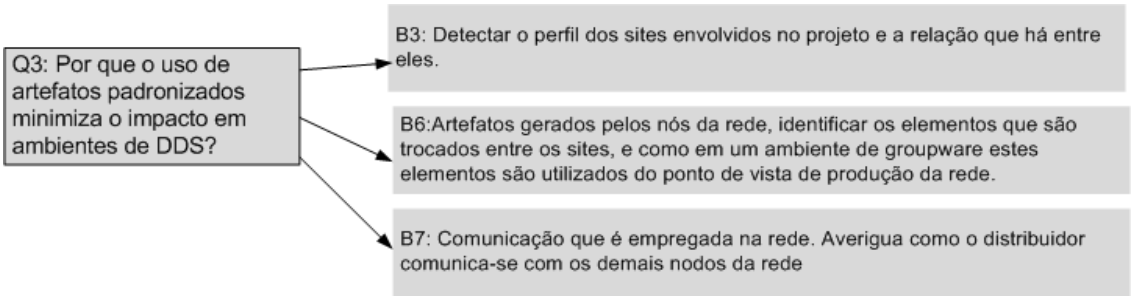

Figura 18 - Mapeamento da questão Q3 em blocos do protocolo

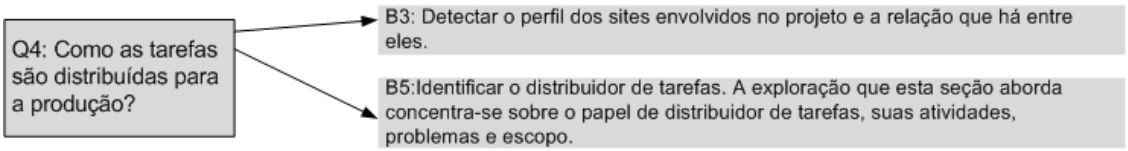

Figura 19 - Mapeamento da questão Q4 em blocos do protocolo

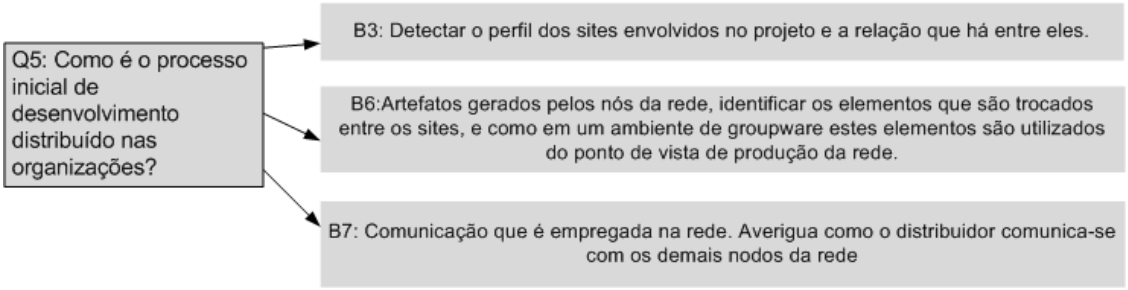

Figura 20 - Mapeamento da questão Q5 em blocos do protocolo

Figura 21 - Mapeamento da questão Q6 em blocos do protocolo

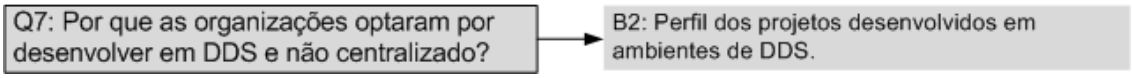

Figura 22 - Mapeamento da questão Q7 em blocos do protocolo

3.4.3 Unidades de análise

As unidades de análise são componentes que identificam quem são os elementos envolvidos na pesquisa. São considerados elementos de análise os objetos com os quais o pesquisador trabalha. Neste trabalho, as seguintes unidades de análise foram detectadas:

- Desenvolvedores que trocam informações direta ou indiretamente com outro nodo.
- Processo de interação (sistemático ou *ad hoc*) entre os nodos.
- Processos de integração entre segmentos do projeto desenvolvido distribuidamente.

3.4.4 Escolha dos casos

Este trabalho tem uma abordagem sobre múltiplos casos, portanto, vários casos serão selecionados. O perfil dos casos escolhidos engloba diversas organizações especializadas no desenvolvimento de software. Para esta pesquisa foram selecionados seis casos.

Estes casos foram selecionados, porque todos têm boa quantidade de sites de desenvolvimento (mais de 2). Destes 6 casos, 4 são de organizações multinacionais com experiência no desenvolvimento distribuído de software, um é referente a uma empresa nacional com desenvolvimento distribuído de software e outro caso é um projeto desenvolvido entre diversas instituições de ensino.

Todos estes casos foram escolhidos, por serem projetos reais e terem um tamanho considerável (acima de 20 colaboradores). Por esta razão, para o desenvolvimento de cada um destes projetos foi despendida uma grande quantidade de horas/homem de trabalho. O estudo de quatro organizações multinacionais com uma experiência de desenvolvimento global colaborou para a consolidação/construção de um modelo estável.

Embora as organizações multinacionais analisadas nesta pesquisa já tenham uma experiência em desenvolvimento distribuído de software, uma colaboração significativa foi o fato de terem projetos com perfis diferentes. Dessa maneira, foi possível analisar quão profundamente o projeto influenciou no DDS empregado pela organização.

O caso 1, desenvolvido distribuidamente entre várias instituições de ensino, colaborou neste projeto de duas maneiras. A primeira foi o acesso total a documentação e ao código fonte. O acesso a essas informações permite análise mais minuciosa da relação do que foi documentado e implementado. A segunda foi a

quantidade de impasses gerada pelo projeto. Esses impasses serviram como questionamento ao modelo, complementando-o e tornando-o mais genérico e consolidado

No projeto indicado como caso 1, além da entrevista, foi possível acesso total ao código fonte, atas de reuniões e documentações do software. Dessa forma, foi possível analisar o que havia sido discutido presencialmente, o que foi desenvolvido de maneira distribuída e como foi desenvolvido e integrado.

O caso 2 foi tipicamente de uma organização de desenvolvimento distribuído de software, e não desenvolvimento global de software. Esse caso colaborou com o modelo a fim de validá-lo com relação a processos de desenvolvimento *onshore*.

Os casos 3 a 6 foram de desenvolvimento global de software. São de organizações com processo de desenvolvimento de software sólido e experiência e know-how no desenvolvimento distribuído de software.

A tabela 3 relaciona as características gerais dos casos. Nesta, são identificadas as certificações, a dispersão física de cada site, a quantidade de sites necessários para a construção do projeto investigado e o modelo de negócio de DDS na qual cada caso emprega.

Tabela 3 - Características gerais dos casos

CASO	Certificações	Dispersão física	Quantidade de Sites	Modelo de negócio
Caso 1	---	Nacional	15	Cosourcing
Caso 2	CMMI 5	Global	5	Offshore Outsourcing / Development centers
Caso 3	CMMI 2	Nacional	3	Onshore insourcing
Caso 4	CMMI 5	Global	4	Offshore Outsourcing / Development centers
Caso 5	CMMI 5	Global	30	Offshore insourcing / outsourcing
Caso 6	CMMI 3	Global	3	Offshore insourcing

3.4.5 Protocolo de pesquisa

O protocolo de pesquisa identifica como o pesquisador deve portar-se perante as unidades de análise. Mais do que isso, identifica de modo imparcial como são coletadas as informações. A figura 23 ilustra os elementos de estudo da tese. Sobre eles é aplicado o protocolo de pesquisa.

O elemento central é o papel do distribuidor de tarefas, independentemente de ser um único elemento central ou cada site possuir um elemento com o papel. Pode ser representado pelo gerente de projeto, ou pelo arquiteto de software.

O protocolo de pesquisa foi aplicado por meio de entrevistas, análise de documentos e código fonte quando disponíveis. As organizações privadas não concedem abertamente o acesso a documentação e ao código fonte do software. Por isso, foi efetuada entrevista presencial com um papel principal (gerente de projeto e/ou desenvolvedor) e, entrevistas com demais desenvolvedores de um mesmo projeto na empresa.

A entrevista principal foi presencial em todos os casos, entretanto as entrevistas com desenvolvedores foram feitas, na maioria das vezes, a distância, mediada por sistemas eletrônicos de mensagens síncronas.

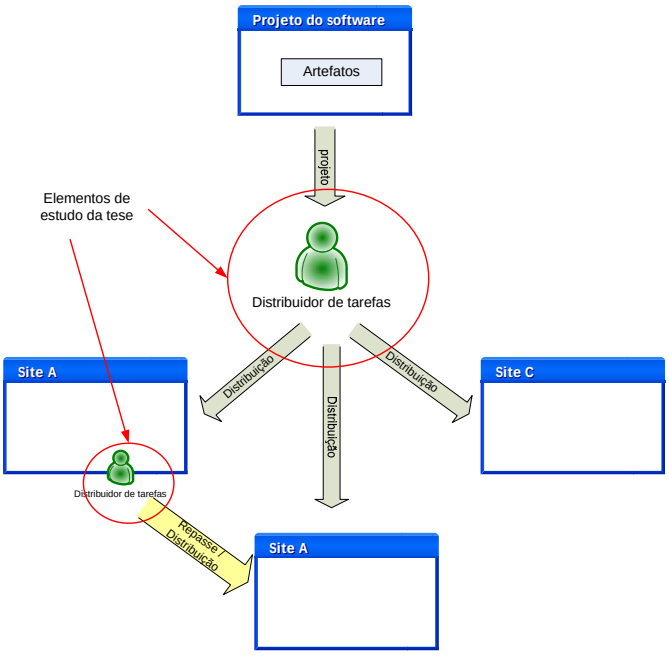

Figura 23 - Elementos de estudo da tese

A aplicação dos dois tipos de entrevistas permitiu validar a informação adquirida e a complementar as informações necessárias detectadas posteriormente.

Utilizando desses meios, entrevista e análise de documentos, foi consolidado o estudo de múltiplos casos.

O protocolo de pesquisa apresenta uma série de etapas, seccionadas em blocos, conforme indica a figura 24.

A primeira seção visa a identificar o perfil da organização da rede, ou seja, indica como a rede é configurada em termos de tamanho, dispersão, coordenação etc.

Na próxima etapa é identificado como o distribuidor de tarefas efetua o gerenciamento da produção considerando os valores identificados na organização da rede. Na etapa seguinte verifica-se como é a produção a partir da tomada de decisão do elemento que distribui tarefas.

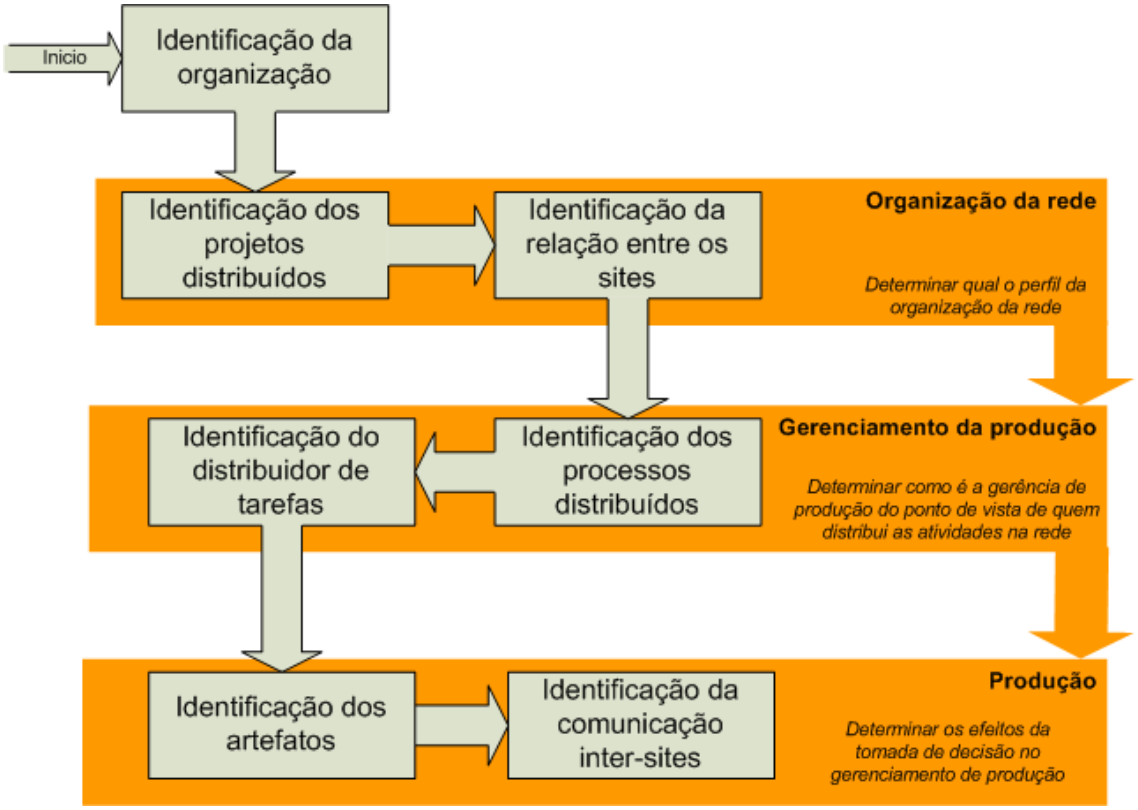

Figura 24 - Etapas do protocolo de pesquisa

As próximas seções descrevem os blocos, o que é investigado em cada bloco e quais as bases teóricas empregadas.

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Base teórica:	Nenhuma em especial
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; Organograma; e Certificações da empresa;

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e

	- Verificar o mecanismo de coordenação em diversos níveis do projeto

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e sub produtos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa;

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de	Identificar artefatos que foram desenvolvidos em rede (plano de

análise:	desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

3.5 Considerações finais

Este capítulo apresentou a metodologia que a tese seguirá como guia. A metodologia visa a identificar variáveis. No primeiro momento – no caso 1 – é aplicada principalmente com o objetivo de testar sua veracidade e realimentar sua fundamentação.

O protocolo de pesquisa serviu como mecanismo de coordenação para a pesquisa investigativa. Com esse mecanismo, foi possível entrevistar e extrair informações de diversas fontes em uma mesma organização. Também foi possível aferir diversos pontos de vistas divergentes em um mesmo caso e determinar a

veracidade da informação. Desse modo foi possível verificar a veracidade do modelo preliminar e determinar elementos do desenvolvimento distribuído de software não detectados somente com a revisão bibliográfica.

CAPÍTULO 4 - M3DS: O MODELO PRELIMINAR

O primeiro contato deste projeto com desenvolvimento distribuído de software foi em 2004 (L'ERARIO et al, 2004), juntamente com a etapa, foi realizada uma primeira revisão bibliográfica. Conforme visto no capítulo 2, foram detectados numerosos estudos de casos de diversos autores.

O primeiro estudo levou a buscar a possibilidade de compreender, de maneira homogênea, os diversos casos com perfis diferentes apresentados na literatura. O principal objetivo naquele momento foi determinar uma maneira de comparar e compreender os diversos estudos de casos. Na literatura, há estudos de casos de desenvolvimento distribuído de software aplicados em grandes corporações e em pequenas empresas, a softwares proprietários e a abertos. Por essa razão, foi elaborado um primeiro modelo de dinâmica de desenvolvimento distribuído de software.

O primeiro modelo compõe uma parte do protocolo de pesquisa, isto é, além do protocolo apresentado na seção 3.4, o modelo preliminar foi conduzido ao caso para que pudesse ser validado. Este capítulo apresenta o modelo preliminar, ou seja, o modelo original elaborado a partir da revisão bibliográfica e da pesquisa-ação (L'ERARIO et al., 2004), realizada pelo Departamento de Engenharia de Produção da POLI/USP.

4.1 O modelo preliminar

Um modelo, segundo Sayão (2001), é uma criação cultural, um artefato mental, destinado a representar uma realidade, ou alguns dos seus aspectos, a fim de torná-los descritíveis qualitativa e quantitativamente e, algumas vezes, observáveis. Os modelos possibilitam descrever objetos e esgotar as possibilidades de sua observação. O aparecimento de modelos é necessário para extrapolar a limitação de percepção humana com relação ao mundo, que apresenta como um desafio permanente sua descrição.

Modelos representam uma analogia, sempre que possível, mas nem sempre desejável, com o objeto real. Assim, é a representação de uma mesma função em diversos materiais e princípios. Pode ser elaborado por meio de formalismos matemáticos, fenomenológicos ou conceituais. Segundo Sayão (2001), é mais simplificada, com vida provisória e permite testar hipóteses, tirar conclusões, generalizar e especializar por meio de processos de indução.

Segundo Herbert Stachowiak (1972), um modelo tem três características básicas. A primeira característica básica de um modelo é o mapeamento, isto é, um modelo deve retratar alguma coisa. A segunda característica é a redução que, nesse caso, indica que um modelo não deve mapear tudo, mas somente as características consideradas relevantes. A última característica é que um modelo deve ser pragmático, ou seja, deve ser criado sobre alguma coisa, por alguém, para alguém e deve ser utilizado em lugar do objeto de estudo original, por alguma razão.

L'Erario e Pessôa (2007) afirmam que a dinâmica de um ambiente de DDS revela o funcionamento da rede, da sua concepção inicial até seu encerramento. Porém, abordar individualmente os sites na rede pode não revelar a influência do projeto sobre sua existência na rede. Portanto, a dinâmica da rede, representada na figura 25, aborda conjuntamente dois componentes fundamentais em um ambiente de DDS.

O primeiro componente indica a dinâmica de uma unidade de produção de software, o **site**. O segundo elemento é atrelado ao **projeto,** desenvolvido em uma rede de produção de software. Sobre essa dinâmica, há uma intersecção que ocorre quando o site precisa desenvolver um subproduto de rede e há uma instância do projeto para vários sites.

Na figura 25 o elemento indicado como *Novo Projeto* indica a concepção de um novo produto de software por alguma organização, em rede ou não.

A seta 1 da figura 25 indica que o plano gerencial do projeto em termos de componentização e divisão de tarefas foi realizado. Após a seta, todos artefatos – ou parte deles –estarão prontos para serem disponibilizados e distribuídos para que a rede e os sites possam desenvolvê-los. O elemento *pronto*, após a seta 1, indica que

um papel de distribuidor de tarefas entra em ação para gerenciar a distribuição das tarefas do projeto na rede.

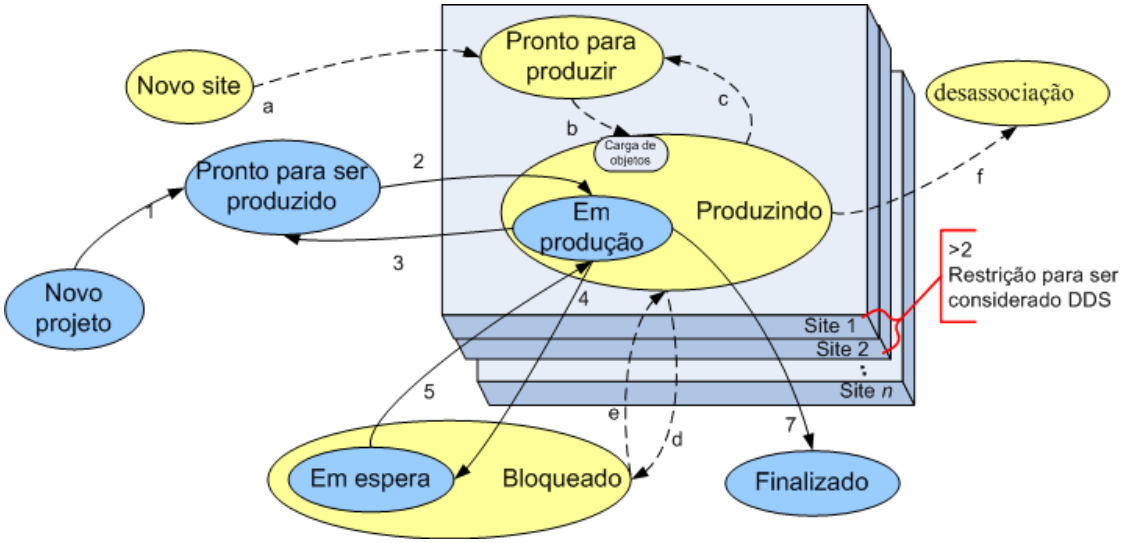

Figura 25 - M3DS: modelo preliminar

Na figura 25, o elemento *Novo site* indica que um novo nó da rede está pronto para associar-se a uma rede de produção de software. A elipse *pronto*, após a seta *a*, indica que o site já está na rede, ou seja, de alguma forma um acordo foi feito entre os dois (site e rede). Um exemplo é o conceito de matriz e filial.

O estado de *pronto* após a seta *a* indica que a organização submeteu-se a alguma análise, resultando essa em uma agregação à rede. No estado de *pronto*, após a seta *a*, o site está aguardando tarefas.

A seta 2 indica que o subproduto foi alocado em produção. A ação ocorre paralelamente à ação indicada pela seta *b*, que muda o estado do site de pronto para *produzindo*.

O estado *Produzindo* efetua algumas tarefas de verificação. Por exemplo, aceitar ou não o trabalho. Se aceito, a tarefa entra em produção, caso contrário, a tarefa retorna e o papel do divisor de tarefas será realocar o trabalho (seta 3). O site eventualmente fica ocioso, aguardando nova tarefa (seta c).

As setas 4 e *d* podem ocorrer quando há dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subprotudo fica bloqueado, pois, não pode ser desenvolvido porque sofre dependência funcional. O site fica ocioso, aguardando uma nova tomada de decisão.

As seta 5 e *e* representam, respectivamente, a resolução de dependência funcional e o retorno do nó na produção do artefato.A seta 7 indica quando o subproduto é finalizado e entregue à rede. Nesse caso, o estado finalizado indica o fim da subtarefa para um determinado site.

A ação indicada pela letra *f* indica a desassociação do site à rede que é removido da mesma e deixa de fazer parte do ambiente de DDS.

Nessa dinâmica, representada pela figura 25 há uma regra de restrição imposta sobre a quantidade de sites disponíveis, pois, para que seja considerado um ambiente de desenvolvimento distribuído de software, é necessário que a quantidade de sites seja maior ou igual a 2.

O capítulo 6 apresenta a versão final do modelo preliminar, denominado M3DS. Houve algumas modificações no modelo por causa da aplicação dos estudos de casos apresentados no capítulo 5. O modelo e os detalhes da versão final serão explicados no capítulo 6.

4.2 Propriedades dos ambientes de DDS

O desenvolvimento distribuído de software pode ocorrer sob várias circunstâncias e de várias maneiras diferentes. Há várias propriedades que diferenciam os ambientes de DDS. Cada propriedade determina o grau de complexidade de trabalho e revela consigo quão difícil é de produzir software em equipes fisicamente dispersas. Além disso, cada propriedade agrega consigo uma série de conceitos aplicados do nível estratégico ao operacional da rede.

Esta seção explora as diversas dimensões de propriedades identificadas em ambiente de desenvolvimento distribuído de software. Revela sobre cada aspecto dimensional qual contexto encontrado na literatura.

Essas propriedades estão atreladas à dinâmica de funcionamento dos ambientes de DDS. Várias delas se tornam importantes, após determinados processos indicados na figura 25. A tabela 4 mostra a relação entre a dinâmica, baseada na figura 25, e as propriedades, abordadas neste capítulo. A primeira coluna da tabela indica qual elemento da figura 25, enquanto a segunda indica qual propriedade é instanciada.

Tabela 4 – A relação da dinâmica com as propriedades dos ambientes de DDS

Posição na figura 1	Propriedade eminente
Seta *a*	Mecanismo de coordenação
Seta *b, c*	Grau de interação
Novo site	Distância física e modelo de agrupamento
Seta 2, 3, b, c	Granularidade do repasse

A figura 26 ilustra as diversas propriedades encontradas em um ambiente de DDS, explanadas nas seções seguintes. Ela contém linhas escalares que compõem as diversas propriedades dos ambientes de DDS.

A relação entre elas é expressa por uma condição mínima, definida na figura como quantidade de sites maior que 1, distância, granularidade de repasse, difusão do processo e grau de interação maiores do que 0 . Esta condição mínima indica o quanto de cada propriedade deve ser configurado para que o ambiente de desenvolvimento seja considerado distribuído.

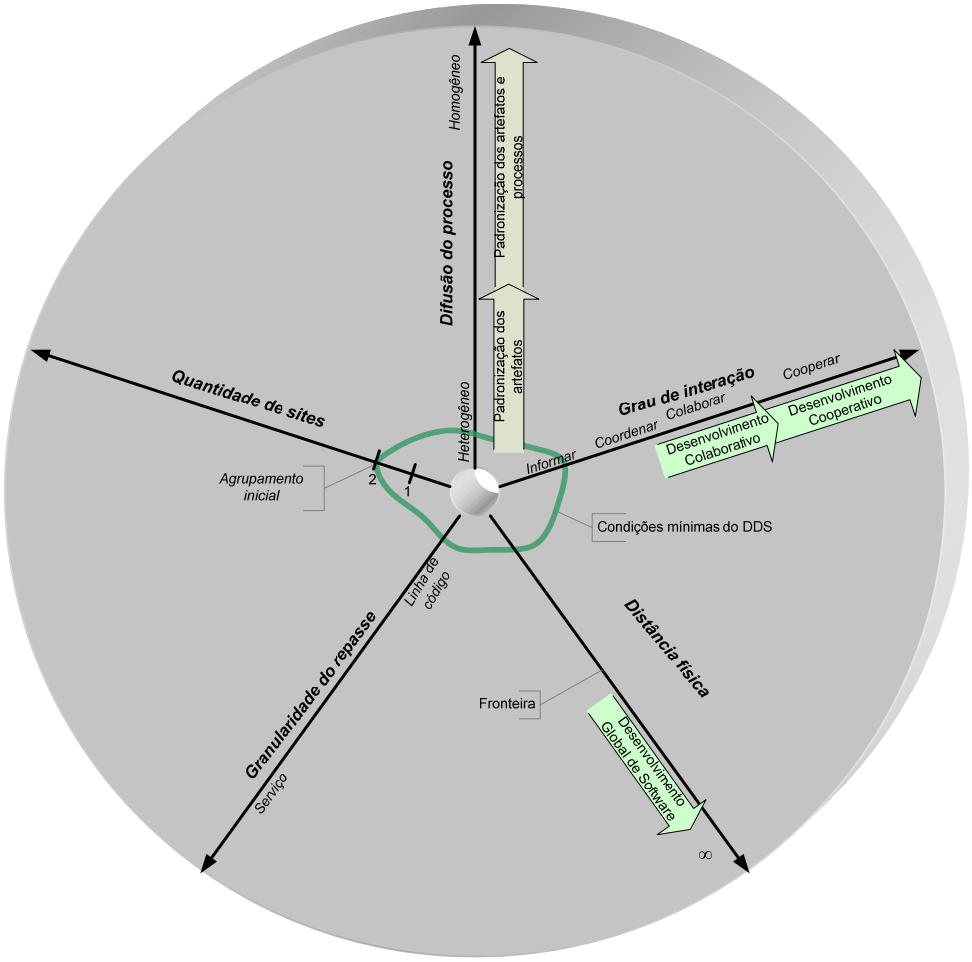

Figura 26 - Propriedades dos ambientes de DDS

Além disso, cada plano, composto por essas propriedades é orientado por um mecanismo de coordenação. A figura 27 ilustra a relação dos mecanismos de coordenação, indicado por um cilindro, propostos por Mintzberg (2003).

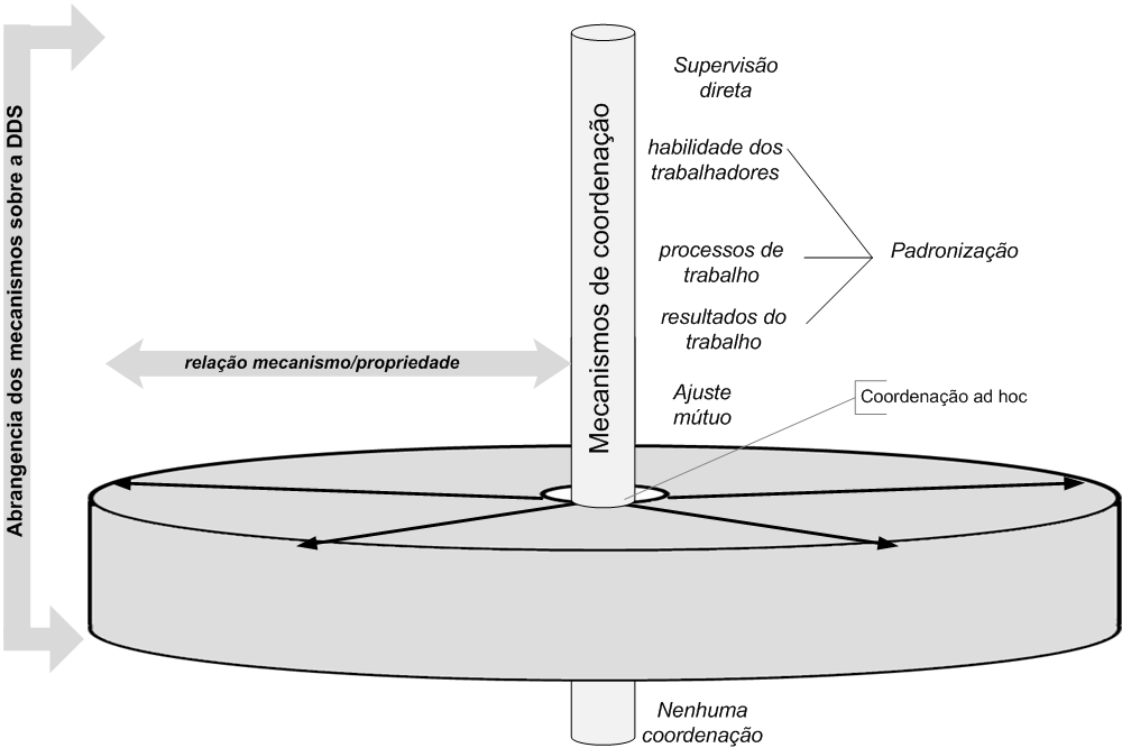

Figura 27 - Relação entre os mecanismos de coordenação e as propriedades dos ambientes de DDS

Os mecanismos de coordenação utilizados pelos ambientes de DDS englobam ajuste mútuo, padronização e supervisão direta e exercem influência contínua sobre as propriedades dos ambientes de desenvolvimento distribuído de software. Isso define resumidamente como o processo de desenvolvimento distribuído ocorre. Por exemplo, se o mecanismo for "ajuste mútuo," ambos os sites têm a mesmo grau hierárquico e precisam definir e ajustar mutuamente os artefatos trocados entre eles.

O mecanismo de coordenação "ajuste mútuo" define que não há nenhuma padronização e os grupos precisam de ajustes contínuos entre artefatos trocados. A padronização como mecanismo de coordenação está subdividida em três categorias: padronização de resultados, processos e habilidades.

Na primeira, enquadra-se a padronização dos artefatos; na segunda, o processo de software como um todo; e a terceira, a padronização com relação as habilidades dos desenvolvedores existentes nos grupos.

A supervisão direta indica total responsabilidade de um grupo, que deve definir como seus grupos subordinados trabalham.

As retas indicadas pela figura 26 são grandezas escalares. A reta definida como difusão do processo indica quão homogêneo é o processo na rede. Nessa propriedade, a homogeneidade ou heterogeneidade do processo de todos os nodos do site é avaliada. A reta definida como grau de interação indica a intensidade de interação dos sites.

Segundo a proposta de Borghoff e Schlichter (2000), quando a interação é apenas informativa, o grau de interação é mínimo, mas quando o objetivo é cooperação, o grau de informação é maior. Para serem considerados distribuídos, os nodos devem ficar a uma distância considerável, um do outro (indicado pela reta definida como distância física da figura 4). A última reta, definida como a quantidade de sites, neste caso deve ser maior do que 1 (um).

As seções a seguir, explicam com mais detalhes o que cada propriedade engloba efetivamente dentro do ambiente de desenvolvimento distribuído de software.

4.2.1 Interação

A interação, de acordo com Barnes (2008), são estímulos enviados entre instâncias com o objetivo de se realizar uma determinada tarefa. A figura 28 ilustra um modelo de interação entre os diversos sites de uma mesma rede. A elipse indicada como abstração de um site de rede, mostra em uma visão macro, o sites sob o aspecto de dinâmica indicadas pela figura 25 e figura 26. As instâncias dessa abstração constituem os sites.

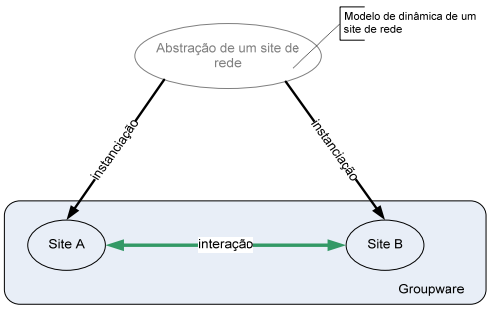

Figura 28 - A interação entre sites, adaptado de Barnes (2008)

O grau de interação revela quanto a informação é trocado entre os nodos e, nesse caso, há uma diferença de intensidade quando o software é desenvolvido de maneira colaborativa ou cooperativa.

Borghoff e Schlichter (2000) utilizam classificação que mostra o grau de comunicação em grupos representada pela figura 30, na qual se pode perceber que, à medida que aumenta o grau de comunicação entre os grupos, aumenta também, a necessidade do uso de suporte de computadores para intermediá-los. De acordo com a figura 30, informar é o nível mais baixo de comunicação.

Na coordenação, há a exigência de que o envio e o recebimento de dados influenciem diretamente suas atividades e informações, por exemplo, utilizando algum recurso compartilhado. Um nível acima, encontra-se a colaboração, em que o envio e o recebimento de dados estão direcionados para o mesmo objetivo do trabalho de um grupo.

As ferramentas computacionais de apoio são denominadas ferramentas de *groupware*. Qualquer ferramenta de software que suporta times com membros que trabalham colaborativamente, interconectando computadores pessoais, segundo Defranco-Tommarello e Deek(2002), é denominada *groupware* ou *Virtual Team System* (VTS).

Esses sistemas podem atender com sucesso uma equipe em um *workspace* compartilhado sob um objetivo comum. Em uma visão macro, Defranco-Tommarello e Deek (2002), propuseram em seu trabalho uma visão em árvore, com a qual explicam como os problemas colaborativos podem ser resolvidos. A figura 29 ilustra a árvore.

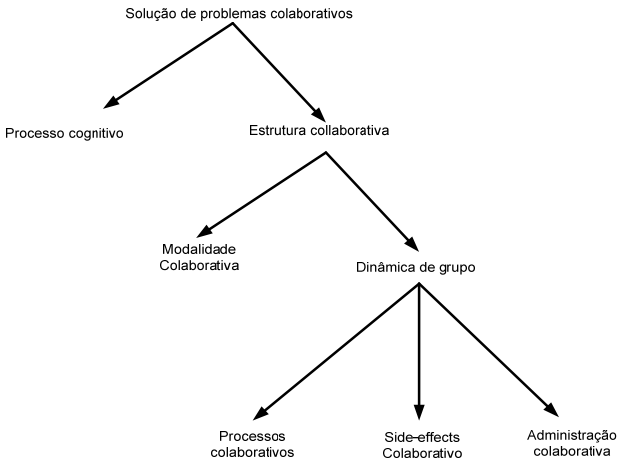

Figura 29 - Árvore proposta por Defranco-Tommarello e Deek (2002)

Mühlbacher e Neumann (1996) afirmam que os sistemas *groupware* são compostos por agentes e que trabalhar com eles em um ambiente distribuído possibilita encontrar a solução muito mais rapidamente do que trabalhar de modo isolado. Os autores citam um caso em que foi possível desenvolver um sistema operacional em um ambiente de agentes distribuídos, como foi o caso do Sistema Operacional Linux.

A figura 30 indica a diferença entre cooperação e colaboração. Mais ainda, define o posicionamento do desenvolvimento distribuído de software, o qual pode ocorrer por colaboração ou por cooperação.

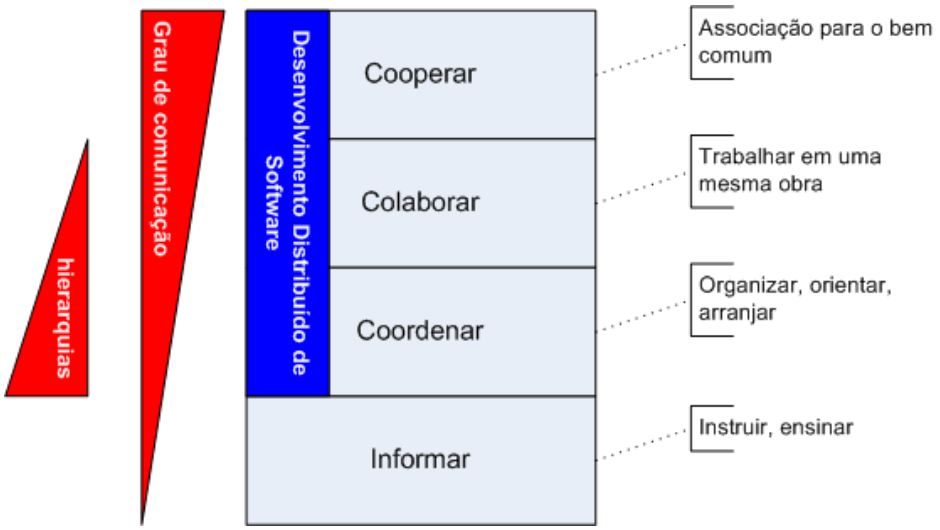

Figura 30 - Diferença entre colaboração e cooperação

No nível mais alto, há a cooperação, limitada por várias características, como metas em comum, plano compartilhado, processamento de dados para suporte e cooperação coordenada. No nível de cooperação, metas do grupo têm prioridade sobre metas individuais. Em geral, as decisões são baseadas no consenso do grupo.

A cooperação é o desenvolvimento para o bem comum. Por isso, tem menor tendência a constituir hierarquias formalizadas. Ao contrário, a coordenação envolve diretamente a formação hierárquica. Borghoff e Schlichter (2000), definem posicionamento do desenvolvimento distribuído de software, que pode ocorrer por colaboração ou por cooperação. Segundo Jørstad; Dustdar e Thanh (2005), os serviços colaborativos podem ser classificados como:

- Compartilhamento de recurso ou conhecimento
- Comunicação e iteração pessoal
- Salas virtuais
- Gerenciamento de organizações

O'Day, Bobrow e Shirley (1996) afirmam que, enquanto na cooperação as relações de poder e os papéis dos participantes no trabalho cooperativo não são questionados, a colaboração envolve negociação cuidadosa, tomada conjunta de decisões, comunicação efetiva e aprendizagem mútua, num empreendimento que se foca na promoção do diálogo profissional.

Pode-se ainda caracterizar a cooperação em síncrona e assíncrona. O primeiro caso envolve o trabalho simultâneo dos participantes; o segundo, o trabalho não simultâneo (BORGES; CAVALCANTI; MACHADO, 1995). Como exemplo, há videoconferência e o uso de e-mail, respectivamente.

Navarro, Prinz e Dodden (1992) definem o grau de sincronismo comparando com a distância física. Em sua classificação, quando a iteração ocorre ao mesmo tempo, é preciso uma ferramenta cuja interface seja distribuída e síncrona. Quando o tempo de comunicação entre sites for diferente, a interface requerida é distribuída e assíncrona.

Há vários conflitos resultantes, Campbell e Van de Walle (2003) defendem que a comunicação assíncrona leva o desenvolvedor próximo ao cliente, mas aumenta a complexidade de desenvolvimento, principalmente na atividade de requisitos.

A tabela 5 é uma representação, proposta por Navarro, Prinz e Rodden. O grau de sincronismo relacionado com o posicionamento físico dos sites indica o comportamento das interfaces de comunicação entre os nodos da rede.

Tabela 5 - Grau de sincronismo: posição física versus tempo (proposto por Navarro, Prinz e Rodden)

	Mesmo tempo	Tempo diferente
Mesmo lugar	Interface face a face	Interface assíncrona
Lugar diferente	Interface distribuída síncrona	Interface distribuída assíncrona

A última linha da tabela 5, "lugar diferente" é uma característica de um ambiente distribuído. Nesse caso há duas interfaces. A primeira é síncrona, ou seja, os

sites efetuam um mesmo trabalho ao mesmo tempo. A segunda coluna está relacionada a uma interface assíncrona, na qual vários sites interagem, porém em tempos diferentes.

4.2.2 Distância geográfica

A distância geográfica está relacionada com o espaço físico entre os sites da rede. Segundo Lopes e Audy (2003), o Desenvolvimento Distribuído de Software, quando atinge proporções globais é chamado de GSD (Desenvolvimento Global de Software). Pode-se considerá-lo como um tipo particular de DDS.

Shami et al. (2004) definem

> "O desenvolvimento de software está crescentemente global, com o desenvolvimento que acontece em várias regiões geograficamente diferentes."

Afirmam, também, a importância das redes sociais em ambientes de DDS, as quais podem reduzir o impacto cultural sobre os sites da rede. Além disso, podem, de certa forma, induzir a uma interação mais agradável entre os elementos da rede.

Conforme a distância física das pessoas aumenta, segundo Siqueira e Silva (2004), torna-se cada vez mais difícil realizar reuniões presenciais com os membros do projeto. Por isso, o desenvolvimento distribuído, reflete uma realidade cada vez mais comum nos projetos de software, em quem *stakeholders* não residem próximos.

4.2.3 Configuração do ambiente tecnológico

A configuração do ambiente tecnológico está relacionada com a configuração dos processos e dos recursos computacionais que dão suporte para o desenvolvimento distribuído.

Está acoplada fortemente a um ambiente centrado em processo, que deve fornecer instruções sobre como o desenvolvedor e/ou demais papéis no desenvolvimento do projeto devem proceder para configurar o ambiente de trabalho.

Como, nesse ambiente, todos utilizam o mesmo processo ou parte dele, é necessária uma padronização mínima, capaz de garantir a troca de artefatos entres sites da rede.

A arquitetura de software, empregada como solução final do produto pode auxiliar o desenvolvimento em ambientes distribuídos, pois, leva os desenvolvedores a utilizarem processos e tecnologias padronizadas.

Além disso, a forma de reuso de componentes, além do tempo, pode reduzir o custo final do produto. Henrich e Morgenroth (2003) propuseram um sistema de busca em base de repositório capaz de identificar possíveis componentes reusáveis em um ambiente DDS.

Henrich e Morgenroth (2003), por exemplo, baseram todo o projeto de busca em protocolos XML. A busca é fundamentada no progresso do processo sobre os artefatos, buscando soluções similares aos artefatos que serão postos em linha de produção. A figura 31 é baseada na pesquisa citada, e ilustra como o processo funciona.

Em seu projeto, Henrich e Morgenroth desenvolveram métodos para busca de componentes sobre um contexto, por exemplo, componentes similares em uma determinada atividade, projeto, ou mesmo componentes utilizados por desenvolvedores em uma determinada fase do projeto.

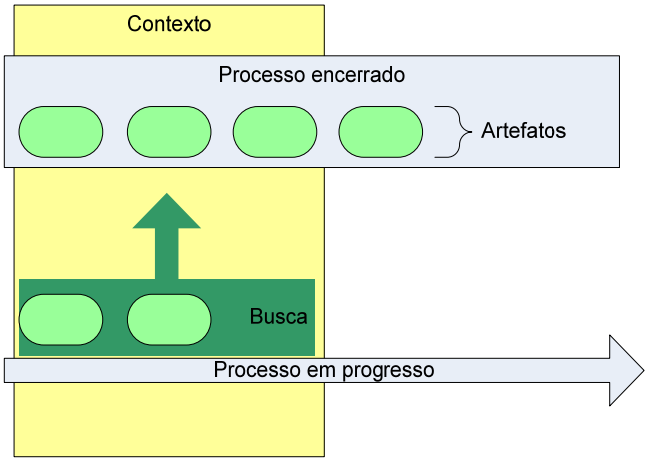

Figura 31 - Sistema de busca de componentes (Baseado em Henrich e Morgenroth 2003)

Em uma visão sobre projeto como um todo, Lee, Hickey e Zhang (2000), apresentam uma abordagem multimetodológica. Essa abordagem implica a existência de quatro etapas.

Na primeira etapa, os autores, indicam a construção de um *framework* conceitual. Nele, deve ser indicado o esquema de trabalho geral da rede. A segunda etapa é a elaboração da arquitetura da solução como um todo. Essa arquitetura

norteará os desenvolvedores em relação tanto ao processo, quanto e principalmente aos recursos tecnológicos empregados. O refinamento dessa primeira arquitetura, o software deve ser projetado e efetivamente desenvolvido na terceira etapa. A última etapa é uma realimentação da primeira, observando o sistema e reavaliando.

Wu e Sahraoui (2004) explicam que o uso de realidade virtual, com recursos multimídia, aliados à inteligência artificial aumenta a produtividade em um CVE (ambiente virtual colaborativo).

A granularidade baixa de colaboração, definida por Hupfer et al. (2004), permite adequação de ferramenta IDE a ambiente de colaboração. Os mesmos autores citam a importância de integrar ferramentas de comunicação síncronas como chat, vídeo, e outras a IDEs de modificação do código em tempo real e simultâneo.

Em abordagem um pouco diferente, baseada na solução, Jørstad; Dustdar e Thanh (2005), afirmam que a arquitetura orientada a serviços (SOA) é uma forma de arquitetura de sistemas distribuídos que, tipicamente, caracterizada pelas seguintes propriedades:

- Visão lógica: o serviço é abstraído sob programa e banco de dados;
- Orientação a mensagem: uma arquitetura orientada a serviços é definida como troca de mensagem entre agentes;
- Orientação a descrição: os metadados são definidos e implementados;
- Granularidade: indica em qual nível as informações ou funções devem ser tratadas. Por exemplo, abordar um *webservice* ou linhas de código;
- Orientação de rede: qual o sentido das informações;
- Plataforma neutra: definir o quão interoperável é o ambiente.

Dossick e Kaiser (1999) afirmam a necessidade, em ambientes distribuídos, do desenvolvimento de metadados. Para tanto, em sua pesquisa, utilizam aplicações de CSCW para definir informações em um ambiente de desenvolvimento distribuído de software.

Grinter (1995) defende a ideia de que o processo de coordenação inicia-se em um sistema de gerenciamento de configuração em um DDS. Por isso, deve-se configurar regras de acesso, níveis de permissões etc.

Em ambientes de CSCW, a formalização de aplicações segundo Chang, Zhang e Thang (2001), pode levar todo o conjunto a homogeneização e aumento da performance da produtividade da rede. Os autores informam, ainda, que a área de pesquisa sobre CSCW está migrando de estudos centrados em problemas oriundos dos sistemas centralizados – como concorrência em simples *thread*, acesso linear à aplicação – para sistemas colaborativos, com concorrência efetivamente paralela, sobre demanda e, além de tudo, ubíqua.

4.2.4 Difusão de processo

A difusão de processo está relacionada ao grau de equivalência dos processos utilizados pelos sites de um determinado. A questão em relevância é averiguar quanto em comum de processos há entre os sites da rede.

Para efetuar essa difusão, Mühlbacher e Neumann (1996) determinaram um conjunto de objetos de negócio comuns a ser compreendido pelos sites. Cada objeto de negócio é composto por um ou mais *workflow* e contém regras de acesso. Além disso, criaram o conceito de objetos móveis, constituído pela informação que trafega entre os participantes. Para tanto, utilizam repositório centralizado com controle de versão.

Em análise mais formal, do ponto de vista matemático, Wu, Yu e Wu (2004), utilizam modelos formais como redes de Petri para especificação de atividades e objetos em um DDS. Especificam, ainda, atividades colaborativas, modelos de inteligência,ciclos de vidas de objetos em ambientes de DDS utilizando redes de Petri.

Na mesma linha de raciocínio, Jalote e Jain (2004), utilizam grafos para especificação formal de processos de rede. Para tanto, é necessário determinar conceitos básicos como restrições operacionais, restrição de conhecimento e restrição de recurso. Em seu projeto, os pesquisadores realizam, baseados nas informações dos grafos, o escalonamento de tarefas.

4.2.5 Modelo de agrupamento

O agrupamento é o conceito, de acordo com o qual, as unidades de desenvolvimento são analisadas sob uma macrovisão e constituem uma organização única. O agrupamento não é apenas um recurso para a criação de organogramas; ao contrário, é um meio para coordenar o trabalho na organização. Segundo Mintzberg (2003), pode ter os seguintes efeitos importantes:

- Estabelecer um sistema de supervisão comum entre as posições e as unidades;

- Requerer posições e unidades para compartilhar recursos comuns;

- Criar medidas de desempenho comuns;

- Encorajar o ajuste mútuo.

Há vários fatores que podem determinar o agrupamento de times. Mintzberg os denomina como bases para o agrupamento. As bases nas quais a organização agrupa posições e as unidades são consideradas agrupamento por conhecimento e habilidade, por processo de trabalho e função, por tempo, por output, por cliente e por local.

Agrupamento por conhecimento e habilidade é constituído de acordo com a habilidade e o conhecimento das pessoas. Por exemplo, basear-se em especialização por área tecnológica, como computação pervasiva, sistemas em tempo real etc.

Quando o agrupamento ocorre por processo de trabalho ou função cada site é composto por um subprocesso do todo ou uma função especifica. Em um ambiente de DDS, pode significar, por exemplo, que o processo é subdividido em subprocessos, executados sequencialmente pelos sites.

O agrupamento por tempo tem o objetivo de aproveitar o tempo em diversos turnos para maximizar a produtividade. Quando um software é desenvolvido ininterruptamente, em um ambiente de DDS, essa característica é denominada desenvolvimento *round-the-clock.*

Agrupamento por output indica que as unidades são formadas por produtos que fabricam ou por serviços que prestam. São agrupamentos, por exemplo, sobre

linhas de produtos. Em um DDS, poderia identificar o tipo de produto em que o site é especializado. Por exemplo, sistemas ERP, sistemas CRM, etc.

Os grupos podem ser formados para lidar com diferentes tipos de clientes, justificando o agrupamento por cliente. Em um ambiente de DDS, essa é uma possível abordagem na qual grandes empresas, não só montam um escritório próximo ao cliente, mas também todo o conjunto de desenvolvimento.

O agrupamento por local indica que os grupos podem ser formados de acordo com o local na qual se encontram.

Mintzberg (2003) isola quatro critérios para o agrupamento. Tais critérios são considerados motivos considerados base (um site primário) para unir-se ou iniciar um ambiente de DDS.

- interdependência do fluxo de trabalho;
- interdependência de processo;
- interdependência de escala;
- interdependência social.

Siqueira e Silva (2004) ensinam que agrupamento é uma forma multidimensional composta por três aspectos: a quantidade de pessoas no grupo, a quantidade de grupos e os papéis exercido pelas pessoas. Dependendo da mescla desses aspectos, torna-se possível montar grupos autossuficientes e diminuir a necessidade de comunicação entre os grupos.

4.2.6 Granularidade de repasse

A granularidade de repasses está relacionada à forma na qual o site entrega seu subproduto para a rede. A granularidade deve considerar se o site repassa apenas linhas de código ou serviços implementados, considerando o fator temporal.

Em ambiente de desenvolvimento distribuído de software, a granularidade de repasse pode ser definida de maneira diferente entre os sites. Para tanto, uma série de fatores temporais e de forma de agrupamento afeta a decisão primária sobre o subproduto.

O repasse, por exemplo, pode ser de linhas de código, porém nesse caso, a codificação e as ferramentas são padronizadas entre os sites. Baseado nessa abordagem de codificação, o site pode repassar o serviço para outro, independentemente do código fonte.

Segundo Jalote e Jain (2004), a granularidade é definida principalmente pela homogeneização de processos, tecnologias e ferramentas da rede. A figura 32 é um exemplo de repasse, considerando a disciplina de codificação. Neste exemplo, a mais alta granularidade é repassar o serviço pronto para ser consumido (utilizado) e a mais baixa granularidade é a linha de código, necessitando, ser acoplada a um método e/ou função.

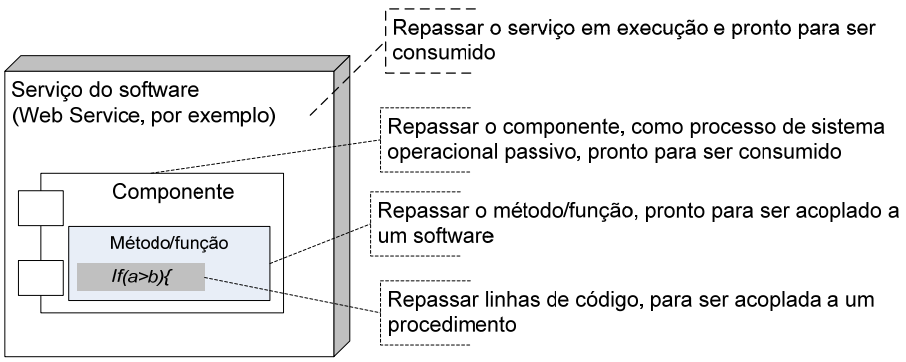

Figura 32 - Exemplo de granularidade de repasse na disciplina de codificação

CAPÍTULO 5 – ESTUDO DE CASOS

Este capítulo apresenta o resultado dos estudo de casos realizados em campo. O protocolo utilizado foi apresentado no Capítulo 3. Cada bloco do protocolo foi investigado em cada caso. Também foi avaliada, para cada caso, a aderência do modelo preliminar, apresentado no capítulo 4.

A tabela 6 indica algumas informações referentes ao trabalho de campo efetuado. A primeira coluna é o nome do caso adotado na tese. A segunda, "Tempo de interação", está relacionada ao tempo despendido com a visitação, entrevistas e conversas por meio de chat na empresa. A terceira coluna, "Tempo de análise", indica o tempo gasto para análise das informações obtidas nas gravações das entrevistas, e nas documentações de software a até mesmo do código fonte. A última coluna da indica as fontes de informações disponíveis em cada caso.

Tabela 6 - Informações sobre o trabalho de campo

Caso	Tempo de interação	Tempo de análise	Fontes de informação
Caso 1	6 Horas	36 Horas	- Arquiteto de software - Documentação do projeto - Código fonte do projeto - Desenvolvedor
Caso 2	4 Horas	26 Horas	- Desenvolvedor - Líder de equipe - Código Fonte - Documentação do projeto
Caso 3	2 Horas	10 Horas	- Desenvolvedor - Código fonte
Caso 4	5 Horas	26 Horas	- Líder de equipe - Gerente de qualidade - Documentação do projeto
Caso 5	4 Horas	16 Horas	- Gerente de desenvolvimento
Caso 6	3 Horas	17 Horas	- Líder de equipe - Desenvolvedor - Documentação do projeto

As próximas seções apresentam os resultados dos estudos. Em cada seção, há um conjunto de tabelas referentes ao protocolo de pesquisa.

5.1 Caso 1

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

O primeiro estudo de caso efetuado não foi feito em uma organização comercial. O protocolo de pesquisa foi aplicado inicialmente em um projeto de desenvolvimento distribuído de software efetuado por diversos laboratórios de pesquisas localizados em universidades distintas.

Do ponto de vista de desenvolvimento distribuído de software, o que se enfatiza neste caso é a quantidade e a diversidade de instituições que trabalham cooperativamente e de maneira distribuída para desenvolver um software como produto final. Foram 15 laboratórios de pesquisa, dos quais a maioria tinha domínio em desenvolvimento de software.

Os laboratórios de pesquisa que atuaram neste projeto são todos de universidades brasileiras. A união teve o objetivo de desenvolver um produto, sendo então uma parceria temporária e em função de um projeto em comum entre os diversos sites.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

Neste caso, somente um projeto foi analisado. Esse projeto de software visa ao desenvolvimento de um ambiente colaborativo que gerencia cursos e atividades de aprendizado, dando suporte ao ensino presencial e eletrônico. O sistema é um

conjunto de ferramentas de software desenvolvidas para auxiliar alunos, professores, instrutores e pesquisadores em suas ações.

O projeto analisado no Caso 1 basicamente trabalhou com a padronização de ferramentas e tecnologia. Toda a padronização determinada (somente ferramentas e tecnologia) era difundida e mesmo o site sem experiência mostrou-se disposto a utilizá-lo. Algumas dessas ferramentas e tecnologias foram: Eclipse, J2EE (1.4), UML 2.0, jUnit, ant, CheckStyle, Docbook, PMD, Enterprise Architect®, JBoss, Hibernate, CVS.

Cada grupo foi responsável por uma área arquitetônica do projeto, de acordo com sua experiência. Porém, embora uma macroarquitetura tenha sido definida, foi constatado que a interface de acoplamento entre os módulos não foi especificada com granularidade fina. Isso fez com que os módulos fossem interpretados como modelos conceituais e cada grupo utilizou sua experiência e seus métodos de desenvolvimento de software sobre ele.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

Havia dois níveis hierárquicos de laboratório, mas, todos tinham seu poder de decisão autônomo. Houve muito ajuste mútuo entre os sites. O nível hierárquico dos participantes do desenvolvimento era praticamente o mesmo. Cada grupo tinha a autonomia de modificar o projeto da codificação até mesmo nas visões conceituais sobre um mesmo objeto. Entretanto, houve impasse nesse caso, pois os grupos divergiam com relações a conceitos sobre um mesmo elemento da arquitetura.

Os sites eram determinados de acordo com os segmentos da arquitetura de solução. A figura 33 indica como foram organizados. De acordo com a experiência de desenvolvimento, um segmento da arquitetura foi atribuído como tarefa para o site.

A divisão de tarefas foi esporádica e determinada em reunião presencial ou em conferências. Para cada laboratório foi atribuído um conjunto independente de software, ou seja, um componente. Não há desenvolvimento contínuo de um mesmo componente de software por vários nodos.

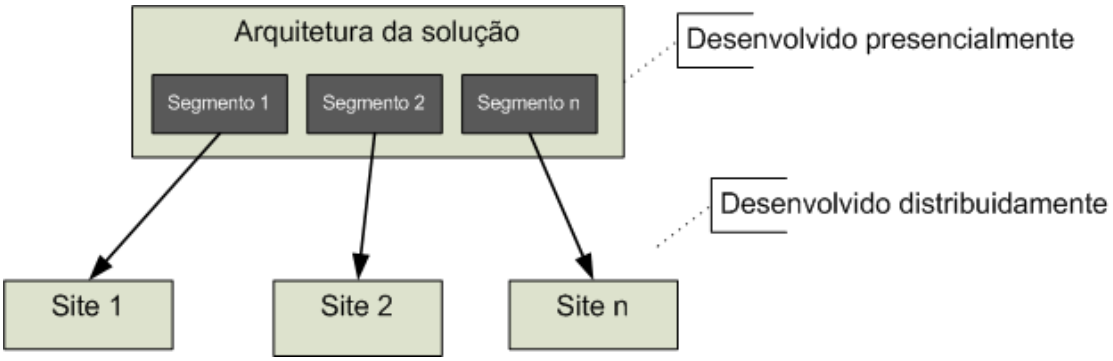

Figura 33 - Distribuição dos sites

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Inicialmente, para o desenvolvimento deste projeto, foi adotado um processo de software baseado em cascata. O processo baseia-se em *milestones* (pontos de checagem) para as transições e pontos de acesso e foi adotado pela facilidade em adicionar *milestone* e agilidade no gerenciamento de previsibilidade.

A adoção, porém, teve algumas desvantagens, tais como rigidez, dificuldade de estimar recursos e necessidades nas fases iniciais, de antecipar todos os requisitos na fase inicial e de administrar mudanças durante a dinâmica do projeto.

Para superação de tais limitações, o processo em cascata foi personalizado e passou a ser semelhante ao modelo espiral, tornando o processo interativo e incremental. Tal agilidade oferece algumas vantagens, pois: não é necessário conhecer todas as características no início, as funcionalidades vão sendo incrementadas a cada ciclo e há, portanto, maior facilidade para mudanças.

Sem processo inicial de configuração formalizado, foi notado que a integração entre algumas ferramentas era abstrata e baseada na experiência de cada site. A ferramenta ant, por exemplo, foi configurada e integrada com a IDE eclipse, baseando-se na experiência de cada site. Não havia um manual explicando como seriam os procedimentos de integração.

O problema tornou-se nítido quando a biblioteca de persistência foi configurada individualmente em cada site, pois houve um grande problema de integração e padronização de nomes e codificação utilizada.

O gerenciamento de configurações do projeto sofreu várias modificações do início ao fim do projeto. A primeira *baseline* estabelecida foi fundamentada em um conjunto de ferramentas, tecnologias e padrões de projeto que empregaria.

Um único repositório central, adotado primeiramente, fez com que o risco do projeto aumentasse. Eventualmente se o repositório central ficasse indisponível, todos os 15 nós não teriam possibilidade de acessar informações e produtos de trabalho.

Em uma segunda configuração, cada unidade montou seu repositório e o integrou com o repositório central, dando mais estabilidade ao processo de gerenciamento de configurações.

A configuração é ilustrada na figura 34. As setas numeradas com 1 indicam a sincronização entre os repositórios locais e o repositório central. As setas com o número 2 na indicam as operações de *check-in/check-out*.

Portanto, o emprego de um sistema distribuído de repositório dá uma disponibilidade maior ao sistema, uma vez que as informações são replicadas entre o repositório central e o repositório local.

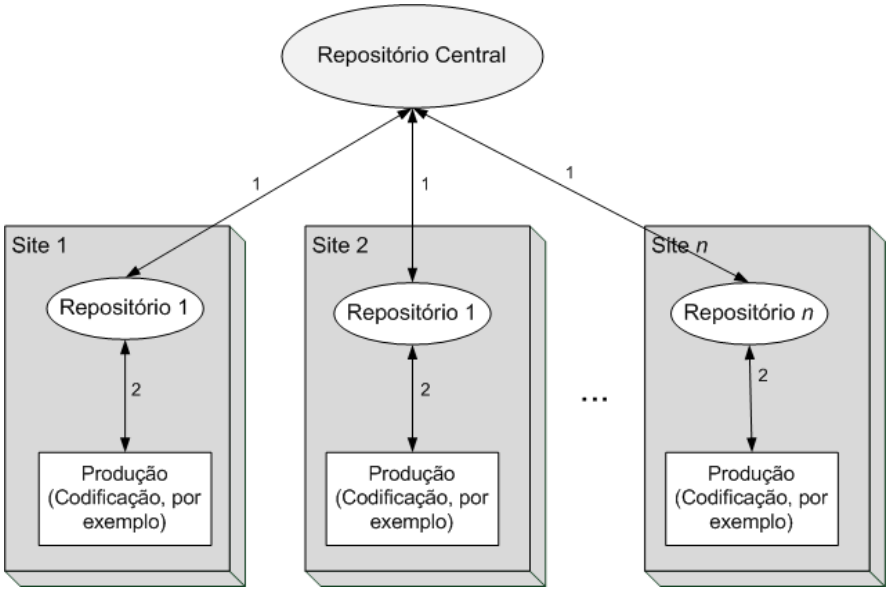

Figura 34 - Gerenciamento dos repositórios do caso 1

O projeto teve o acesso ao código controlado por meio de *check-in/check-out*. Se um site solicitasse o código para modificação, automaticamente impedia o outro de modificá-lo. No caso, a granularidade de modificação tem seu escopo voltado a módulos e arquivos e não a linhas de código.

Um conjunto de documentos foi empregado para manter a padronização do processo. Alguns documentos como os que representavam as especificações dos requisitos eram discutidos presencialmente. Um dicionário de terminologia foi empregado com o objetivo de reduzir a ambiguidade.

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

Um *framework* geral foi definido como premissas básicas que cada site seguiu para construir seu processo. Uma visão compartilhada do projeto foi criada. Essa visão foi estabelecida em reuniões presenciais ou em videoconferência.

Houve um fornecedor terceirizado, pois, um dos *sites* repassou o serviço de codificação. O processo teve que ser repassado e adaptado para o time de desenvolvimento.

Algumas tecnologias como UML *components* não eram de conhecimento dos desenvolvedores. Por isso, a solução encontrada foi adequar os diagramas para UML 2.0. O *site* que repassou a codificação monitorou o processo do time de desenvolvimento com o objetivo de garantir a integridade dos produtos de trabalho.

Cada *site* adotou um time específico, de acordo com as necessidades e capacidades. A atividade de delegação de tarefas foi representada na figura do expertise, que em reunião presencial ou conferência eletrônica decidia sobre a divisão das tarefas.

A partir de uma visão compartilhada, indicando os principais resultados esperados, foi montada uma infraestrutura de amparo, composta por processos e por um conjunto de conhecimentos básicos capaz de promover a integração do produto com o mínimo de impacto de retrabalho. Porém, ambiguidades oriundas de especificações, principalmente do modelo da arquitetura, e da distância física fizeram com que houvesse retrabalho no momento da integração entre os subprodutos de cada nó.

Com os subprodutos prontos, o projeto era efetivamente integrado. Antes da integração uma análise era feita para verificar e validar efetivamente se as interfaces eram compatíveis. Em alguns casos foi necessário retrabalho de codificação, pois foram detectadas incompatibilidades originárias de ambiguidades na definição das interfaces.

O expertise foi o responsável pelo controle das tarefas do site. Para isso, precisava conhecer praticamente tudo do projeto – dos modelos teóricos até a codificação.

Por causa desse conhecimento, o guideline de codificação foi desenvolvido a partir dos expertises dos sites. Dessa maneira, a codificação, embora padronizada, era definida pelo expertise, que além de complementar o modelo, embutia no estilo de codificação as experiências do grupo local.

Além disso, o complemento dado pelo grupo ao modelo, em conjunto com o "novo guideline" gerado tornava o processo de integração entre os módulos bastante complexo ao ponto de descartar e reimplementar partes do software.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

A integração entre subprodutos foi definida por meio de interfaces de troca de mensagens e ligação com o banco de dados.

A definição superficial do processo de integração fez com que alguns módulos precisassem de redesenvolvimento. O gerenciamento de mudanças é centrado em um repositório de informações. Atualmente, cada site possui sua base sincronizada com uma base central.

Os artefatos que não se acoplavam, ou seja, que não se adequavam ao restante maior da aplicação eram redesenvolvidos. Despendeu-se muito tempo para retrabalho, porém não há dados oficiais sobre o montante de tempo.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

Muitas tarefas foram desenvolvidas com o auxilio de videoconferência e

sistema de mensagem instantânea. É o caso, por exemplo, do design do produto e da integração, porém não houve armazenamento de histórico de comunicação.

5.1.1 Aderência ao modelo preliminar

Pelo fato de ser um projeto acadêmico e principalmente aberto, todas as fontes, além dos desenvolvedores e arquiteto puderam ser acessadas. Dessa maneira, foi possível efetuar uma análise do processo de desenvolvimento distribuído e do projeto, bem como analisar o que estava documentado e o que estava sendo gerado como código fonte.

A partir da análise foi detecto que padronização somente da arquitetura e das tecnologias não era suficiente para a progressão homogênea e natural do projeto.

Alguns problemas emergiram da inexperiência das equipes no desenvolvimento distribuído de software. Por exemplo, as interfaces que relacionavam os segmentos, em alguns casos, não foram bem esclarecidas, resultando em um problema posterior, durante a integração.

A maneira como o empacotamento foi criado para desenvolver um segmento da arquitetura foi escolhida de acordo com a experiência de cada time. O resultado final foi um conjunto de pacotes heterogêneos, que, embora para cada site funcionasse adequada e corretamente, fazia com que se uma terceira pessoa acessasse os códigos fontes, tivesse que compreender que cada segmento correspondia a uma parte da implementação da arquitetura e que cada uma destas partes seguia um padrão diferente.

A colaboração para o modelo foi principalmente a distribuição de tarefas. Seguindo padrões, o caso 1 comprovou que é possível segmentar e dividir as tarefas para que diversos sites distintos possam desenvolvê-lo.

Ainda que detectado superficialmente durante a confecção do modelo preliminar, a fase de integração de artefatos é um fator critico de sucesso para o projeto e para que o DDS funcione adequadamente.

Outra colaboração para o modelo preliminar foi a "gestão de configurações" indicada na figura 46. Quando em desenvolvimento, um projeto precisa de processos de gestão de configuração para permitir que os colaboradores

(codificadores, arquitetos, analistas) utilizem ferramentas integradas e que um desenvolvedor não interfira diretamente no trabalho de outro, além de poder utilizar sempre a versão mais atualizada e correta do documento ou código que está criando.

Foi detectada a importância e a complexidade do estado de "pronto para produzir". Quando uma organização prontifica-se a desenvolver de maneira distribuída, alguns processos de gerenciamento de configurações, padrões, tecnologias, processos de trabalho precisam ser explicitados e difundidos a fim de homogeneizar os sites. Se estas padronizações não ocorrerem, o risco do projeto aumenta e os problemas emergem durante a integração dos subprodutos.

5.1.2 Aspectos peculiares

Como o caso era o desenvolvimento de um projeto distribuído entre várias instituições de ensino, foi notada a heterogeneidade dos nodos com relação aos processos de trabalho. Constatou-se, também, que definir somente a tecnologia não foi suficiente para compatibilizar os produtos de trabalhos.

Houve intervenção humana nos processos e artefatos gerados a partir da macroarquitetura. Isso ocorreu, porque os papéis não foram definidos e a hierarquia entre os elementos era praticamente a mesma. A consequência foi o alto grau de ajuste mútuo entre os sites.

De certa forma, o arquiteto não tinha autonomia, pois suas funções eram distribuídas entre diversos sites. Embora tentasse adequar as interfaces dos produtos de trabalho, era subordinados aos sites e precisava seguir o que o site local adotava, tentando estabelecer um padrão entre todos os demais. Assim, todo site tentava padronizar os demais, a partir do conhecimento agregado localmente.

Um dos grandes impasses detectado em reunião foi a integridade entre o que foi definido e o que foi implementado; porque o programador/arquiteto alocado em um site podia modificar a estrutura de seu modulo de acordo com sua experiência, já imaginando uma interação futura. Criou-se, por causa disso, uma divergência relativamente grande entre o que foi definido em alta granularidade e o que foi implementado. Como consequência, houve muita recodificação.

Não foi constatada documentação de *deploy* no repositório do projeto. Segundo Bass, Kazman e Clements (2003), a estrutura do projeto do software, deve conter entre outros elementos um processo (ou *framework*) de instalação/implantação. Por isso, somente membros do projeto conseguiam efetuar operações de *check-out* do repositório e executar o software.

Na documentação do projeto não foram constatadas quais métricas foram empregadas.

5.2 Caso 2

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

A segunda organização na qual o protocolo de pesquisa foi aplicado, caso 2, é uma organização multinacional, com diversos *sites* de desenvolvimento instalados no Brasil, Índia, Estados Unidos, entre outros.

É uma organização especializada na área de tecnologia. Atualmente, presta serviços para bancos e demais organizações. A empresa possui certificação CMMI nível 5 em vários sites de desenvolvimento. Desenvolve software de forma global (GSD) habitualmente e novos colaboradores são preparados para trabalharem a distância.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

O projeto estudado neste caso, foi um software que implementa uma tese de doutorado sobre mineração de dados, cujos requisitos envolvem grande quantidade de cálculos e busca em base de dados distribuída.

O software foi desenvolvido para ser executado em ambiente Mainframe. Há integração entre aplicativos COBOL e aplicativo que foi desenvolvido na linguagem C.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

A pesquisa revelou que parte dos sites é composta por apenas uma pessoa (*home office*), associada a um dos sites de desenvolvimento. O agrupamento é decidido de acordo com um modelo hierárquico definido na figura 35.

Em cada projeto, o nível hierárquico mais alto é representado pelo gerente de negócios. Um degrau abaixo está o gerente de projetos. Quando o projeto entra em fase de desenvolvimento, o gerente de projeto é alocado para gerenciar um ou mais gerentes técnicos, que comandam os desenvolvedores. Os gerentes técnicos podem decidir se sua equipe trabalha em nodos compostos por um ou mais indivíduos.

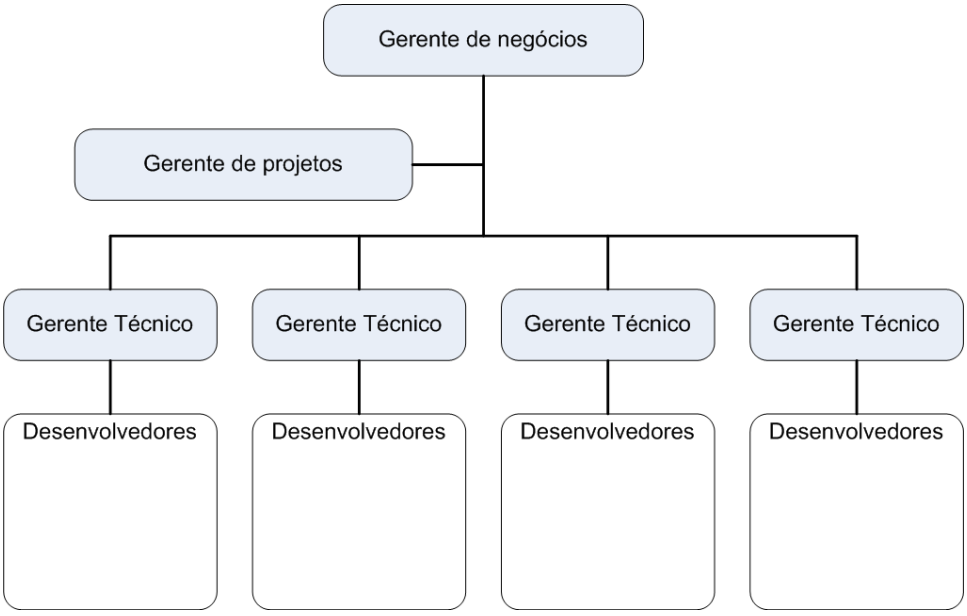

Figura 35 - Hierarquia nos projetos – Caso 2

A quantidade de sites varia de acordo com o projeto, o qual pode ser total ou parcialmente desenvolvido de forma distribuída. O projeto analisado tinha aproximadamente 25 sites.

A coordenação é efetuada pela padronização de processos e tecnologias. O desenvolvedor é treinado em uma unidade centralizada, por isso agrega o processo organizacional e toda a conduta da empresa.

A agregação dos times de desenvolvimento depende do projeto. Há homogeneidade relativamente grande dos processos organizacionais, inclusive os processos de trabalho e todo desenvolvedor deve seguir rigidamente os processos em que foi treinado, seja de levantamento de requisitos, integração, codificação, teste e demais processos de desenvolvimento. Por essa razão, a organização mantém conjuntos de especialistas em determinadas áreas, que os reúnem, de maneira centralizada e/ou distribuída de acordo com os requisitos do projeto.

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Os sites possuem alto grau de homogeneidade, basicamente centrada em processos e condutas organizacionais (todo colaborador é treinado no nodo central) e nos artefatos, que dependem de treinamentos e do projeto. O gerenciamento de configurações é elaborado de acordo com cada projeto. A utilização de ferramentas de controle de versão e de ambientes de desenvolvimento integrado é intensa.

Um desenvolvedor deve efetuar o *deploy* e entregar o produto para o gerente técnico. Sempre seguem o mesmo processo: o desenvolvedor principal acessa o repositório, varre o código procurando erros (caixa branca), otimiza (se necessário), efetua o *deploy* e entrega para o gerente técnico. Neste instante, são efetuadas e testadas as integrações.

A organização possui certificado CMMI nível 5 e grande parte de seus projetos de software utilizam uma versão customizada e proprietária do RUP (Rational Unified Process).

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

Todos os sites são tratados como organizações autônomas. Cada um deve estimar seu tempo de trabalho para desenvolver a tarefa, antes de desenvolvê-la. Essa informação fica armazenada em uma base de dados e é utilizada como uma métrica do projeto. A figura 36 indica o fluxo de distribuição de tarefas.

A tarefa é iniciada pelo gerente de negócios e segmentada até chegar ao desenvolvedor. Para estimar a duração e o custo do projeto, cada site deve informar ao seu nível hierárquico superior quanto tempo leva para desenvolver determinada tarefa. Assim, o gerente de negócios tem dados estimados para cada projeto e pode compará-lo com uma base de dados de projeto já existente.

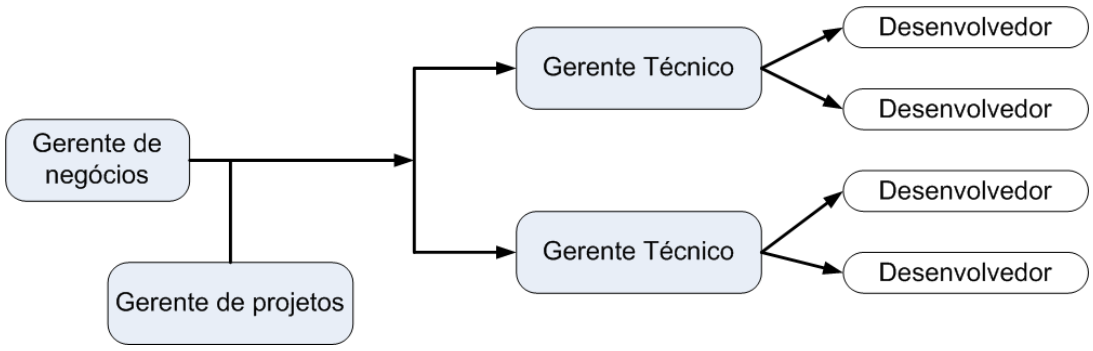

Figura 36 - Distribuição de tarefas

O papel de distribuidor de tarefas é extenso. Qualquer parte do projeto

pode ser desenvolvida de forma distribuída e qualquer individuo pode ser um distribuidor de tarefas, desde a gerência de negócios até a codificação.

As tarefas raramente são rearranjadas e o projeto é preparado para ser produzido de forma distribuída. Raramente há impasses (dependências funcionais, por exemplo). Qualquer artefato pode ser desenvolvido distribuidamente e é atômico. Todo desenvolvedor recebe um conjunto básico de documentação dos requisitos que precisa desenvolver.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Todos os artefatos podem ser desenvolvidos de maneira distribuída, desde que os padrões definidos sejam seguidos rigidamente. Neste caso em particular, somente o levantamento de requisitos foi centralizado.

Cada componente foi desenvolvido de maneira autônoma, não dependendo dos demais. Quando ocorria uma dependência, o desenvolvedor ficava ocioso, porém não houve realocação de desenvolvedores no projeto, porque o custo de realocação e retorno do desenvolvedor no projeto é alto.

Poucos artefatos tiveram problemas de integração. Todos os desenvolvedores recebem uma documentação vasta e muito bem detalhada de suas atividades, evitando, assim, problemas de integração e interpretação.

Houve conflito no idioma, que embora seja inglês, o sotaque e a influência de diversos países dificultam a compreensão nas videoconferências realizadas. No projeto, havia mexicanos, americanos, indianos e brasileiros, pois a ideia de distribuir a produção é otimizar o projeto *round-the-clock*. A redução de tempo e custo são os

principais argumentos usados pela organização para desenvolver distribuidamente.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

Basicamente há dois tipos de software de comunicação entre os sites. O primeiro é um mensageiro instantâneo; o segundo, um software de gerenciamento de groupware. O software de gerenciamento de *groupware* controla serviços de comunicação assíncrona e o agendamento de reuniões, e-mail etc. As interações assíncronas são utilizadas para agendar reuniões síncronas, enviar documento a grupos de desenvolvedores. Todos os dados são armazenados e cada projeto tem seu histórico de comunicação.

O histórico de comunicação é empregado empiricamente para determinar quais treinamentos a organização precisa desenvolver. O volume intenso de comunicação é armazenado, mas muito não é aproveitado, pelo menos até então, pelo fato de a organização considerar inviável extrair outras informações, senão informalmente requisitos de treinamento dessa fonte.

O site central controla as licenças de software em todos os sites. Tudo deve ser instalado/desinstalado de acordo com a autorização do nó central.

No projeto em questão pouco foi desenvolvido presencialmente. Isso ocorreu porque o gerente de negócios era dos Estados Unidos, o gerente de projetos do Brasil, alguns gerentes técnicos eram do Brasil, da Índia e dos Estados Unidos.

5.2.1 Aderência ao modelo preliminar

Demonstrar a importância da homogeneização dos sites foi uma das principais colaborações do projeto. Quase que todos os projetos desenvolvidos pelo caso 2 são globais, aproveitando o fuso horário e as características particulares de cada time e região. O fato de todos os colaboradores serem treinados em processos

organizacionais possibilita uma interação mais natural durante o desenvolvimento do projeto.

A padronização das ferramentas integradas de desenvolvimento foi identificada também como um fator crítico para o sucesso do projeto. Os procedimentos de gestão de configuração são claros e evidentes, operações de acesso ao repositório, busca, restauração e atualização, são constantes e controlados sistematicamente.

A padronização do código também foi uma grande colaboração, pois a interpretação do código fonte desenvolvido por um colaborador e editado posteriormente por outro é mais simples. Todo código deve seguir um padrão rígido de comentário e formatos (endentação, nome e tipo de variáveis, empacotamento, nome de classes, métodos ou funções e arquivos).Por causa disso, todos os desenvolvedores compreendem com certa facilidade o código desenvolvido por outro.

5.2.2 Aspectos peculiares

Uma das unidades pesquisadas foi um desenvolvedor que atuava no regime de *home office,* que era membro de um site e respondia diretamente a ele. A estrutura do *home office* reflete a estrutura que o colaborador utiliza dentro do site. Havia dois computadores, um serviço ADSL com uma VPN que o ligava ao site. Toda ligação era feita por meio de VOIP, utilizando a própria estrutura de trabalho.

O ambiente de trabalho do colaborador em sua casa era claro e organizado. Ele informou que um dos treinamentos mais exigidos pela empresa era de organização do ambiente de trabalho. Segundo ele, as rotinas de guardar todo recurso visível, trancar notebooks com cadeados, desligar ou hibernar computadores, trancar os papéis visíveis sobre a mesa em uma gaveta são comuns dentro da empresa e se o funcionário deixar de fazer algumas destas atividades, mesmo que seja apenas para atender um breve telefonema, sofrerá penalidades determinadas pela organização.

O código fonte produzido no projeto estudado era todo comentado. Os comentários seguem os padrões estabelecidos no inicio do projeto. Praticamente todo trecho de código (função, método, algoritmos) era documentado. Por isso, que a

compreensão do código fonte por um desenvolvedor externo ao projeto ou um novo desenvolvedor não era tão difícil.

5.3 Caso 3

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

O terceiro caso é uma organização nacional, cujo principal ramo de negócio é o desenvolvimento de software. Durante a aplicação do protocolo, possuía certificação CMMI nível 2, hoje tem certificação CMMI nível 3. Neste trabalho, esta empresa será denominada como caso 3.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

Este caso aborda um projeto de software cujo objetivo é controlar vendas por meio de laptops. O software foi desenvolvido para baixa plataforma, utilizando tecnologia web.

Embora a organização utilize uma versão adaptada do RUP, o processo utilizado nesse projeto de desenvolvimento teve o ciclo de vida basicamente em cascata.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

Há uma quantidade pequena de sites, agregados basicamente por função e local. O motivo de divisão e criação de novos nodos é o custo e a qualidade, pois de acordo com a empresa, no interior de São Paulo, onde um dos sites está alocado, o custo dos desenvolvedores é até 20% menor do que na capital paulista.

Há padronização nos artefatos (documentação, empacotamento e código fonte) e também nos processos de trabalho. As unidades distantes da central foram implantadas exclusivamente para atenderem a demanda do site central. Por essa razão, não há entrada de requisitos a partir de um *site* de desenvolvimento.

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Todo site é treinado por um colaborador do *site* central. Este colaborador desloca-se da central até o nodo de desenvolvimento onde treina os demais colaboradores. Desta maneira, os processos de trabalho são difundidos e homogeneizados.

A configuração de cada site é controlada principalmente por três servidores. O primeiro servidor é denominado servidor de desenvolvimento, neste há um sistema de controle de versões, e da instalação das ferramentas utilizadas para desenvolver o software. O segundo, o *active directory*, é responsável por autenticar os usuários do site local e criar uma área de diretório para cada usuário. O terceiro , o *Proxy*, fornece conexão com internet para o site local.

A ideia é que todos os sites tenham um sistema de controle de versões sincronizado com um repositório central, embora nem sempre isso seja possível devido a restrições de largura de banda.

Há uso constante de ferramentas integradas de desenvolvimento, que

permitem ao desenvolvedor acesso constante e trivial aos repositórios do projeto.

Os subprodutos são desenvolvidos sobre um *framework* criado pelo site central. O *framework* contém um conjunto básico de aplicações CRUD em banco de dados e documentação para estendê-lo e utilizá-lo. Por essa razão, a integração entre subprodutos é considerada trivial pelos desenvolvedores e há poucos erros oriundos de integrações detectados durante a fase de teste.

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

A padronização para distribuir tarefas está fortemente centrada na especificação dos requisitos. A tarefa recebida é de implementação, ou seja, o que foi desenvolvido em rede foi basicamente código. A tarefa vem na forma de documento de especificação de requisito. Os sites são homogêneos, pois o coordenador, alguém treinado na unidade central, da área é responsável pelo treino da equipe. Ele repassa a tarefa para seu time e controla as tarefas internamente.

A figura 37 é um organograma da distribuição de tarefas. A área denominada de *Quality assurance* é a mediadora entre o que a empresa denomina de *Project Office* e *Business*. Por meio dessa, os parâmetros de qualidade são aferidos e refinados.

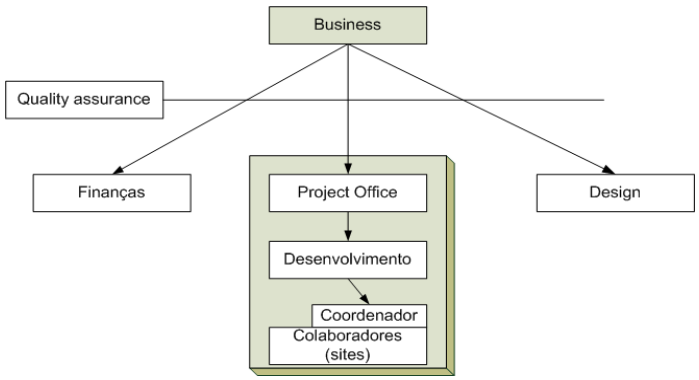

Figura 37 – Organização e distribuição de tarefas do caso 3

A distribuição de tarefas inicia-se pela área da organização denominada *Business*. O *Project Office* recebe a tarefa e inicia-se o desenvolvimento. Neste caso, a codificação é tratada como desenvolvimento e somente ela é desenvolvida de maneira distribuída.

O distribuidor de tarefas é o *Project Office*. As tarefas são divididas em formas de requisitos. Cada grupo recebe um conjunto de requisitos bem detalhados, com diagramas de caso de uso e outros. O distribuidor de tarefas está associado fortemente ao arquiteto de software. O P.O. também estabelece o *framework* de trabalho.

Os parâmetros de qualidade estão fortemente associados a cálculos de ponto por função. Quando um novo site é aberto, o Quality Assurance o induz a desenvolver um projeto inicial sob condições críticas de prazo. Dessa forma, o Q.A. obtém as métricas iniciais do site.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Os artefatos desenvolvidos de maneira distribuída são basicamente códigos fontes. A iteração sobre o artefato é uma extensão dos componentes CRUD que o desenvolvedor recebeu no framework desenvolvido pelo site central.

A linguagem utilizada neste projeto foi PHP. Toda a estrutura de empacotamento, padronização de variáveis, nome dos arquivos e diretórios, padronização de comentários está explicita no *framework* criado pelo site central.

Não há exatamente uma documentação de toda a padronização de codificação, o desenvolvedor utilizava os códigos disponíveis no framework como *guideline* do projeto de software.

Os testes eram realizados pelo próprio desenvolvedor e pelo *quality assurance*. Nesse caso, seriam testes específicos e de caixa branca, visando a detectar falta e/ou falhas de padronizações.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

A comunicação entre os sites é efetuada principalmente por e-mail. Não é comum a organização utilizar sistemas de mensagem instantânea, porém em alguns casos utilizam videoconferência para treinamento a distancia. A comunicação entre sites não foi armazenada, pois a empresa considerou desnecessário. Houve transporte de CDs, nos quais haviam *frameworks* de trabalho, entre uma unidade e outra.

O nível hierárquico denominado como *Project Office* é responsável por integrar os artefatos. O *Quality assurance* faz revisão caixa branca no código. A integração era realizada pelo site de desenvolvimento e pelo *Project office* em algumas instâncias.

5.3.1 Aderência ao modelo preliminar

A colaboração deste projeto para o modelo preliminar concentra-se basicamente no primeiro ciclo de desenvolvimento do site. Fica evidente a importância da homogeneização inicial de processos, tecnologias e também um *framework* comum de trabalho, no qual todos os desenvolvedores utilizam as mesmas ferramentas integradas.

5.3.2 Aspectos peculiares

O ambiente de trabalho é bem informal. O site analisado estava alocado dentro de uma instituição de ensino que de certa forma exerce forte influência sobre a organização. O setor de produção fica no mesmo recinto onde os servidores estão alocados. Dessa maneira, qualquer estudante com acesso à produção também tinha, ao menos, acesso físico aos servidores.

A organização não obriga os desenvolvedores, que na maioria são alunos da instituição de ensino, a treinar na matriz. Os processos enfatizados pela organização são ligados diretamente a codificação. Processos de trabalhos como o caso 2 (organização do ambiente de trabalho) não foram identificados entre os desenvolvedores.

As métricas iniciais do site foram estipuladas a partir de um projeto pioneiro. A central induz o site a produzir determinado software, sob condições extremas, com tempo e custo críticos. A partir dos primeiros resultados, a matriz consegue estabelecer os primeiros valores das métricas empregadas em projetos de software.

5.4 Caso 4

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

O caso 4 foi efetuado em uma organização multinacional, com certificação CMMI 5. Empresa exclusivamente de desenvolvimento de software, atua em diversos segmentos, mas, principalmente no financeiro. A organização tem inúmeros clientes pelo mundo inteiro e sites com grande quantidade de funcionários. Há dois sites no Brasil, um com 700 funcionários e outro com 1500 funcionários aproximadamente.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

Os projetos desenvolvidos seguem padronização fortemente associada aos processos de trabalho. São padronizados os processos de trabalho, composto por atividades rotineiras que os desenvolvedores precisam efetuar, os processos de consultoria e até os processos de relacionamento com o cliente.

Os projetos de software envolvem softwares desenvolvidos para arquitetura Mainframe. Porém a organização também desenvolve projetos de software de integração entre sistemas legados e arquitetura Java e/ou .Net.

Todo ciclo de vida de um tipo de projeto é determinado em uma base de processos e, a partir de então, todos os projetos similares devem seguir o mesmo ciclo. Dessa forma, há padronização no desenvolvimento de projeto e consequentemente aplicação mais eficaz relacionada a métricas de software.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

A organização possui vários sites localizados no Brasil, na Argentina, na Índia, nos Estados Unidos etc. Os alocados no Brasil e na Argentina foram consolidados como sites de *offshore* para o site Americano.

A unidade centralizadora é o site nos Estados Unidos. Naquela unidade, são definidos os padrões arquitetônicos para as soluções e a formalização dos processos

utilizados por todos os sites. Os demais sites foram criados inicialmente como unidades de *offshore*.

Nos sites há equipes especializadas em determinadas tecnologias e/ou processos. Há possibilidade de sites trabalharem com colaboradores em *home office*. Neste caso, o colaborador segue os mesmos procedimentos de seu site.

A condução do negócio é centralizadora. A figura 38 indica o início do processo de desenvolvimento de software identificado no caso 4. Alguns sites possuem unidade de condução de negócio e, nesse caso, são responsáveis pela venda e/ou entrega dos produtos. Para cada projeto, os líderes, arquitetos e gerentes determinam como as tarefas serão distribuídas e como será a interação entre os sites.

São criadas contas de cliente, as quais podem conter vários projetos. Cada projeto pode ser desenvolvido em *offshore*, parte pode ser desenvolvida *nearshore*.

A hierarquia está explicita na figura 40. Para cada cliente há uma conta associada, para cada conta pode haver vários programas associados e para cada programas pode haver vários projetos de software associados.

Somente os sites com processos de condução de negócio podem conduzir a venda para o cliente. Por isso, somente eles são centralizadoras e iniciam a distribuição de tarefas.

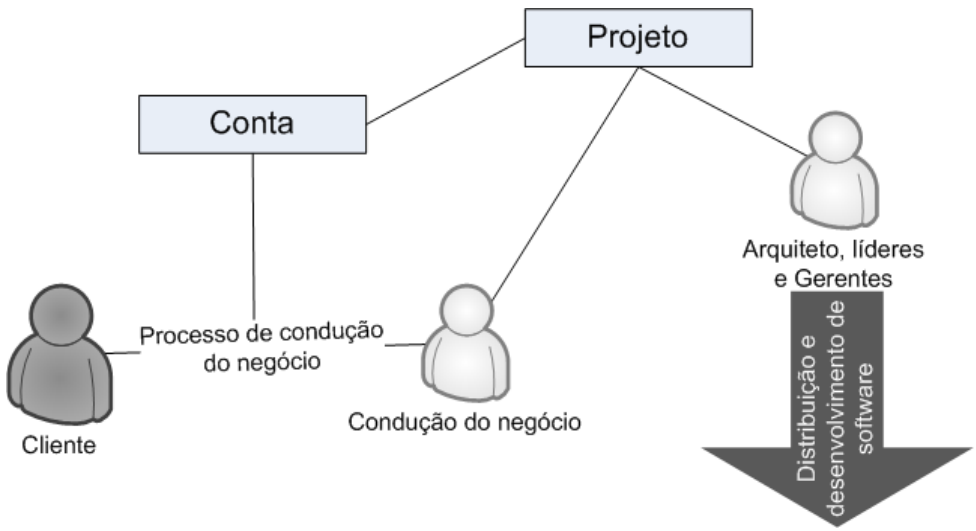

Figura 38 - Início do projeto e desenvolvimento distribuído no caso 4

A figura 38 indica uma relação em que o projeto possui arquitetos, líderes de desenvolvimento e gerentes que determinam como os sites devem interagir. Esse

processo de interação entre os sites é determinado pela padronização de processos de trabalhos, artefatos e tecnologias de desenvolvimento utilizadas (IDE, repositórios, e outros).

A base de processos da organização contém todos esses processos, mas há flexibilidade, pois um processo pode ser adaptado ou, raramente, criado para atender um projeto. Em seguida, é armazenado na mesma base, podendo ser utilizado por demais projetos similares.

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Os sites são homogêneos e utilizam os mesmos processos e ferramentas de desenvolvimento. Há projetos em que a organização aloca um *site* dentro da própria organização do cliente. Nesse caso, há uma interface entre o site que está alocado no cliente e um site de desenvolvimento da empresa. A interface deve padronizar documentos e código que se originam no cliente e entram na organização. A figura 39 ilustra este processo.

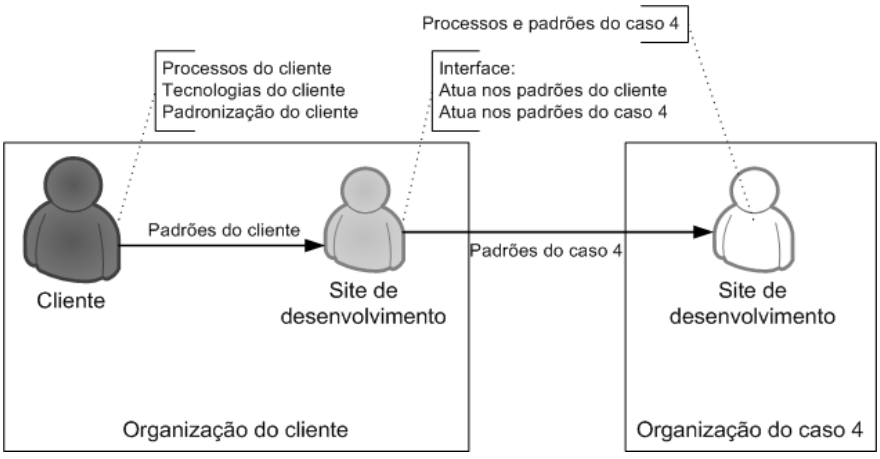

Figura 39 - Desenvolvimento de software alocado no cliente - caso 4

Segundo Costa (2003), é possível aumentar a qualidade e a produtividade de uma fábrica de software com a padronização do processo de recebimento de serviços de construção de software. A figura 39 é uma ilustração que indica que há uma interface de padronização entre objetos que não estão nos padrões da organização e os objetos padronizados. A empresa explica que o trabalho de homogeneizar mantém sua produtividade e qualidade, além de padronizar as métricas de produção.

Toda organização trabalha sobre um *framework* comum de processos de engenharia. A base comum que o processo deve seguir é definida durante o processo de arquitetura do software. Essa base possui processos de gerenciamento de configurações que todos colaboradores de um mesmo projeto devem seguir rigidamente. Também na fase de arquitetura, as tarefas são divididas entre os sites de desenvolvimento.

A integração é definida durante o processo de arquitetura. A definição sistemática de ferramentas de desenvolvimento integradas, junto a processos de codificação, empacotamento, teste e integração garante que durante o processo de integração de subprodutos o resultado atenda as necessidades de custo/benefício definidas no início do projeto.

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

O arquiteto (ou arquitetos) de software é o elemento distribuidor de tarefas. Ele projeta a solução de acordo com os recursos disponíveis. Durante o processo de arquitetura, são considerados a distribuição de tarefas e os processos de gerenciamento de configurações, testes, implantação, entre outros. Isso minimiza ou

elimina os conflitos ocasionados pelo desenvolvimento distribuído, que, no caso 4, é global na maioria das vezes.

Após integração, o caso 4 possui uma organização interna especializada em testes de integração. Dependendo da complexidade, a aplicação é levada a essa organização interna para que ela possa fazer as validações e testes. Dessa maneira, a base de processos é realimentada, fazendo com que as soluções de problemas já resolvidos sejam institucionalizadas por toda a organização.

A empresa considera que há três maneiras de desenvolver distribuidamente um projeto. Na primeira, há um gerente de projeto em um site e um líder de desenvolvimento em outro. Na segunda, o líder de desenvolvimento e o gerente de projetos estão em um mesmo site, porém coordenam trabalho de outro site. E a outra forma ocorre quando o projeto está alocado em um site com gerente de projetos e líder de desenvolvimento. As três maneiras são ilustradas na figura 40.

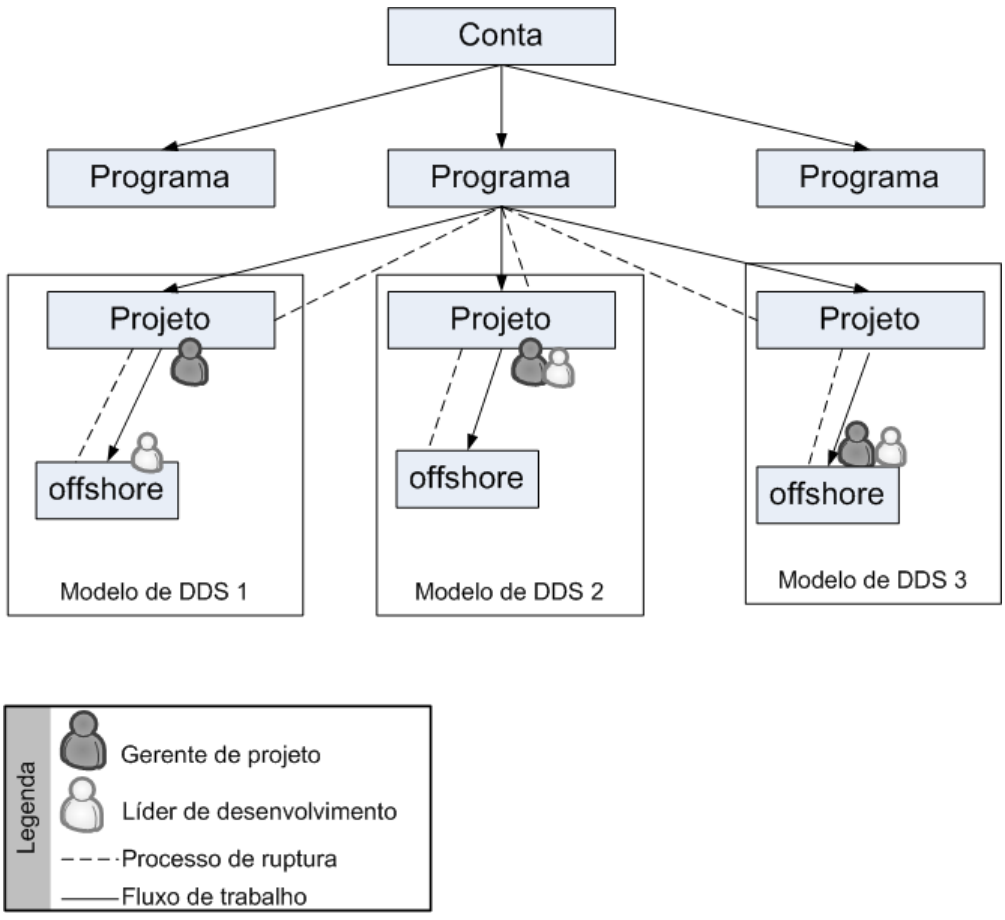

Figura 40 - Fluxo de trabalho e alocação de tarefas do caso 4

Na aplicação do protocolo de pesquisa, foi detectado também um processo de distribuição de tarefas que a organização denomina processo de ruptura. O processo está indicado na figura 40 como uma linha tracejada.

O processo de ruptura é responsável por segmentar o projeto de forma que ele possa ser desenvolvido de maneira distribuída. Esse processo se inicia após o estabelecimento de um programa para um cliente. Não há regra fixa para a segmentação da conta em programas e nem do programas em projetos. Depende da complexidade da aplicação e da localização do cliente e do site.

O conceito de programa, indicado na figura 40, segundo a organização é o mesmo conceito de programa explicado no PMBoK (PMI, 2004).

O desenvolvimento *follow-the-sun* é considerado complexo para os projetos da organização. O repasse de código para outro desenvolvedor tem custo de retrabalho e de microgerenciamento, fazendo com que o custo final do produto seja maior e consequentemente torna inviável esse tipo de desenvolvimento.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Todo artefato do projeto pode ser produzido em rede. A integração é efetuada por um conjunto de processos comuns e por um conjunto bem estruturado de ferramentas de integração e desenvolvimento.

Todo artefato é padronizado antes de entrar em produção. Artefatos oriundos de clientes são adaptados para ingressarem nos sites a fim de que o processo possa ser seguido e as métricas possam ser mantidas (figura 39).

O processo de integração e ferramentas definidas na fase de arquitetura garantem que os subprodutos possam ser integrados da maneira mais trivial possível,

minimizando impactos oriundos da distância física. Todos os artefatos são rastreados e o gerenciamento de configuração é considerado fator crítico para o sucesso dos projetos.

Não há informação de que a produção em rede é mais veloz que o desenvolvimento centralizado, entretanto, há métricas que indicam o custo da produção e também a proximidade com o cliente, justificando o desenvolvimento distribuído no caso 4.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

Toda comunicação síncrona e assíncrona do caso 4 é armazenada. A comunicação síncrona é armazenada somente no equipamento do desenvolvedor e é utilizada para fins de auditoria. A comunicação assíncrona (e-mail) é armazenada em servidores da organização com o objetivo de auditoria. O custo de minerar informações na base de dados de comunicação entre sites é alto. Por isso, até o momento da aplicação deste protocolo era inviável efetuar varreduras em busca de informações.

Toda a comunicação é efetuada por meio de Internet. Há uma estrutura com VPN para a troca sigilosa de informações entre os sites.

5.4.1 Aderência ao modelo preliminar

Durante o desenvolvimento do projeto, foi detectado que a organização já tem um tratamento específico quando o software é desenvolvido de maneira distribuída. Na fase de arquitetura do software, iniciam-se os primeiros procedimentos que o caso 4 denomina ponto de ruptura.

O ponto de ruptura de um projeto, segundo essa organização ocorre quando é preparado para ser desenvolvido de maneira distribuída.

5.4.2 Aspectos peculiares

A organização perpetua o máximo possível seus processos de trabalho. Mesmo quando há uma organização cliente envolvida, é desenvolvida uma interface que mantém homogêneos os artefatos do caso 4.

A organização considera o teste de software como parte de seu valor agregado ao produto final. Por isso, criou outra organização, também no âmbito mundial, para testes de software. Nessa organização de testes, são validados os modelos de arquitetura utilizados nas soluções finais.

5.5 Caso 5

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

O quinto caso é uma organização multinacional, que desenvolve principalmente software para tecnologia embarcada. O principal ramo de negócio não é software e sim produtos eletrônicos. Os softwares produzidos pela organização têm basicamente duas finalidades. A primeira é ser embarcado no dispositivo eletrônico comercializado para o usuário final. A segunda é atender a demais organizações que desenvolvem software para os seus dispositivos eletrônicos, sendo então os softwares classificados como kits de desenvolvimento.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

O departamento de marketing inicia o projeto de acordo com as necessidades identificadas no mercado. Os requisitos são levantados pelo departamento de marketing da organização e posteriormente analisados com a equipe de projetos. A figura 41 indica como é o processo inicial de um projeto.

A organização desenvolve hardware e software para um produto final, entretanto, a abordagem deste trabalho restringe-se ao software.

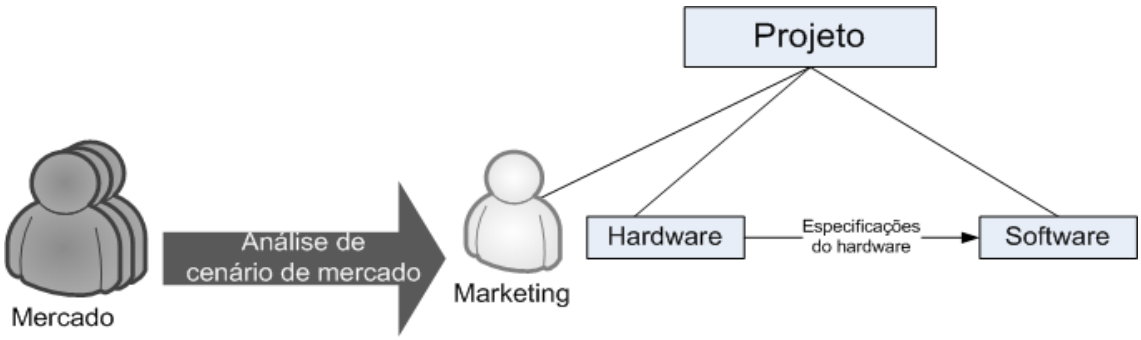

Figura 41 - Início de um projeto no caso 5

Todo o projeto segue um ciclo de vida norteado por uma base de processos existentes na organização e armazenados em um banco de dados. Cada projeto segue uma arquitetura de solução específica, de acordo com os requisitos e com o hardware. Entretanto, há um conjunto comum de tecnologias utilizadas em quase todas as soluções. É caso do sistema operacional e de alguns processadores (*hardware*).

Os sites são autônomos quanto a certificações. O site analisado tinha CMMI 5 e ISO, entretanto nem todos os sites têm a mesma certificação.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

Todos os sites são organizações autônomas, com um número relativamente grande de funcionários e desenvolvem software para um determinado departamento de marketing, independente de o departamento pertencer ou não à unidade. A padronização, nesse caso, possibilita que vários sites possam desenvolver um mesmo projeto, embora cada projeto tenha uma solução específica.

Os sites são distribuídos de acordo com uma estratégia de mercado, voltada para a distribuição do produto final e não do software. Por exemplo, o site do Brasil tem como atividade de negócio, coletar requisitos do mercado nacional e de países vizinhos. Entretanto, os projetos de software são divididos em diversos sites para atender a demanda específica de um determinado projeto e aproveitar o alto grau de especialização em determinadas tecnologias/processos por sites específicos.

Há mais de 30 sites no mundo com diversas finalidades. Cada um possui um, assim denominado pelo caso 5, *pool de recursos,* que é um conjunto de recursos (humanos, processos e tecnologia) para lidar com determinadas soluções. Esse conjunto se repete nos diversos sites de desenvolvimento. Com isso, é possível distribuir o projeto de um site para os demais.

Há sites que iniciaram somente como unidades de *offshore*, o caso do Brasil, para depois se tornarem unidades de negócio também (criando um departamento de mercado) e efetuarem operações em *nearshore*.

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Há processos globais que todos os sites devem seguir rigidamente. Entretanto, pode haver diferença entre *sites*, de acordo com o perfil do departamento de mercado e de acordo com o produto final.

No caso 5, o problema de integração foi mais explícito, pois a arquitetura da solução pode variar muito, exigindo um processo especializado de integração. Embora a organização possua um conjunto padronizado de ferramentas e processos de trabalho, erros oriundos da integração podem advir de duas maneiras.

A primeira ocorre quando os componentes não se integram. Nesse caso, houve alguma no processo de *design*. Alguma interface não foi especificada adequadamente de forma que os módulos de software precisam retornar para a fase de projeto.

O segundo erro pode ocorrer quando uma interação não prevista surge no teste. Após integração, o software é testado e é plausível que alguma interação, oriunda de alguma rotina de baixo nível possa provocar diferentes resultados, quando utilizada no desenvolvimento de algum aplicativo de alto nível.

A figura 42 ilustra o processo. Processos de integração e de configuração são considerados fatores críticos do processo.

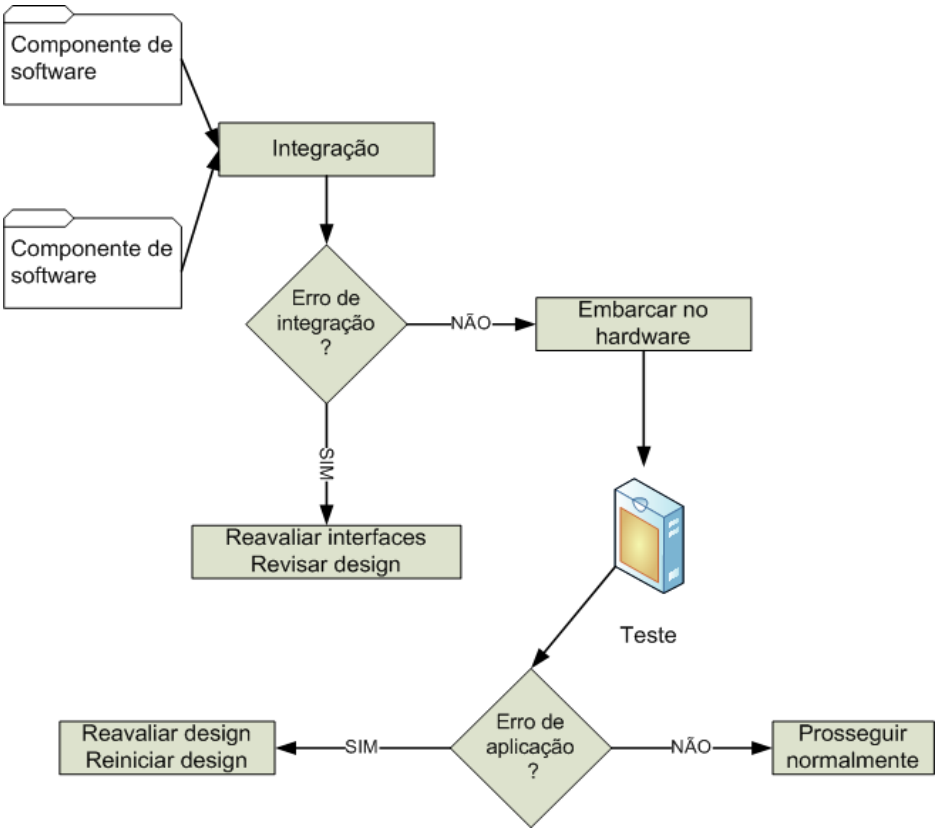

Figura 42 - Integração e teste do caso 5

A organização utiliza ferramentas de desenvolvimento prontas, disponíveis no mercado, porém desenvolve também parte de suas ferramentas de desenvolvimento e as disponibiliza para seus parceiros.

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

A complexidade do produto final determina a maneira como as tarefas do projeto de software serão distribuídas. O projeto da solução é elaborado de forma que as tarefas possam ser desenvolvidas em DDS. Dessa maneira, o arquiteto (ou arquitetos) de software é o distribuidor de tarefas.

A distribuição de tarefas é determinada de acordo com o produto final. A figura 43 explicita como um projeto é iniciado e como as tarefas são inicialmente distribuídas.

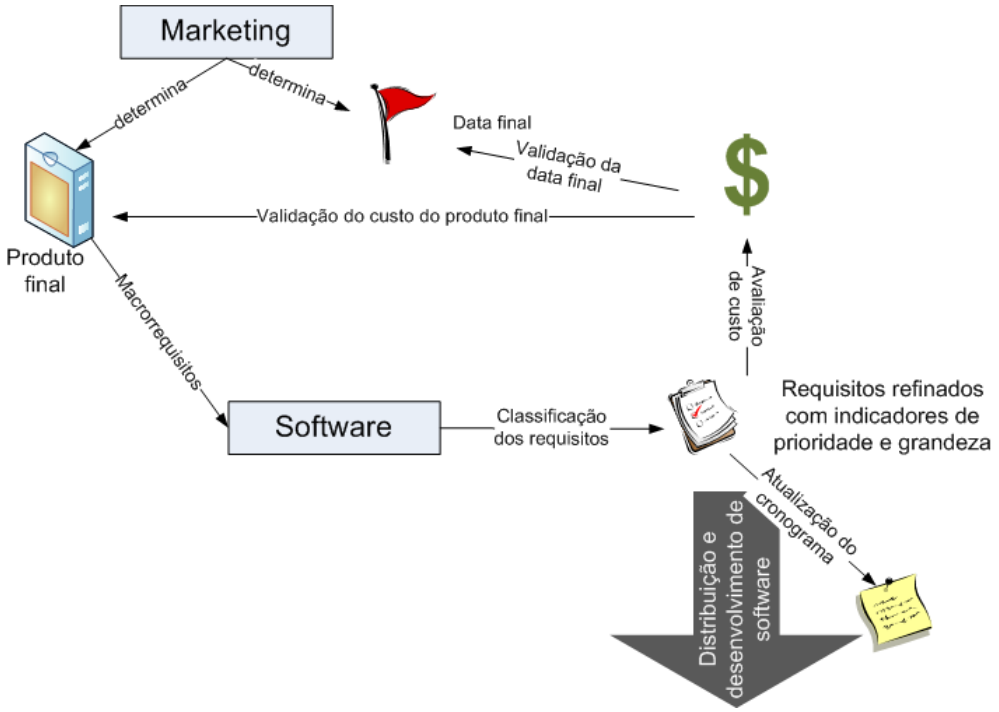

Figura 43 - Inicio do projeto e desenvolvimento distribuído no caso 5

Um mapeamento das interdependências é elaborado antes de distribuir as atividades. O processo ocorre assim que a equipe de desenvolvimento de software recebe os macro requisitos do departamento de marketing. Nessa mesma etapa, as interdependências são resolvidas previamente por meio de planejamento. Os conflitos gerados por dependências não resolvidas nessa parte são considerados, pela organização, uma ciência.

O custo da ociosidade é alto demais para o caso 5. Por isso, os sites dificilmente ficam ociosos. Isso ocorre, porque, há homogeneidade eficaz entre eles e são, constantemente, encaminhadas atividades de desenvolvimento ou do departamento de marketing local ou do departamento de marketing de outro site.

A organização mantém parcerias com terceiros para a garantir a escalabilidade da produção e a distribuição de tarefas sem sobrecarregar sua estrutura operacional. Este conceito é descrito detalhadamente por L'Erario (2009).

A organização do caso 5 não compara a produtividade local com a produtividade em rede. Apesar disso, a segunda é mais favorável para o caso 5por duas razões. A primeira é a proximidade do mercado de dispositivos eletrônicos e a segunda é o aproveitamento do fuso horário para testes de integração e desenvolvimento de componentes.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Os artefatos desenvolvidos em rede são basicamente código. Entretanto, há possibilidade de se desenvolver a parte de negócios de maneira distribuída. A análise dos riscos e da complexidade de algumas variáveis já identificadas por um site é

utilizada em uma base global para determinar o risco de um projeto. Exemplo: um novo processador que será testado por uma equipe pouco experiente.

Os requisitos de um projeto podem ser negociados em rede, porém. Nem sempre o departamento de marketing envia o projeto para o site de produção local., porém são negociados prazos de entrega e a classificação dos requisitos (quais são críticos, quais necessários, quais poderão ser implementados futuramente). Após a negociação, os requisitos são refinados e enviados para produção.

A documentação de teste é considerada fundamental para o projeto. O teste é efetuado em várias etapas do projeto, porém o teste final, na qual o software é embarcado ao hardware é considerado extremamente importante. Após este teste, se aprovado o produto é colocado em produção e comercializado.

As métricas de produtividade são globais e utilizadas em todos os sites. Não há comparativo entre produção local e produção em rede, entretanto a organização justifica a produção em rede, porque atende mercados locais e reduz o custo.

A organização utiliza um conjunto padronizado de sistemas operacionais para dispositivos embarcados para facilitar e evitar o desenvolvimento a partir do zero.

Há casos em que os artefatos apresentam erros. Neste trabalho são consideradas apenas duas situações: erros oriundos da integração, ou seja, quando não foi possível integrar os componentes e erros de interações não previstas após integração. Em ambos os casos, os componentes retornam para a fase de projeto, e o ciclo de produção é repetido.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

Toda a comunicação é efetuada via Internet. A comunicação é restrita e poucos funcionários têm acesso externamente a processos e produtos em estágio de produção.

O acesso a mídia de comunicações (internet, gravadora de CD, portas USB para gravação de arquivos), de maneira geral é restrita para a maioria dos colaboradores.

Parte do histórico de intercomunicação assíncrona é armazenada em servidores. O objetivo é garantir um mecanismo de auditoria para controle de patentes. O histórico não é utilizado para auxiliar o processo de desenvolvimento distribuído de software. Arquitetos, programadores e outros colaboradores do desenvolvimento elaboram planos de treinamento e capacitação a partir da experiência.

5.5.1 Aderência ao modelo preliminar

A colaboração deste caso abrangeu vários aspectos do modelo. O primeiro aspecto foi o processo inicial de desenvolvimento distribuído de software, pois, inicialmente, alguns colaboradores foram treinados nas unidades mais experientes e implantaram a unidade fabril nacional.

A integração e o teste de integração foram revelados explicitamente neste caso. Comumente, alguns projetos utilizam uma nova plataforma de desenvolvimento pouco testada. Além disso, a grande complexidade de cada projeto favorece o surgimento de uma interação não prevista na etapa de arquitetura. Se isso ocorrer, os componentes precisam ser reavaliados, realimentando o ciclo de desenvolvimento.

A fase de teste de integração agrega o hardware e o software em um produto que será comercializado para o usuário final. Durante o teste, podem ser identificados erros não previstos. Se isso ocorrer, o ciclo de desenvolvimento novamente se repete a partir do processo de arquitetura. Por isso, a organização considera críticas para o sucesso do projeto a etapa de integração e a de teste etapas.

A realocação de tarefas quase não ocorre, entretanto os sites possuem parcerias com organizações externas para manter sua estrutura de produtividade de software escalável.

5.5.2 Aspectos peculiares

O departamento de marketing da organização considera o sigilo de informações sobre novos produtos crucial para o sucesso. Por isso, é comum os

visitantes serem revistados antes de entrarem nos departamentos de produção da empresa.

A maioria das entrevistas principais, nas quais o protocolo de pesquisa foi aplicado, foram gravadas. Neste caso, a autorização para conduzir uma gravação foi difícil.

5.6 Caso 6

Primeiro Bloco: Identificação da organização

O que detectar:	Concepção organizacional da empresa.
Pontos de análise:	Identificação da empresa (nome/fantasia); Identificação do contato primário para a pesquisa (nome e cargo); Missão da empresa; Principal atividade de negócio; Atividades secundárias de negócio; e Organograma.

O caso 6 foi aplicado em organização que desenvolve software para entidade financeira. Embora atualmente a organização de desenvolvimento de software seja autônoma, desenvolve projetos de software somente para essa entidade financeira.

A demanda gerada pela instituição financeira fez com que várias unidades de desenvolvimento fossem estabelecidas. A instituição financeira cresceu principalmente em função de aquisições de outras instituições, por isso, enfrenta processos de padronizações entre as unidades financeiras adquiridas e a regionalidade de cada uma das unidades.

A figura 44 ilustra como o processo de padronização ocorre no cliente do caso 6. A padronização dos processos de negócios é estabelecida pela instituição financeira global. As sedes locais, iniciadas a partir da instituição financeira, recebem a padronização automaticamente. Entretanto, quando há casos de aquisição, é preciso padronizar os processos da organização adquirida, desde a matriz até as sedes locais.

Todas as sedes locais (originadas ou adquiridas) têm suas particularidades regionais, oriundas da cultura e das leis de cada local. Além disso, há demanda muito grande de software, principalmente com relação aos sistemas legados em funcionamento nas instituições adquiridas. Por causa disso, além do custo de

produção, a organização de desenvolvimento de software analisada neste caso distribui-se em várias unidades pelo globo.

Figura 44- Organização do cliente do caso 6

Com relação à organização de desenvolvimento de software, a matriz do cliente está localizada na China. Por isso, os maiores projetos são iniciados lá e distribuídos pelos demais sites de produção de software.

Não há empecilho para um site iniciar um projeto. Geralmente, os sites concentram-se em desenvolver projetos de software globais, para atender toda a organização financeira, mas também realizam projetos das sedes locais, que também podem ser desenvolvidos de maneira global.

Segundo Bloco: Identificação dos projetos distribuídos

O que detectar:	Perfil dos projetos desenvolvidos em ambientes de DDS. Detectar projetos desenvolvidos de maneira distribuída, na qual a organização tem participação.
Base teórica:	Prikladnicki, Audy e Evaristo (2003), Sa e Maslova (2002), Ebert e Neve (2001), Zanoni e Audy (2002), Bass, Kazman e Clements (2003), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificação do(s) projeto(s); Identificar normas/padrões empregadas; Identificação do ciclo de vida do(s) projeto(s): - Quais os objetivos do projeto; e Identificar a arquitetura do projeto.

A maior parte dos projetos são desenvolvidos para a área financeira. Basicamente há projetos para sistemas legados, desenvolvidos na Linguagem Cobol e

executados em Mainframe e projetos que integram Mainframe com baixa plataforma (PC), utilizando Java, .net, Visual Basic etc.

O site da organização analisada possui certificação CMMI nível 2. Os demais sites são certificados de maneira autônoma, como é o caso da Índia que possui certificado CMMI nível 4.

Embora utilizem *frameworks* de desenvolvimento global e processos globais, alguns projetos locais dependem de um arquiteto local para determinar a arquitetura da solução. Portanto, há certa flexibilidade quanto aos processos de arquitetura. No entanto, há um conjunto restrito de componentes de arquitetura que podem ser utilizados, restringindo o modelo da solução e mantendo certa padronização mundial.

Os projetos desenvolvidos pela organização utilizam geralmente dois tipos de arquitetos. Um, denominado como "arquiteto alto nível", é alocado próximo aos analistas de negócio e um, "baixo nível", desenvolve trabalho próximo aos desenvolvedores.

O time de arquitetos é dependente da solução. Se a solução envolver plataforma PC e Mainframe, serão necessários dois times. Cada time pode ser composto por várias pessoas, também distribuído globalmente.

Terceiro Bloco: Identificação da relação entre os sites

O que detectar:	Detectar o perfil dos sites envolvidos no projeto e a relação que há entre eles.
Base teórica:	Alstyne (1997), Barthelmess(2003), Damian e Zowghi (2002), Dutta e Roy (2005), Siqueira e Silva (2004), Mintzberg (2003)
Pontos de análise:	Quantidade de sites (por projeto); Identificar o agrupamento; Identificar as bases de agregação da rede; Identificar o tipo de relação tem os nodos: *offshore*, *nearshore* etc; Identificar a relação hierárquica dos participantes; Identificar o mecanismo de coordenação; e Verificar o mecanismo de coordenação em diversos níveis do projeto;

Há uma unidade centralizadora de negócios, porém o projeto é desenvolvido globalmente. Todos os sites de desenvolvimento possuem times especializados em todas as tecnologias. Cada site possui *pools* de recursos humanos utilizados em projetos locais ou distribuídos.

Há duas razões para a existência de sites distribuídos. A primeira é o aproveitamento do fuso horário para o desenvolvimento de software. Embora não haja desenvolvimento *follow-the-sun*, o fuso horário agiliza o desenvolvimento com relação à integração de artefatos e testes de integração.

A segunda é que há várias unidades financeiras dispersas no planeta, cada uma com suas particularidades pessoais e legais. Há necessidade também de atender unidades financeiras que foram adquiridas e ainda estão passando por processos de padronização com a unidade global.

As unidades externas foram criadas com o propósito de atuarem como sites de *offshore*, atendendo a matriz em Hong Kong e *nearshore* atendendo a unidades financeiras próximas.

Quarto Bloco: Identificação dos processos distribuídos

O que detectar:	Determinar a relação dos sites em termos de processo. Identificar como é a integração do produto e subprodutos e a solução eventuais problemas.
Base teórica:	Alstyne (1997), Battin et al.(2001), Carmel e Agarwall (2001), Chang, Zhang e Tiang (2001), Herbsleb e Grinter (1999), Prikladnicki e Audy (2002), L'Erario et al. (2004), Facin (1998)
Pontos de análise:	Identificar a homogeneidade dos processos dos sites: - Nível de homogeneidade: de tecnologia a ferramentas integradas; Identificar o processo de gerenciamento de configurações; Identificar o gerenciamento de fornecedores (terceiros); Avaliar a integração dos subprodutos, envolvendo processos e ambientes de integração; Identificar o uso de modelos CMMI / PMBOK; e Identificar ferramentas integradas de produção;

Há padronização intensa dos processos de software. Todos os processos de trabalho realizados pelo desenvolvedor são documentados e frequentemente quem atua no desenvolvimento recebe treinamento com o objetivo de homogeneizar e difundir processos estabelecidos.

Os *frameworks* de desenvolvimento também são padronizados. Dessa maneira, a organização garante que a arquitetura seja globalmente homogênea, porém flexível para atender a projetos locais e/ou específicos.

Há uma equipe responsável em manter o processo global de desenvolvimento. Por isso, há métricas globais, ou seja, que todos os sites devem utilizar, mas, os sites podem conter métricas especificas.

Todo o processo de gerenciamento de configurações é padronizado de acordo com o projeto. O uso de ferramentas de controle de versões é intenso a fim de manter sempre atualizado o repositório dos projetos.

Quinto Bloco: Identificação do distribuidor de tarefas

O que detectar:	Identificar o distribuidor de tarefas. A exploração que esta seção aborda concentra-se sobre o papel de distribuidor de tarefas, suas atividades, problemas e escopo.
Base teórica:	Augustin, Bressler e Smith (2002), Becker et al. (2001), Carmel e Agarwall (2001), Herbsleb e Grinter (2003)
Pontos de análise:	Identificar os distribuidores de tarefas; Identificar se o distribuidor de tarefas é o expertise; Identificar como as tarefas são distribuídas; Identificar como as tarefas são rearranjadas em caso de necessidade; Identificar os fatores que conduzem o distribuidor a rearranjar tarefas; Identificar parâmetros de produtividade empregados no processo de distribuição; e Identificar o que o distribuidor de tarefas precisa conhecer (em termos de processo, tecnologia, etc.) dos demais sites para distribuir a tarefa.

Os arquitetos que atuam próximos aos analistas de negócio, denominados como arquitetura alto nível são centralizados.

As tarefas não são rearranjadas. Quando um site fica ocioso, o gerente e/ou líder de desenvolvimento notifica o cliente imediatamente. Não há métricas entre o desenvolvimento centralizado e distribuído, entretanto, justifica-se o desenvolvimento distribuído pela quantidade de clientes dispersos.

A figura 44 indica o funcionamento do fluxo de trabalho do caso 6. Até o levantamento de requisitos, o trabalho é centralizado na unidade de negócio. A partir da arquitetura alto nível o trabalho passa a ser distribuído.

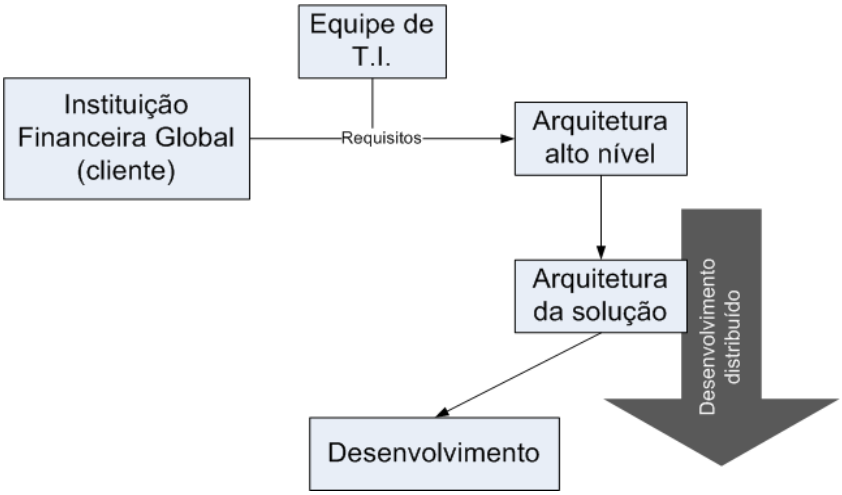

Figura 45 - Fluxo de trabalho do caso 6

Os arquitetos de alto nível negociam com os arquitetos de baixo nível a fim de decidirem como serão distribuídas as tarefas.

Sexto Bloco: Identificação dos artefatos

O que detectar:	Esta seção aborda os artefatos gerados pelos nós da rede, visa identificar os elementos trocados entre os sites, e como em um ambiente de groupware, esses elementos são utilizados do ponto de vista de produção da rede.
Base teórica:	Antunes (2005), Augustin e Bressler (2002), Chang, Zhang e Jiang (2001), Borghoff e Schlichter (2000), Grinter (1995), Clerc, Lago e Van Vliet (2007)
Pontos de análise:	Identificar artefatos que foram desenvolvidos em rede (plano de desenvolvimento de software, caso de negócio, plano de iteração, métricas, riscos, etc.); Identificar como foi o processo de interação sobre cada artefato; Identificar quais os principais conflitos gerados pelas interações; Identificar se os artefatos produzidos em redes tiveram custo/tempo menor de produção comparado a parâmetros locais; Identificar as dificuldades encontradas em se adequar artefatos que tiveram problemas; Identificar como são tratados artefatos não padronizados; e Identificar se o tratamento a artefatos não padronizados demandou um tempo considerável.

Os documentos de requisitos são levantados pela unidade financeira cliente.

As integrações iniciais de um projeto não são triviais. Entretanto, os projetos nos projetos mais maduros, os processos e ferramentas de integração são utilizados trivialmente por desenvolvedores mais experientes.

Problemas originados na integração geram módulos postergados de integração e desenvolvimento. Se o problema não for crítico, o módulo é entregue mesmo com falhas.

Para erros não ocorrerem na integração, os testes de integração são intensos. Em alguns casos chegam a custar até 2/3 do tempo de desenvolvimento de um determinado requisito.

Sétimo Bloco: Comunicação inter-site

O que detectar:	Esta seção analisa a comunicação empregada na rede. Estuda como o distribuidor se comunica com os demais nodos da rede
Base teórica:	Antunes (2005), Augustin, Bressler e Smith (2002), Barthelmess(2003), Battin et al (2001), Borghoff e Schlichter (2000), Defranco-Tommarello e Deek (2002)
Pontos de análise:	Identificar o tipo de mídia de comunicação empregada; Identificar o objetivo específico de cada mídia dentro das interações; Detectar se foi armazenado um histórico da comunicação; Detectar se o histórico armazenado serviu como base para os novos sites; e Verificar qual parte do projeto foi desenvolvida presencialmente;

Há um histórico de comunicação síncrona armazenada. A organização é cautelosa com relação ao acesso a dados e código fonte. Os computadores não têm unidade de CD, acesso a portas USB e outros recursos que possibilitariam ao funcionário a apropriar-se de algum código e/ou informação.

5.6.1 Aderência ao modelo preliminar

Neste caso foi detectado que há um ciclo de maturidade durante a integração de componentes. No início do projeto, foi detectado que os processos de integração, mesmo que especificados, não são seguidos com trivialidade pelos desenvolvedores. No entanto, com a evolução do projeto, os desenvolvedores passam a integrar os componentes de forma mais trivial.

5.6.2 Aspectos peculiares

Este caso tem uma semelhança com o caso 2. Os processos de desenvolvimento de software, distribuição de tarefas, gerenciamento de projetos foi praticamente igual ao apresentado pelo caso 2. A diferença é que o caso 6 tem um único cliente.

Foi notada também a questão da exigência no sigilo de informações e acesso aos meios de comunicação dentro do site analisado. Os computadores, salvo alguns, não tinham acesso à Internet (exceto e-mail da empresa), nem acesso a dispositivos de armazenamento USB ou qualquer mídia armazenável (disquetes, CDs etc).

5.7 Análise dos resultados

Os casos foram analisados individualmente, com várias realimentações e validações dos dados obtidos de maneira cíclica. Para este trabalho, foi apresentado a versão final dos casos.

A tabela 7 resume as contribuições de cada caso. A segunda coluna (Contribuição – M3DS) representa quais estados foram adicionados e/ou refinados no modelo final a partir das informações obtidas nos casos. A terceira coluna da tabela 7 representa que nível de padronização – de acordo com os conceitos apresentados na seção 6.3 - que foi empregado pelo caso. A quarta coluna identifica quais entidades responsáveis em distribuir as tarefas na rede.

Tabela 7 - Resultado da análise dos casos

Caso	Contribuição (M3DS)	Padrões	Distribuidor de tarefas
1	P0, P1	Tecnologias, interfaces	Comitê
2	P1, P8, P9	Ferramentas integradas	Gerente de negócios, projetos e líderes de desenvolvimento
3	P1, P8, P9	Processos de trabalho	Coordenador de desenvolvimento
4	P1, P9	Ferramentas integradas	Arquiteto, líderes/gerentes de projetos
5	P1, P2, P5, P4	Ferramentas integradas	Arquitetos, gerente de projetos
6	P1, P9	Ferramentas integradas	Gerente de projetos

5.7.1 As proposições da tese

No capítulo 3 foram apresentadas algumas proposições. Durante a aplicação do protocolo de pesquisa em cada um dos casos, as proposições foram discutidas e avaliadas. Elas nortearam a pesquisa, fazendo parte do protocolo aplicado em campo.

P1: A heterogeneidade dos processos e/ou dos artefatos torna o desenvolvimento mais complexo.

A heterogeneidade dos processos torna e/ou dos artefatos tornam o desenvolvimento mais complexo. Por isso, foram constatados no caso 1 muitos impasses oriundos da heterogeneidade dos sites.

Além disso, há outra variável embutida nesta proposição. Não somente os sites e/ou artefatos precisam ser homogêneos, mas também há necessidade de conhecimento do contexto do projeto que deve ser compartilhado por todos os desenvolvedores – codificadores, arquitetos etc. Percebe-se isso no caso 6, no qual, foi constatado que as primeiras integrações de um projeto são mais difíceis, mas, as demais tornam-se triviais para os mesmos desenvolvedores. Se há, no entanto, mudança de desenvolvedores, as integrações tendem a ser mais trabalhosas, já que dependem de conhecimento prévio.

Além disso, não somente um ambiente comum de desenvolvimento/integração torna ágil o DDS, mas também os processos empregados nestas ferramentas. Todo elemento que constitui o contexto relacionado ao projeto exerce diretamente influência sobre a produção distribuída.

Como o projeto para o desenvolvimento distribuído de software é sensível ao contexto, cada mudança precisa de disseminação com o objetivo de homogeneizar os sites da rede e reduzir ambiguidades oriundas de interpretações diferentes sobre um mesmo artefato. Diversos trabalhos como Pasala et al. (2008) e Chaves et al(2008) abordam a disseminação de informações sobre uma rede de produção distribuída de software.

O projeto analisado no caso 6 demonstrou que os procedimentos de integração tornam-se mais naturais à medida que o projeto evolui. Nesse caso, no inicio do projeto, os processos de integração eram demorados e desgastantes, perdia-se tempo em função da falta de prática dos desenvolvedores. Com a evolução do projeto, o tempo de realização dos processos reduziu, isso já que os desenvolvedores se habituaram a eles.

P2: O impacto cultural compromete o desenvolvimento.

Os casos 2, 5 e 6 ofereceram indicativos que esta proposição é verdadeira. Nesses casos, havia desenvolvedores de diferentes nacionalidades em diferentes países. Houve relatos de problemas relacionados a feriados (calendários diferentes), religião e idioma. O sotaque do idioma nativo (região onde está alocado o site) foi

severamente criticado nos casos 2 e 6. Mesmo utilizando uma linguagem padrão, o inglês, os dois casos citaram vários problemas decorrentes da baixa qualidade da pronuncia.

Em um âmbito geral, ambos os casos relataram que houve dificuldade de interpretação em videoconferências e/ou audioconferências, justificando a necessidade de troca de mensagens textuais síncronas. Nacionalmente, os dois casos relataram a dificuldade de compreender o idioma inglês quando proferido por um chinês ou um indiano. Apesar disso, os casos, afirmaram que a padronização de artefatos minimiza as ambiguidades oriundas de interpretação divergente.

P3: Artefatos não padronizados podem gerar impasses.

A padronização de um contexto como um todo – artefatos produzidos, ferramentas de integração/desenvolvimento, processos de trabalho – cria um conjunto comum de conhecimento usado entre os sites. A padronização do contexto, agindo como um mecanismo de comunicação e coordenação, por ajuste mútuo ou supervisão direta, reduz drasticamente a quantidade de impasses.

No caso 1 houve pouca padronização, limitando-se somente à tecnologia. O resultado foi um conjunto de produtos de trabalho que não puderam ser integrados. Nesse caso, a seta 10 (figura 46) do M3DS foi acionada por várias vezes durante a execução do projeto, ou seja, foi durante a integração que se percebeu a necessidade de reprojetar módulos ou até mesmo redesenvolvê-los para que pudessem ser integrados.

O caso 5 revelou interface de padronização entre o cliente e a organização que desenvolve software. A interface é estipulada quando há desenvolvimento de software dentro de um cliente, seguindo os padrões do cliente. A organização do caso 5 considera a implantação desta interface mais viável que aceitar o ingresso de artefatos não padronizados em sua rede de produção.

O caso 6 aborda com certa naturalidade artefatos não padronizados originados somente da codificação de novos projetos. Esse caso considera a possibilidade de erros durante a integração e consequentemente reprojeto. Tais erros são oriundos de plataformas, trechos de código e arquiteturas de solução utilizadas pela primeira vez. Após seu uso e consolidação, esses produtos de trabalho são incorporados ao know-how, fazendo parte da padronização. Todos os demais produtos de trabalho gerados são padronizados.

P4: O histórico da comunicação síncrona/assíncrona pode auxiliar novos sites.

O histórico da comunicação não foi empregado por nenhum dos casos para auxiliar sites nodos a ingressarem na rede. Exceto no caso 1, os demais casos armazenam histórico pelo menos da comunicação assíncrona. Entretanto eles afirmam que é inviável utilizar o montante de mensagens para ajudar novos sites. Os históricos ficam armazenados com o objetivo de auxiliar auditoria, caso seja necessária.

Foi citado no caso 2 que a inspeção periódica dos registros de comunicação ajuda a determinar os treinamentos necessários. Entretanto, essa inspeção não é feita de maneira sistemática, ou seja, o gerente apenas lia habitualmente as mensagens, que não eram poucas.

O caso 6 foi severo quando questionado sobre histórico de comunicação. Segundo o entrevistado, seria necessário todo o tempo do projeto para que o histórico fosse analisado e, por isso, era utilizado apenas para fins de auditoria.

P5: O distribuidor de tarefas é composto por vários níveis, de acordo com o grau de coordenação nele exercido e conforme a granularidade do repasse que deve ser entregue.

Em um projeto de software, o distribuidor de tarefas pode atuar a partir do projeto/design, repassando segmentos inteiros de projetos ou somente tarefas de codificação. São mais comuns os casos em que há uma hierarquia comercial definida e

os sites atuam como entidades codificadoras do que casos em que podem gerar requisitos esporadicamente.

Há casos em que vários sites geram requisitos para um determinado software. Isso pode ocorrer quando um software será executado em âmbito global, por exemplo, e, por causa disso, precisa conter especificidades de cada país em que será utilizado.

Dificilmente as tarefas são rearranjadas pelo distribuidor de tarefas, porque o tempo de configuração de um colaborador entre um projeto e outro representa um custo alto. Este fato foi verificado nos casos 2, 3, 4, 5 e 6.

Podem existir vários tipos de distribuidores de tarefas. Geralmente são divididos por áreas funcionais do projeto ou por áreas de proximidade com o cliente.

No caso 5, o desenvolvimento é distribuído de acordo com a especialidade de cada grupo em lidar com determinada tecnologia.

Já no caso 6, a distribuição de tarefas inicia-se em função da área de negócios em que cada site atua.

Nos casos 2 e 4, a distribuição pode ser efetuada tanto por tecnologia, quanto pela área de negócios em que o site atua.

P6: O distribuidor de tarefas adota de acordo com a necessidade do projeto uma série de parâmetros para distribuir atividades, informal ou formal, que indica a produtividade da rede.

Os parâmetros adotados pelos distribuidores de tarefas são métricas globais, que todos os sites utilizam ou métricas locais, que apenas um site local utiliza. No entanto, nem sempre as métricas utilizadas na produção de software são suficientes para que as tarefas sejam alocadas.

No caso 6, por exemplo, o gerente/arquiteto próximo ao cliente contata os demais sites, averiguando a possibilidade de desenvolver o produto e distribuir as tarefas.

Nos demais casos, o procedimento foi similar. Sempre há um contato presencial e/ou por meio de áudio/videoconferência ou e-mail entre gerentes,

arquitetos e líderes de desenvolvimento antes de as tarefas serem alocadas em produção. Cada líder conhece a capacidade produtiva de seu time. Por isso, é elemento crítico na tomada de decisões de quem e como o projeto será desenvolvido.

Métricas obtidas em projetos anteriores também são utilizadas como parâmetros de distribuição. O caso 2 em particular, permitia ao desenvolvedor determinar sua medida de produção de acordo com um requisito. Esta medida era comparada há uma base existente e agregada aos desenvolvedores da equipe. Com isso mesmo os líderes não ligados diretamente com o time conseguiam determinar sua capacidade produtiva.

CAPÍTULO 6 – O MODELO DE DINÂMICA DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Este capítulo apresenta o Modelo de Dinâmica de Desenvolvimento Distribuído de Software, chamado de M3DS. O modelo baseia-se no modelo preliminar, apresentado no capítulo 4 e, que após a aplicação do protocolo de pesquisa, foi modificado, aproximando-se da dinâmica de desenvolvimento distribuído de software.

O M3DS é apresentado de duas maneiras neste capítulo. Na primeira, o modelo é apresentado no formato de máquina de estados. Entretanto não são seguidos os formalismos definidos por Hopcroft e Ullman (1979). O diagrama representa o fluxo de dados e a mudança de estados.

Na segunda, o M3DS é apresentado na forma de redes de Petri. Nesse caso, há formalismo matemático e possibilidade de visualizar a simulação da produção em um simulador de redes de Petri. Após apresentação do modelo, a seção 6.2 detalha cada um dos estados identificados.

6.1 Definição do M3DS

A dinâmica de um ambiente de DDS revela o funcionamento da rede, de sua concepção inicial até seu encerramento. Porém, abordar individualmente os sites na rede pode não revelar a influência do projeto tem sobre sua existência na rede.

Portanto, a dinâmica da rede, representada na figura 46, aborda conjuntamente dois componentes fundamentais em um ambiente de DDS. O primeiro, indica a dinâmica de uma unidade de produção de software. O segundo, é atrelado ao projeto desenvolvido em uma rede de produção de software. Sobre essa dinâmica, há uma intersecção, que ocorre quando o site precisa desenvolver um subproduto de rede, e uma instância do projeto para vários sites.

Na definição do M3DS por máquina de estados, as setas regulares representam as transações entre os estados do projeto enquanto as setas tracejadas representam as transações entre os estados do site em uma rede de produção de

software. Para que uma transição seja disparada, as tarefas atribuídas ao estado anterior precisam ser finalizadas com ou sem sucesso. Por exemplo, se as tarefas alocadas em P4 – Em integração – forem concluídas com sucesso, a transição 8 é disparada, almejando P4. Caso estas sejam concluídas com falha, a transição 10 é disparada e o estado P1 é realimentado.

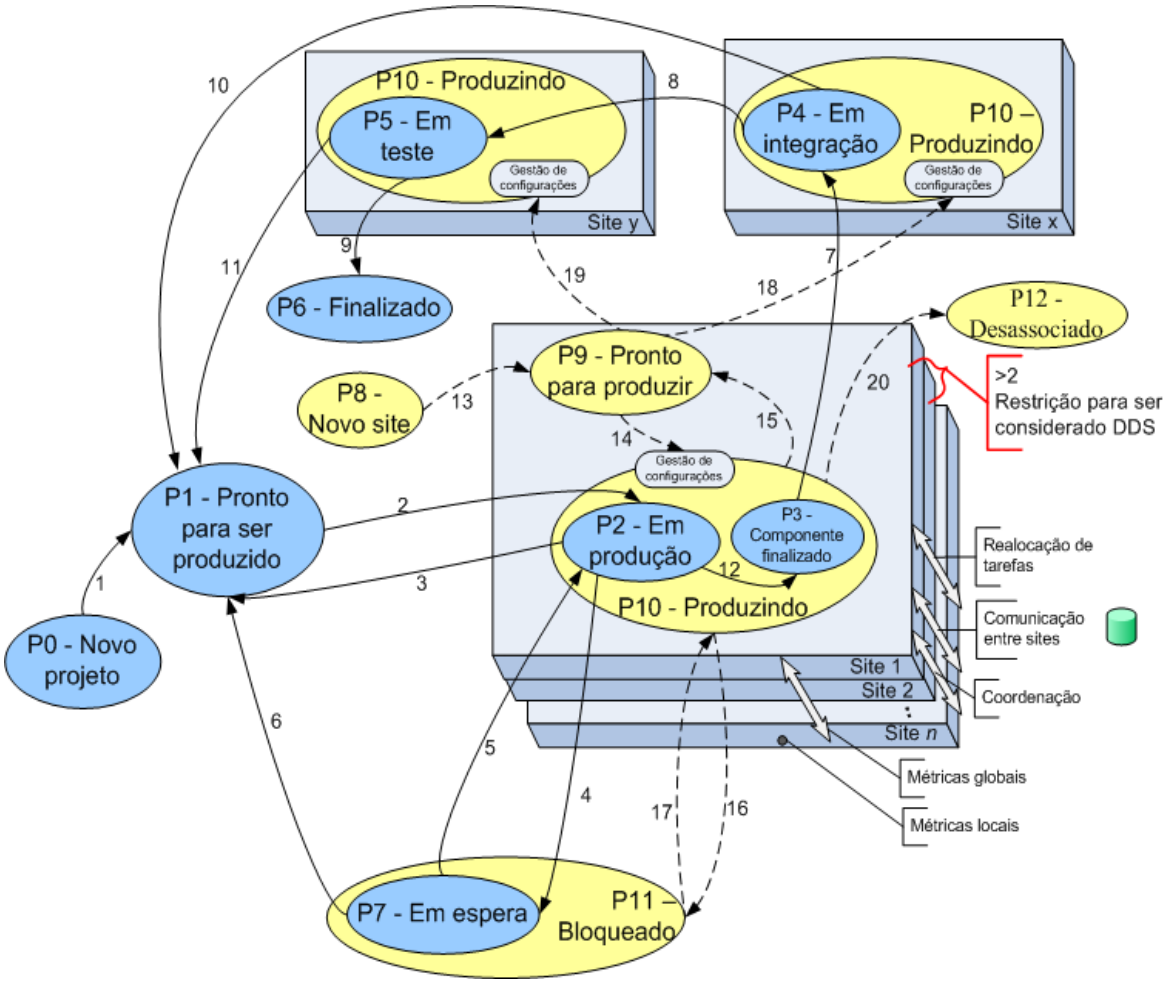

Figura 46 - M3DS

O modelo apresentado em máquina de estados foi mapeado em redes de Petri. A ferramenta utilizada para modelar foi a PIPE2, descrita por Bonet et al. (2007).

A segunda representação é mais formal. Foram necessárias quatro etapas para o mapeamento.

Na primeira, mapearam-se transições (setas) da figura 46 em uma tabela. A tabela 8 contém a relação entre as transições e os estados.

Para cada intersecção entre o estado (P) e a transição (T), foi marcada a letra "E" quando a transição T conduz ao estado P. A letra "S" foi marcada quando a transição T conduz a mudança de P para outro estado. As identificações das transições seguem o mesmo padrão das setas da figura 46. A seta cuja, identificação na figura 46 é 1, na rede de Petri tem o nome T1, por exemplo.

Tabela 8 - Relação entre estados e transições do M3DS

Estados / Transições	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
T1	S	E											
T2		S	E										
T3		E	S										
T4			S					E					
T5			E					S					
T6		E						S					
T7				S	E								
T8					S	E							
T9						S	E						
T10		E			S								
T11		E				S							
T12			S	E									
T13									S	E			
T14										S	E		
T15										E	S		
T16											S	E	
T17											E	S	
T18										S	E		
T19										S	E		
T20											S		E

A segunda etapa de mapeamento foi transformar a tabela 8 em uma rede de Petri. A relação das transições foi seguida fidedignamente, originando duas redes de Petri (figura 47 e figura 48). A figura 47 representa as transições do modelo somente pelo ponto de vista do projeto.

A figura 48 representa o conjunto de transições do modelo referentes ao site. Mesmo que ainda independentes, as duas redes de Petri possibilitam simular a produção de um ambiente de DDS.

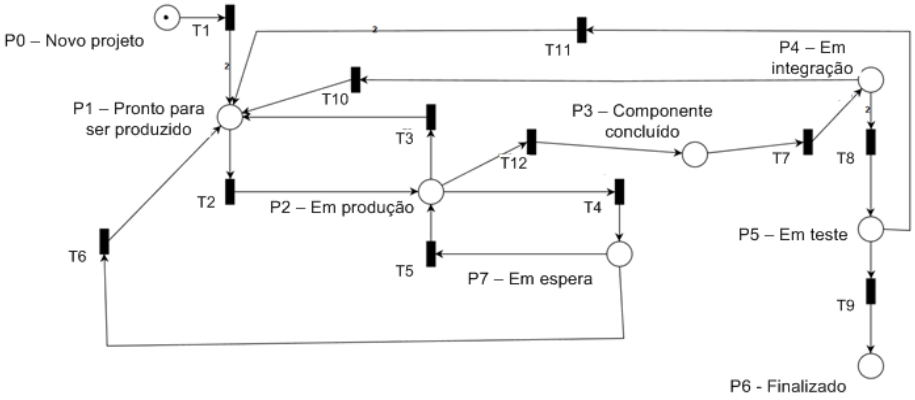

Figura 47 - M3DS – Ponto de vista do projeto

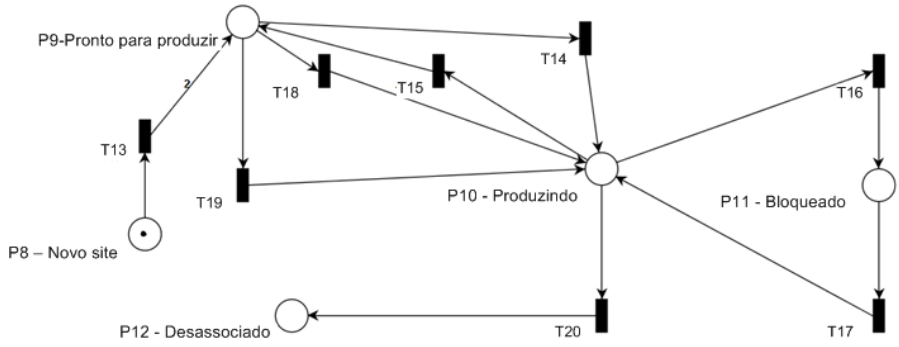

Figura 48 - M3DS - Ponto de vista do site

A terceira etapa consiste em determinar quais transições eram dependentes e, consequentemente, precisavam ser executadas de modo simultâneo. O resultado foi a união de algumas transições. A tabela 9 representa a dependência.

Para cada linha da tabela, foram mapeadas quais transições exigem simultaneidade e qual o conjunto de entradas e saídas dessas transições. Por exemplo, as transições 2 e 14 (T2,14) são executadas paralelamente. Quando o conjunto de transições é disparado, os estados mudam de P1, P9 para P2, P10.

Tabela 9 - Mapeamento das dependências das transições

Estados / Transições	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
T2,14		S	E							S	E		
T5,17			E					S			E	S	
T8,19					S	E				S	E		
T7,18				S	E					S	E		
T4,16			S					E			S	E	

Na quarta e última etapa, a tabela 9 foi utilizada para conduzir a união entre a rede de Petri que representa as transições do projeto (figura 47) e a rede de Petri que representa as transições de estados dos sites (figura 48).

A figura 49 é uma representação final do M3DS em redes de Petri. Com esse modelo, é possível iniciar uma simulação. O objetivo da simulação é determinar quais condições são necessárias para que uma mudança de estado ocorra, como é o caso indicado por Chaves et al (2007). Esse modelo não tem o objetivo de revelar a capacidade de produção de uma rede de desenvolvimento de software. Por isso, foram utilizadas redes de Petri e não modelos de processos estocásticos.

O cenário que a figura 49 apresenta é uma organização com dois sites e um projeto de software. Quando executada, esta rede de Petri representa as condições para que um projeto de software seja desenvolvido de maneira distribuída por dois sites.

O modelo apresentado na figura 49 é uma instância específica do modelo apresentado na figura 46.

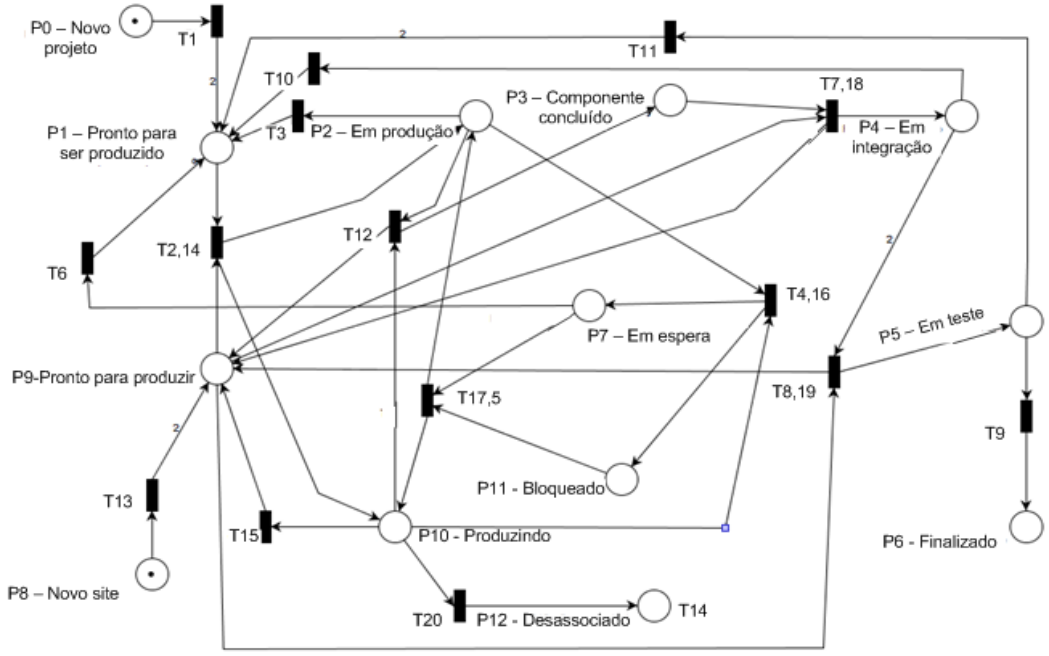

Figura 49 - M3DS em redes de Petri

Similarmente ao conceito de regras de encadeamento apresentado por Schoueri (1998), a figura 50 apresenta o fluxo normal de execução do M3DS em redes de Petri.

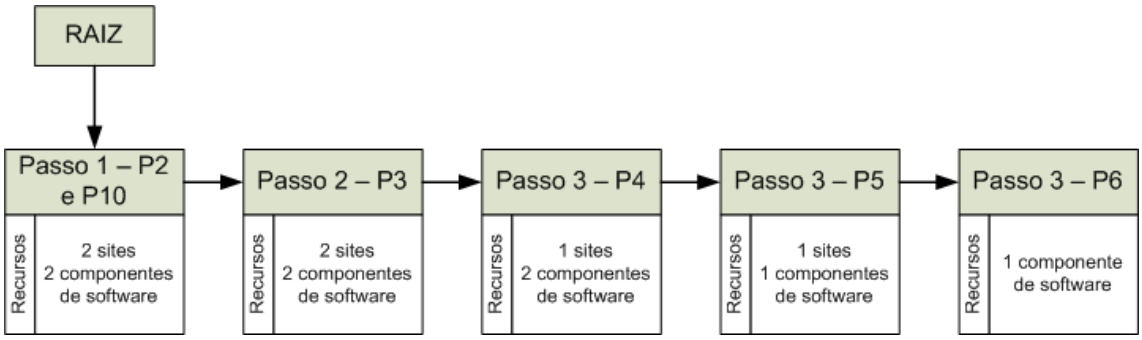

Figura 50 - M3DS fluxo normal da rede de Petri

6.2 Descrição do modelo

Esta seção explica todos os estados do modelo e identifica quais estados existentes e quais condições são necessárias para a mudança de estado.

6.2.1 Estado P0: Novo Projeto

Esse estado indica a concepção de um novo produto de software por alguma organização, em rede ou não. Nessa fase, além da concepção do projeto há a elaboração de uma visão compartilhada para que todos os nós envolvidos na produção possam compreender a razão do produto final.

Em alguns casos, o novo projeto inicia-se diretamente com a negociação entre alguns ou todos os sites envolvidos diretamente na produção do software. Nesse caso, inicialmente todos os sites conhecem o projeto de antemão e há uma interação entre eles. Essa situação é mais comum quando há parceria de mesmo nível hierárquico (hierarquia relacionada ao projeto) entre os sites e/ou há necessidade de que todos os sites tenham conhecimento sobre a capacidade produtiva de todas as demais unidades de desenvolvimento de software.

O mecanismo de coordenação empregado nessa situação é o de ajuste mútuo. O projeto é negociado de acordo com as características de produção informadas individualmente pelos sites. É comum esse cenário quando há desenvolvimento de software livre, por exemplo.

O processo está ilustrado na figura 51. O caso 1 é um caso típico desse modelo de projeto. No caso 1, a coordenação foi basicamente por ajuste mútuo e a granularidade do repasse foi de linhas de código a componentes.

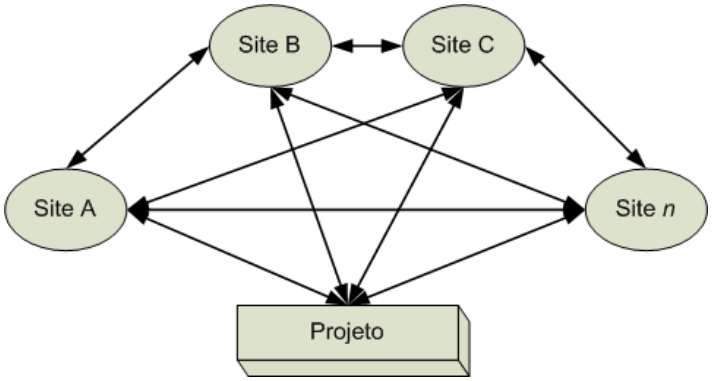

Figura 51 - Novo projeto: sites com mesmo nível hierárquico

O segundo caso ocorre quando um projeto é introduzido na rede por um site em particular. Nesse caso, há um site que conhece a capacidade produtiva dos demais e consegue distribuir posteriormente as atividades de desenvolvimento. O processo está ilustrado na figura 52. Os casos 3, 4, 5 e 6 desenvolvem comumente utilizando esse modelo.

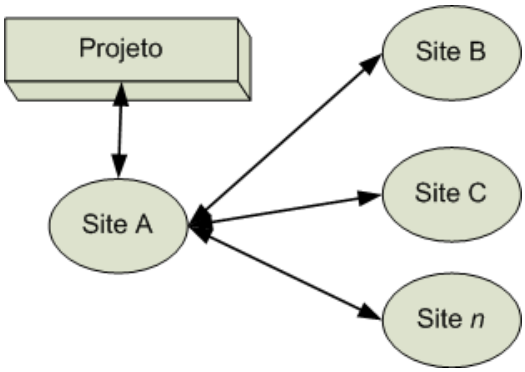

Figura 52 - Novo projeto: um site com maior hierarquia

Há também um modelo misto. O projeto é distribuído por vários sites a várias unidades de produção de software. É o que ocorre em alguns projetos no caso 2. O processo está ilustrado na figura 53.

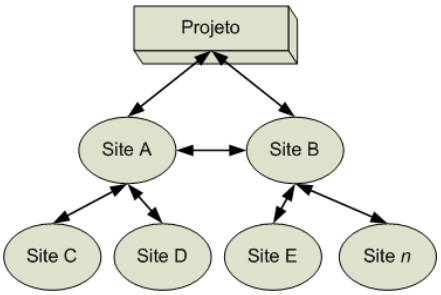

Figura 53 - Novo projeto: mais de um site coordena a distribuição do projeto

6.2.2 Estado P1: Pronto para ser produzido

Para alcançar o estado P1 é necessário que seja disparada a seta 1 (ou transição T1 na rede de Petri). Esse disparo Indica que o plano gerencial do projeto em termos de componentização e divisão de tarefas foi realizada. Após a seta, todos, ou parte dos artefatos, estarão prontos para serem disponibilizados e distribuídos para que a rede e os *sites* possam desenvolvê-los.

Esse estado indica que um papel de distribuidor de tarefas entra em ação para gerenciar a distribuição das tarefas do projeto na rede. Representa a segmentação e alocação das tarefas nos diversos sites de produção. Nesse estado, inicia-se o processo de ruptura.

O processo de ruptura, ilustrado pela figura 54, é definido como o processo no qual um projeto de software é preparado para ser desenvolvido em rede. Não somente há segmentação do produto final em módulos, mas também são definidos processos e ferramentas de integração.

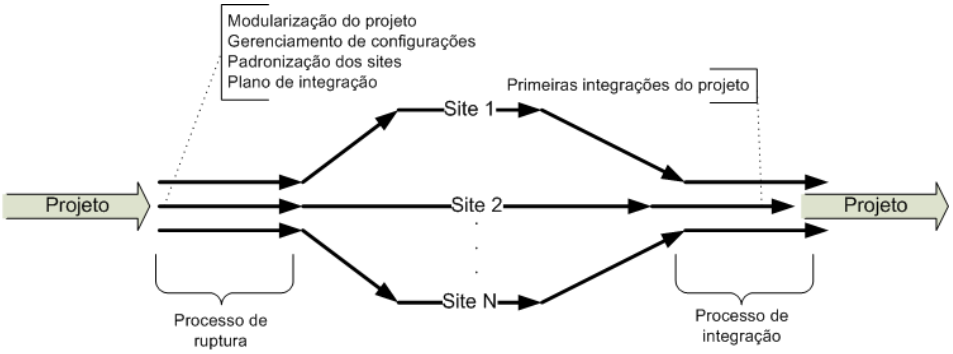

Figura 54 - Processo de ruptura do projeto

O estado P1 também pode ser alcançado caso as setas 3, 6 e/ou 11 sejam disparadas. A seta 3 é disparada quando a tarefa é rejeitada assim que entrar em produção. Quando uma dependência funcional é resolvida, o subproduto retorna ao estado de produção, executando a seta 6. A seta 11 é disparada quando um produto de trabalho sofreu alguma integração e está testado e pronto para realimentar o ciclo de produção.

6.2.3 Estado P2: Em produção

Esse estado representa uma tarefa que tem como resultado um subproduto de trabalho. Para almejar o estado P2, a seta 2 precisa ser disparada. A seta indica que o subproduto foi alocado em produção. Essa etapa implica enviar para o site a tarefa e os dados sobre ela. Podem ser enviados, por exemplo, códigos mais atualizados, scripts de banco de dados, requisitos do software etc.

A gestão de configurações é executada toda vez que um site inicia a produção. A produção do projeto foi dividida em três categorias: em produção, em integração e em teste. Todos esses casos caracterizam a produção do projeto, porém o escopo deste trabalho de pesquisa determinou que a integração e o teste do projeto constituem estados críticos para o DDS.

Esse estado também pode ser alcançado quando a seta 5 (T4) for disparada. A seta é disparada quando uma dependência funcional é resolvida, retornando então o produto de trabalho para a produção.

6.2.4 Estado P3: Componente finalizado

O componente é finalizado quando sua construção é concluída. Nesse caso, indica o fim do processo de codificação de um componente ou módulo de software. Esse estado sinaliza que o componente está apto a ser integrado com outro.

As operações de *check-out* são efetuadas após a conclusão do componente. A gestão de configurações está presente explicitamente nessa etapa.

6.2.5 Estado P4: Em integração

Após a codificação de dois ou mais componentes, o processo de integração é iniciado. Nesse caso, apenas um site é responsável por integrar dois ou mais componente, gerando outro produto de trabalho. A seta 7 indica o inicio deste estado.

Não há necessidade de somente um único site integrar todos os componentes de um projeto. Entretanto, somente um único site relacionado ao projeto pode integrar exclusivamente dois ou mais componentes do projeto gerando outro produto de trabalho. A figura 55 exemplifica o processo de integração efetuado entre quatro componentes e dois sites.

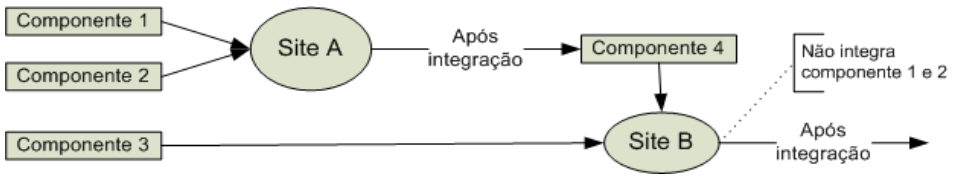

Figura 55 - Exemplo de um processo de integração

A integração deve ser estipulada no inicio do projeto, durante o processo de ruptura (figura 54). Se o processo de integração não for definido claramente, aumenta-se o risco de surgirem interações não previstas (disparando a seta 11). Isso fará com que os componentes retornem para o estado P1, e reinicie a fase de *design*.

6.2.6 Estado P5: Em teste

Quando a seta 8 é disparada inicia-se o procedimento de teste. Nesse caso o teste é da integração do software. Após essa etapa, o projeto pode ser finalizado, mudando seu estado para P6. O teste também pode gerar outro produto de trabalho, realimentando o ciclo por meio do estado P1.

6.2.7 Estado P6: Finalizado

O projeto é concluído. Esse estado significa que o projeto foi totalmente montado e concluído. Para alcançá-lo, a seta 9 precisa ser disparada.

6.2.8 Estado P7: Em espera

As dependências funcionais entre artefatos gerados pelos sites podem ocasionar bloqueio em dada tarefa. Esse estado representa o bloqueio da tarefa em relação a um projeto.

As setas 4 e 16 atuam paralelamente quando há dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subproduto fica bloqueado. Este estado ocorre juntamente com o estado P11.

6.2.9 Estado P8: Novo site

Indica que um novo nó da rede está pronto para associar-se a uma rede de produção de software. Para ser considerado um novo site, devem ser estipulados um mecanismo de coordenação, agrupamento e processo de software. Esse estado indica que o site está apto a trabalhar em equipe com os demais sites.

Um site pode ser criado a partir de um modelo já existente, importando processos e modelos de desenvolvimento. Os procedimentos de replicação são detalhados por Fabri et al.(2004), Fabri et al. (2007) e Trindade (2004).

6.2.10 Estado P9: Pronto para produzir

Para que o estado P9 seja alcançado é necessário que a transação indicada pela seta 13 seja disparada.

A seta 13 representa os procedimentos iniciais de produção, tratados, neste trabalho, como configuração do ambiente para o projeto.

Quando um projeto é iniciado, é necessária a configuração mínima de um ambiente de trabalho, que pode ser composto, por exemplo, de ferramentas de groupware, ferramentas CSCW, IDEs, sistemas de controle de versões, servidores e outros.

Essa transição implica configuração integrada entre os sites. Em outras palavras, a configuração desse ambiente não é de centralizada, pois depende do projeto e de todos os sites. As configurações são replicadas em vários casos nos vários sites.

A figura 56 indica o processo de configuração.

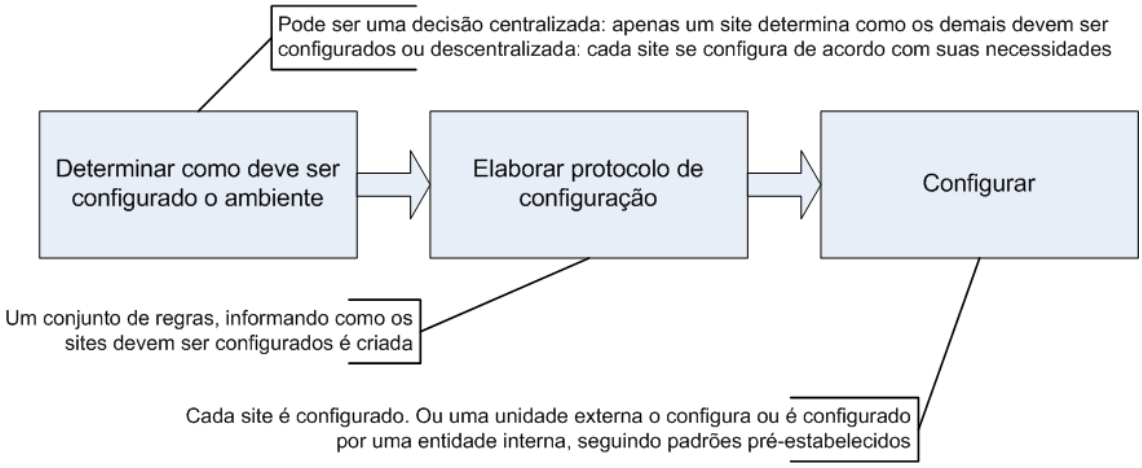

Figura 56 - Processo de configuração em um ambiente de DDS

Quando P9 é alcançado, o *site* já está na rede, ou seja, de alguma forma um acordo foi feito entre o *site* e a rede, que pode ser, por exemplo, o conceito de matriz e filial. O estado de *pronto* após a seta 13 indica que a organização submeteu-se a alguma análise, resultando essa em uma agregação à rede. Nesse estado, o site está aguardando tarefas.

Nessa etapa do processo, os desenvolvedores do projeto estão aptos para a produção em rede, ou seja, já foram treinados e estão familiarizados com a produção de software de maneira distribuída.

Para manter a produtividade, é necessário preservar o conhecimento entre os desenvolvedores da equipe. Trindade (2006) apresenta em seu trabalho como preservar o conhecimento em uma fábrica de software.

O estado pode ser alcançado também após o disparo da seta 15. Essa transição indica que o site fica ocioso eventualmente, aguardando nova tarefa. Isso ocorre, porque, quando um site termina sua tarefa no projeto, fica aguardando novas instruções, sem dependência funcional de outro site.

6.2.11 Estado P10: Produzindo

Esse estado significa que o site está executando uma tarefa e retornará, como resultado, um subproduto. Pode ser alcançado após o disparo da seta 14 (T3). Se isso ocorre, a produção inicia-se, ou seja, as atividades referentes ao desenvolvimento do produto que foram atribuídas ao site são executadas. A etapa ocorre paralelamente à Seta 2.

P10 inicia-se pelos procedimentos de gestão de configuração. Nesse processo inicial, os objetos são carregados/atualizados de forma que o site possa produzir sobre as últimas versões ou versões estáveis de um componente. Geralmente, quando é um procedimento de codificação, o desenvolvedor efetua operações de *check-out* para iniciar o trabalho e *check-in* para postar o resultado no repositório de versões.

As setas 17, 18 e 19 também conduzem ao estado P10. A seta 17 representa a resolução de dependência funcional e o retorno do nó na produção do artefato. As setas 18 e 19 representam a alocação de um único site para testar e integrar módulos de software.

6.2.12 Estado P11: Bloqueado

Podem ocorrer bloqueios na produção por dependências funcionais. Por exemplo, um site precisa de um produto de trabalho de outro site para prosseguir. Isso significa que o site está bloqueado para uma determinada tarefa.

Para tanto, as transições 4 e 16 ocorrem quando há uma dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subproduto fica bloqueado, pois não pode ser desenvolvido pelo fato de sofrer dependência funcional.

6.2.13 Estado P12: Desassociado

A ação indicada pela seta 20 indica a desassociação entre o site e rede. Ele é e deixa de fazer parte do ambiente de DDS. Essa transição ocorre quando o site deixa de atuar no projeto/grupo. Há, portanto, desassociação do site com relação aos demais sites e projeto.

6.3 Considerações finais

As redes de produção de software podem ser formadas pelos mais diversas motivos e meios. Além disso precisa ser considerado quando a constituição dos sites é *ad-hoc* ou vertical. Quando a constituição é *ad-hoc*, a agregação tende a ser temporária, ocorrendo por motivos de grande influência, isso faz com que sites heterogêneos, trabalhem em comum.

As propriedades dos ambientes de DDS, apresentadas na seção 4.2 mantiveram-se equivalentes no modelo final (M3DS).

A agregação vertical tende a ser mais durável. Ela se constitui, por exemplo, de uma fábrica e seus fornecedores. Nesse caso, há padronização mínima entres os sites, permitindo que o site principal receba subprodutos dos demais de forma a agregar, sem retrabalho, em seus produtos.

O nível de padronização determina quão bem as informações fluem e interagem naturalmente na rede. Em um ambiente de desenvolvimento distribuído de software, a padronização pode ser avaliada em níveis, como indica a figura 57.

Em um primeiro nível, a padronização pode ser abordada como a padronização da tecnologia, ou seja, os nodos adotam uma tecnologia como forma padronizada para desenvolvimento colaborativo. Essa forma de padronização foi adotada inicialmente pelos projetos ManWapp explicado por L'Erario et al.(2004) e no caso 1.

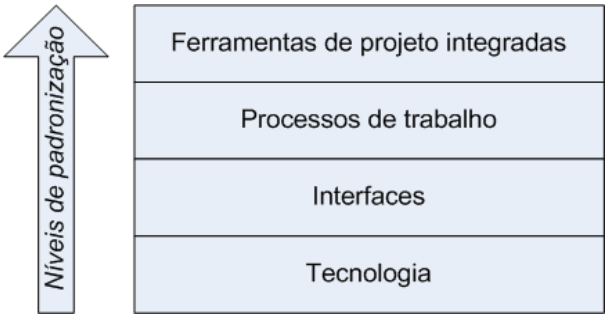

Figura 57 - Níveis de padronização

No segundo nível de padronização, as interfaces são adotadas como instrumento de padronização. Nesse caso, os nodos determinam como são as interfaces dos subprodutos e como acoplá-los.

Os processos de trabalho são padronizados quando os sites adotam o mesmo processo de trabalho. Como objetivo máximo de padronização, a adoção de ferramentas de projeto integrada auxilia e norteia as equipes a trabalharem cooperativamente.

De uma forma mais geral, Facin (1998), que explora a indústria automobilística em seu trabalho, afirma que o uso de ferramentas integradas auxilia o desenvolvimento de projeto.

Tal apoio é revelado em problemas que são detectados proativamente e processos que são testados computacionalmente antes de serem efetivamente realizados. Um exemplo disso é citado o *crash-test*. O uso de sistemas computacionais com recursos de comunicação permite numerosos testes por meio de simulação, antes que o *crash-test* seja efetivamente realizado.

Os casos 2, 3, 4, 5 e 6 utilizam ferramentas integradas de produção de software. Antes mesmo de entrar em produção, os gerentes têm uma visão sobre

como o produto será desenvolvido pela rede. Essa visão é norteada pelas ferramentas integradas de produção em vários níveis.

No âmbito da codificação, esses casos utilizam ferramentas que permitem ao desenvolvedor efetuar procedimentos de gestão de configurações e integração com certa automaticidade.

Em um plano mais gerencial, são utilizadas ferramentas que auxiliam a condução e definição do processo de software.

Há ainda uma divergência que deve ser considerada sobre dois fatores de padronização em ambientes de DDS. A tecnologia exige, de certa forma, um processo de trabalho adequado a ela. Dependendo da tecnologia adotada no produto final, o processo de trabalho deve ser modificado para comportar a nova tecnologia. Essa adequação pode ocorrer nas diversas atividades do desenvolvimento do produto final. Uma modificação mais drástica, no nível de arquitetura da solução, por exemplo, pode afetar todos os níveis de processo de trabalho.

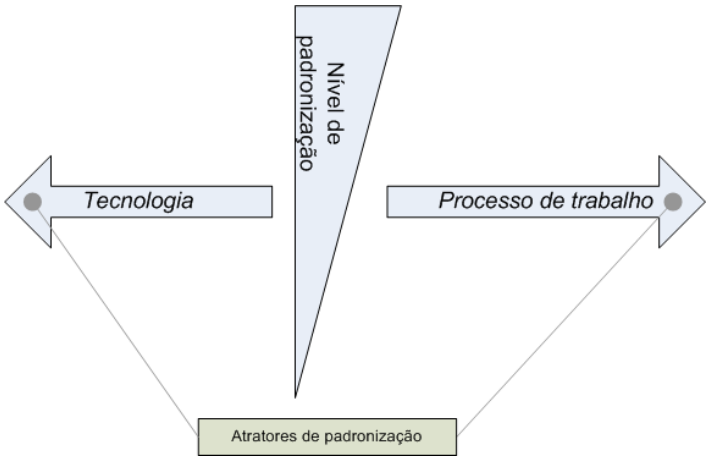

Figura 58 - Atratores de padronização

A figura 58 ilustra os atratores de padronização. Nela, os atratores possuem um significado similar aos apresentados por Milnor (1985). A tecnologia e o processo de trabalho são atratores de padronização que nem sempre atuam convergentemente. Por isso, se a tecnologia utilizada no desenvolvimento tem embutido um processo de trabalho divergente daquele que a organização já utiliza, é necessária uma força de trabalho a fim de compatibilizá-los.

CAPÍTULO 7 - CONCLUSÕES

As contribuições deste trabalho almejam construir um modelo de dinâmica de desenvolvimento distribuído de software. Para tanto, diversas teorias, como organização do trabalho (MINTZBERG, 2003), foram analisadas com o intuito de agregar valor à teoria de desenvolvimento distribuído de software. Assim, foi possível estabelecer uma posição mais precisa da rede de produção, mapeando-a em estados que podem ser analisados qualitativa e/ou quantitativamente. O modelo M3DS é um modelo geral de dinâmica. Seu objetivo é demonstrar o funcionamento do desenvolvimento distribuído.

Diversos estudos de caso foram feitos na área, como é o caso de Suzuki e Yamamoto (1999), Carmel e Argawall (2001) e Shami et al. (2004). Porém, não foi detectado nenhum estudo de caso de posicionamento da rede.

Para auxiliar a análise sistemática de múltiplas redes de produção, algumas variáveis foram estudadas e analisadas inicialmente em dois estudos de casos – L'Erario(2004) e Incubadora (2005). Estudo semelhante foi efetuado posteriormente em mais organizações.

O modelo aqui apresentado mostrou-se coerente a produção distribuída em todos os casos. As variações de produção entre um caso e outro são consequências da especificidade de cada organização e da configuração inicial da rede.

Nos casos 2 a 6, uma configuração rigorosa e especifica era aplicada sobre os projetos e os sites. Nesses casos, os projetos foram preparados para serem distribuídos. A elaboração da solução escolhida pelo arquiteto, junto com analistas de negócio, influencia altamente a distribuição de tarefas.

Uma configuração de ambiente é determinada junto com o projeto e a distribuição de tarefas. Foi detectados que os sites eram homogêneos porque todos seguiam os mesmos processos.

No caso 1, como não havia divisão sistemática do projeto e homogeneização dos sites, houve muitos impasses como dependência funcional e repúdio de tarefa.

A interação instantânea por meio de conferência, chat e VOIP entre os desenvolvedores, durante todo o projeto mostrou-se necessário em todos os casos. Embora os casos com processos mais sistemáticos, casos 2 a 6, tenham um processo de divisão de tarefas – processo de ruptura – formalizado, todos os desenvolvedores julgam necessária a interação entre os elementos dos times. Além disso, a coordenação, seja esta técnica ou gerencial, também se mostrou essencial por todo o ciclo de vida do produto.

Os casos apresentaram diferentes aspectos culturais que influenciaram na produção de software. Estes aspectos, que compreenderam a pronúncia de um mesmo idioma, o calendário referente a feriados, a religião e outros que compõem a cultura organizacional – explicado por Hofstede (2005) – diretamente impactaram na interação entre todos os indivíduos envolvidos em um projeto e foram acentuados pela distância física.

A coordenação foi apresentada neste trabalho sobre dois pontos de vista: técnico e gerencial. O primeiro incorpora os protocolos, as técnicas de interação entre os *stakeholders* de um mesmo projeto. O segundo é uma abordagem organizacional, incluindo a coordenação do trabalho. Embora sobre pontos de vista diferentes, em ambos, o objetivo geral da coordenação, nos casos, foi dividir um projeto em tarefas (unidade de realização de trabalho com o objetivo de evoluir o produto), alocá-las e gerenciá-las desde sua produção até integração.

No desenvolvimento distribuído de software alguns problemas que localmente são sutis tornam-se mais acentuados em função da distância física. Entretanto, estes problemas não representam um empecilho para que as organizações desenvolvam software de maneira distribuída. Sejam qual for o motivo e o modelo de negócios que a tornam viável – por aquisição, instalação de filial, parcerias, etc. – a distribuição da produção de software agrega muitos benefícios para a organização (apresentados no capítulo 2) e para a região na qual o site é instalado.

7.1 Trabalhos futuros

Algumas sugestões de trabalhos futuros foram detectadas durante a aplicação dos casos apresentados no capítulo 5. Tais sugestões não o constituem escopo deste trabalho de pesquisa.

A compreensão detalhada do processo de ruptura e integração foi atingida neste trabalho. Cada projeto analisado tem um processo de ruptura definido. Não faz parte do objetivo deste trabalho analisar e comparar os processos de ruptura e integração, entretanto, uma contribuição significativa como continuidade deste trabalho seria reestruturar o protocolo de pesquisa de forma que analise somente esses dois processos.

Durante os casos, também foi constatado que o conjunto de informações derivadas das comunicações, síncronas e assíncronas, é armazenado apenas para fins de auditoria. Outro viés de estudo, seria identificar se há possibilidade e como utilizar sistematicamente tal base de informações para treinar novos sites.

Os casos revelaram que, dificilmente, uma tarefa é rearranjada. Isso ocorre, porque o custo de configuração dos desenvolvedores entre projetos diferentes é alto. Entretanto, no caso 4, um projeto inteiro foi migrado de um site para outro. A migração foi viabilizada por fins econômicos, mas exigiu muitas horas/homem de trabalho e elevou drasticamente o custo do projeto. O trabalho de determinar como e quão oneroso é realocar tarefas também é uma possibilidade de estudo futuro.

Considerando a dinâmica do desenvolvimento distribuído de software apresentado neste trabalho, há também a necessidade de determinar como as informações e o conhecimento devem ser difundidos na rede. O cenário pode-se denominar propagação de informações sensíveis ao contexto e que não necessariamente se propagam no mesmo fluxo de dinâmica do desenvolvimento distribuído.

O desenvolvimento *follow-the-sun* foi detectado no caso 2. Entretanto, a granularidade de repasse foi de componentes. Segundo o caso 4, não há como desenvolver um projeto totalmente *follow-the-sun*, pois, em algum instante a granularidade do repasse pode reduzir de componente para linhas de código.

De acordo com o caso 4, quem receber as linhas de código terá o retrabalho de compreender o que está implementado para continuar o trabalho, despendendo uma quantidade significativa de tempo, por isso, considera esse tipo de desenvolvimento inviável. Uma contribuição futura seria analisar a dinâmica do desenvolvimento *follow-the-sun* e estudar as possíveis variáveis que influenciam esse cenário mais específico.

REFERÊNCIAS BIBLIOGRÁFICAS

ADACHI, T.; SHIH, L. C.;ENKAWA, T**. Strategy for supporting organization and strutucturing of development teams in concurrent engineering**. The International Journal of Human Factors in Manufacturing; v.4, n.2, p.101¬20, 1994.

ALSTYNE, Marshall Van. **The state of network organization: a survey in three frameworks**. in Journal of Organizational Computing & Electronic Commerce, 7(3); pp.83-151, 1997.

ANTUNES, P. Groupware: **Conceitos Fundamentais e Caracterização dos Principais Blocos Construtivos**. Relatório Técnico. Disponível em: <http://www.di.fc.ul.pt/tech-reports>. Acesso em: julho 2005.

AUDY, Jorge Luis Nicolas; PRIKLADNICKI, Rafael . **Desenvolvimento Distribuído de Software: Desenvolvimento de Software com Equipes Distribuídas**. 1. ed. Rio de Janeiro: Campus/Elsevier, 2007. v. 1. 211 p.

AUGUSTIN, Larry; BRESSLER, Dan; SMITH, Guy. **Accelerating Software Development Through collaboration**. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2002, Orlando. ICSE'02. Orlando: ACM, 2002. p. 559 - 563.

BARCUS A., MONTIBELLER, G. **Supporting the allocation of software development work in distributed teams with multi-criteria decision analysis**. OMEGA - The International Journal of Management Science, 2008.

BARNES, J., "**Implementing the IBM® Rational Unified Process® and Solutions**", IBM Press, 2008

BARTHELMESS, Paulo. **Collaboration and coordination in process-centered software development environments: a review of the literature**. In: INFORMATION AND SOFTWARE TECHNOLOGY, 8., 2003,. Information and Software Technology.: IEEE, 2003. p. 911 - 928.

BASS, Len; KAZMAN, Rick; CLEMENTS, Paul. **Software architecture in practice**. 2. ed. Sp: Addison Wesley, 2003.

BATTIN, Robert D. et al**. Leveraging resources in Global Software Development**. IEEE transactions on Software Engineering, v. 18, n. 2. Março/Abril 2001. p. 70-77.

BECKER, Simon et al. **A Delegation Based Model for Distributed Software Process Management**. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, 8., 2001, Tókio. Proceedings... Tókio: ACM, 2001. p. 130 - 144.

BERG, H. C. et al. **Formal methods of program verification and specification.** Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

BERTO, R. M. V. S. ; NAKANO, Davi Noboro . Metodologia da pesquisa e a engenharia de produção. **Revista Produção**, Porto Alegre / RS, v. 17, n. 2, 2001.

BONET P.,et al. PIPE v2.5: **A Petri Net Tool for Performance Modelling** (PDF format). Proc. 23rd Latin American Conference on Informatics (CLEI 2007), San Jose, Costa Rica, October 2007.

BORDEN, Alexander, NETT, Bernhard e WULF Volker. Coordination Practices in Distributed Software Development of Small Enterprises, icgse,pp.235-246, INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING (ICGSE 2007), 2007

BORGES, M. R., CAVALCANTI, M. C., MACHADO, M. L., **Suporte por Computador ao Trabalho Cooperativo**. XV Congresso da Sociedade Brasileira de Computação, Jornada de Atualização em Informática, Canela, 1995.

BORGHOFF, U., SCHLICHTER, J., **Computer-Supported Cooperative Work – Introduction to Distributed Applications**. 1. ed. Berlin: Springer, 2000.

BRANDÃO, C.R.**Repensando a pesquisa participante**. São Paulo, Editora Brasiliense, 1987.

BROWN, Alan W.; WALLNAU, Kurt C.. The Current State of CBSE. IEEE Software, v. 15, n. 5. setembro/outubro 1999. p. 37-46.

BURTON R.M.; OBEL B. **Strategic Organizational Diagnosis and Design**. Kluwer Academic Publishers, 1998.

CAMPBELL, Catherine Lowry; VAN DE WALLE, Bartel. Asynchronous Requirements Engineering: Enhancing Distributed Software Development. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: RESEARCH AND EDUCATION, 2003, Newark. **ITRE 2003**. Newark: ACM, 2003. p. 133 - 136.

CARDOSO, J.; VALETTE, R. **Redes de Petri.** Florianópolis: Editora da UFSC, 1997. 212p.

CARMEL, Erran; AGARWAL, Ritu. Tactical Approaches for Alleviating Distance in Global Software Development. **IEEE Software**, v. 18, n. 2. março/abril 2001. p. 22-29.

CATALDO, et al. On Coordination Mechanisms in Global Software Development. IN PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING, **ICGSE**. IEEE Computer Society, 2007

CHANG, Carl K.; ZHANG, Jia; JIANG, Tsang Ming. Formalization of Computer Supported Cooperative Work Applications. In: IEEE WORKSHOP ON FUTURE TRENDS OF DISTRIBUTED COMPUTING SYSTEMS, 8., 2001. **FTDCS'01**. IEEE, 2001.

CHAVES, A. P. et al**.** Ferramenta para instanciação de processos de software que permite o gerenciamento de projetos de desenvolvimento distribuído. In: 2ª CONFERÊNCIA IBÉRICA DE SISTEMAS E TECNOLOGIAS DE INFORMAÇÃO, 2007, Porto. **Novas Perspectivas em Sistemas e Tecnologias de Informação**. Porto, 2007. v. II.

_____. et al**.** Um modelo baseado em context-awareness para disseminação de informações em um Ambiente de Desenvolvimento Distribuído de Software. In: XXXIV CONFERENCIA LATINOAMERICANA DE INFORMÁTICA (CLEI 2008), 2008, Santa Fe. Anais da XXXIV **Conferencia Latinoamericana de Informática**, 2008. p. 1365-1374.

CHRISSIS, Mary Beth; KONRAD, Mike; SHRUM, Sandy. **CMMI®: Guidelines for Process Integration and Product Improvement**. Addison Wesley, 2003. 688 p.

CLEETUS, K. J. **Definition of concurrent engineering**. Technical Report. Concurrent Engineering Research Center(CERC), West Virginia University, 1992.

CLERC, V., LAGO, P., e VAN VLIET. Global Software Development: Are Architectural Rules the Answer?. In PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING (Agosto 27 - 30, 2007). **ICGSE**. IEEE Computer Society, Washington, DC, 225-234. DOI= http://dx.doi.org/10.1109/ICGSE.2007.21

COSTA, Ivanir. **Uma Proposta de Modelo para a Criação e a Organização de Processos de Produção em um Contexto de Fábrica de Software**. 2003. Tese (Doutorado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 2003.

CROOM, S. Methodology Editorial. **International Journal of Operations and Production Management**, v.22, n.2, p. 148-151, 2002.

CZAJA, Ronald; BLAIR, Johnny. **Designing Surveys**: a Guide to Decisions and Procedures. Pine Forge, 2005.

DAMIAN, Daniela E.; ZOWGHI, Didar**.** The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In: JOINT INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING, 14., 2002, Minneapolis. **RE'02**. Minneapolis: IEEE, 2002.

DAVID, R. & ALLA, I-I. Petri Nets for Modeling of Dynamic Systems — A Survey. **Automatica**, v. 30, 11.2, p. l75—2Ul, I994

DEFRANCO-TOMMARELLO, Joanna; DEEK, Fadi P.. Collaborative Software Development: A Discussion of Problem Solving Models and Groupware Technologies. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 35., 2002. **HICSS-35'02**. Hawaii: IEEE, 2002.

DELEN, Dursun; BENJAMIN, Perakath C.; ERRAGUNTLA, Madhav. **An integrated toolkit for enterprise modeling and analysis**. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, United States: Simulation. Phoenix, Arizona: Acm, 1999. p. 289 - 297.

DOSSICK, S.E. e KAISER, G.E. CHIME: A Metadata-Based Distributed Software Development Environment. JOINT SEVENTH EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SEVENTH ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING, September 1999, pp. 464-475

DUTTA, Amitava; ROY, Rahul. Offshore Outsourcing: Counteracting Forces and Their Dynamic Effects. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 38., 2005, Hawaii. **HICSS 2005**. Hawaii: IEEE, 2005. p. 1 - 9.

EBERT, Christof; NEVE, Philip De. Surviving Global Software Development. **IEEE Software**, v. 18, n. 2, p.62-69, Março/Abril 2001.

ECO, Umberto. **Como se faz uma tese**. 18. ed. São Paulo: Perspectiva, 2003. 182 p. (Coleção: ESTUDOS, 85).

EDWARDS, H. Keith; SRIDHAR, Varadharajam. Analysis of the Effectiveness of Gloval Virtual Teams in Software Engineering Projects. In: HAVAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 36., 2003, Hawaii. **Proceedings Hawaii**: IEEE, 2003.

EISCHEN, Kyle. Software Development: An outsider´s View. in **Computer,** p.36 IEEE, Maio 2002.

ELDEN, M.; CHISHOLM, R.F.. Emerging varieties of action research : introduction to the special issue. **Human Relations**, vol. 46, no. 2. 1993.

FABRI, J. A. et al. Tutorial: **Desenvolvimento e Replicação de uma Fábrica de Software**. In: IV Simpósio Internacional de Melhoria de Processo de Software, 2004, São Paulo. Anais do IV Simpósio Internacional de Melhoria de Processo de Software - Tutorial, 2004. v. 001. p. 3-1-3-50.

_____. ; TRINDADE, A. L. ; A. L'ERARIO, A. ; PESSOA, M. S. P. . **A Organização de uma Máquina de Processo e a Melhoria do Processo de Produção de Software em um Ambiente de Fábrica**. In: VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimento, 2007, Lima - Peru. Anais das VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimento, 2007. v. 1. p. 229-236.

_____ ; _____ ; BEGOSSO, Luis R ; A. L'ERARIO, Alexandre ; SILVEIRA, Fábio L F ; PESSOA, Marcelo S P . **Techniques for the Development of a Software Factory: case CEPEIN-FEMA.** In: 17th International Conference Software & Systems Engineering an their Applications, 2004, Paris. Annais of 17th International Conference Software & Systems Engineering an their Applications, 2004. v. 3.

_____. **Uma Proposta de Modelo para a Criação e a Organização de Processos de Produção em um Contexto de Fábrica de Software**. 2007. Tese (Doutorado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

FACHIN, Odilia. **Fundamentos da metodologia**. São Paulo: Saraiva, 2001.

FACIN, Ana Lúcia Figueiredo. **A relevância do uso de sistemas computacionais para obtenção de informação integrada na engenharia simultânea**. 1998. 117 f. Dissertação (Mestrado) - Departamento de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 1998.

FOREMAN, J. W. Gaining competitive advantage by using simultaneous engineering to integrate your engineering, design, and manufacturing resources. In: CASA/SME AUTOFACT '89 Conference, Oct, 1989. Anais GOLDSTEIN, D. An agent-based architecture for concurrent engineering. **Concurrent Engineering: research and applications**, v.2, n.2, p.117-23, Jun, 1994.

GALLIANO, A. Guilherme. **O método científico: teoria e prática**. Harbra, 1979.

GRINTER, Rebecca E. Using a Configuration Management Tool to Coordinate Software Development. In: PROCEEDINGS OF THE CONFERENCE ON ORGANIZATIONAL COMPUTING SYSTEMS, 1995, Milpitas. **COOCS 95**. Milpitas: ACM, 1995. p. 168 - 177.

GRUDIN, Jonathan. CSCW: History and Focus. **IEEE Computer**, New York, v. 27, n. 5, p.19-26, Maio 1994.

GRUNDY, John; HOSKING, John; MUGRIDGE, Warwick B**..** Inconsistency Management for Multiple-View Software Development Environments. **IEEE Transactions on software engineering**, v. 24, n. 11, p.960-981, nov. 1998.

HASEGAWA, K. Modeling, **Control and Deadlock Avoidance of FMS**, In: CONFERÊNCIAS PEQUENINAS, **XI CBA**, São Paulo, SBA, 1996, p.37—5l.

HENRICH, Andreas; MORGENROTH, Karlheinz. Supporting Collaborative Software Development by Context-Aware Information Retrieval Facilities. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 14., 2003, Praga. **DEXA'03**. Praga: IEEE, 2003.

HERBSLEB, James D.; GRINTER, Rebecca E.. Architectures, Coordination, and Distance: Conway. **IEEE Software**, v. 16, n. 1, p.63-70, 1999.

_____.; MOCKUS, Audris.. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. **IEEE Transactions on Software Engineering**, v. 29, n. 3, p.1-14, 2003.

HOFSTEDE, Geert; HOFSTEDE, Gert-Jan. **Cultures and Organizations: Software of the Mind**. New York: McGraw-Hill U.S.A., 2004.

HOPCROFT, John E.; ULLMAN, Jeffrey D**.. Introduction to automata theory, languages, and computation**. Reading: Addison-Wesley, c1979. 418 p. (Addison-Wesley series in computer science) ISBN 0-201-02988-X

HUPFER, Susanne et al. Introducing Collaboration into an Application Development Environment. In: COLLABORATIVE SUPPORTED COMPUTER WORK, 4., 2004, Chicago. **CSCW** '04. Chicago: ACM, 2004.

INCUBADORA virtual da fapesp - TIDIA-AE. Disponível em: <http://incubadora.fapesp.br/projects/tidia-ae/>. Acesso em: 26 out. 2005.

JALOTE, Pankaj; JAIN, Gourav. Assigning Tasks in a 24-Hour Software Development Model**.** In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 11., 2005, Busan. **APSEC'04**. Busan: IEEE, 2004.

JØRSTAD, Ivar; DUSTDAR, Schahram; THANH, Do Van. A Service Oriented Architecture Framework for Collaborative Services. In: INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISE, 14., 2005, Manchester. **WETICE'05**. Manchester: IEEE, 2005.

KAPLAN, Simon M.; BIGNOLI, Celsina; TOLONE, William J.. Process, space and software development support**.** In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP, 9., 1994. **International Software Process Workshop**.: IEEE, 1995. p. 100 - 103.

KEEGAN, John. **Inteligência na Guerra.** Companhia Das Letras, 2006.

KO, Hoo Sang et al. Concurrent Process Flow Analysis for Collaborative Process Planning. IECON -PROCEEDINGS-, USA, p. 1814-1818. 2003.

KOTLARSKY, J., VAN FENEMA, P. C. e WILLCOCKS, L. P.. Developing a knowledge-based perspective on coordination: The case of global software projects. **Inf. Manage**. 45, 2 (Mar. 2008), 96-108. DOI= http://dx.doi.org/10.1016/j.im.2008.01.001

L'ERARIO, Alexandre ; PESSOA, Marcelo Schneck de Paula. A dinâmica e as suas propriedades dos ambientes de desenvolvimento distribuído de software: um estudo de caso. In: XII CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 2006, San Luis. **Anais do XII Congreso Argentino de Ciencias de la Computación**, 2006. v. 1. p. 12-22

_____. et al. Desenvolvimento distribuído de software para sistemas pervasivos: um estudo de caso. In: SIMPÓSIO BRASILEIRO DE SISTEMA DE INFORMAÇÃO, 1., 2004, Porto Alegre. Simpósio. Porto Alegre: **SBSI**, 2004. p. 163 - 170.

_____. PESSÔA, Marcelo Schneck de Paula. An Analysis of the Dynamics and Properties of the Distributed Development of Software Environments: A Case Study. **Software Engineering Research and Practice** 2007: 471-477

_____. As alianças estratégicas no desenvolvimento distribuído de software. In: ANAIS DO ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO. **ENEGEP** Salvador. 2009

_____; PESSÔA, M. S. P.. Um método de ensino e práticas de desenvolvimento distribuído de software para cursos de graduação. In: COBENGE - CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 2008, São Paulo. Anais do XXXVI **Congresso Brasileiro de Educação em Engenharia**, 2008.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos da metodologia científica**. 6. ed. São Paulo: Atlas, 2005.

LAURINDO, Fernando J.B. **Tecnologia da Informação: Eficácia nas Organizações**. Futura, 2002.

LE, Taowen; W, Huang; ZHANG, Jin. China as a Software Outsourcing Outlet: Status, Enabling Factors, International Impact, and Growth Determinants. **Wireless Communications, networking and mobile computing**, 2007: IEEE, 2007. p. 6115 - 6121.

LEAVITT Neal, The Changing World of Outsourcing, **Computer**, vol. 40, no. 12, pp. 13-16, December, 2007.

LEE, James D.; HICKEY, Ann M.; ZHANG, Dongsong. ColD SPA: A Tool For Collaborative Process Model Development. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 33., Hawaii. **HICSS**. Hawaii: IEEE, 2000.

LOPES, L. T.; AUDY, J. L. N. Em busca de um modelo de referência para engenharia de requisitos em ambientes de desenvolvimento distribuído de software, in: WORKSHOP EM ENGENHARIA DE REQUISITOS, Piracicaba – SP, **WER03**, Brasil, p329-342, 2003.

MACIEL, P.; LINS, R.; CUNHA, P. Introdução às Redes de Petri e Aplicações, **X Escola de Computação**, Unicamp, 1996.

MALINIAK, L.Teamwork is the key to concurrent design. **Eletronic Design**, v.39, n. 1, p.37-50, Jan, 1991.

MARTIN, Patrick. A Management Information Repository for Distributed Applications Management. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS, 2., 1996.

MARTINS Raul C. B., Sistemas formais e desenvolvimento de software, Tech. Report CCR 064, Centro Científico Rio – IBM Brasil, Rio de Janeiro, Brasil, nov. 1988

MATLOFF, Norman. Offshoring: What Can Go Wrong?. **IEEE IT Pro**, n. 1, p.39-45, Jul/ago. 2005.

MILNOR, John. On the concept of attractor**. Communications In Mathematical Physics**, Springer Berlin / Heidelberg, p. 177-195. 1 jun. 1985.

MINTZBERG, Henry**. Criando organizações eficazes: Estruturas em cinco configurações**. 2. ed. São Paulo: Atlas, 2003. 334 p.

MÜHLBACHER, Robert; NEUMANN, Gustaf. Towards a Framework for Collaborative Software Development of Business Application Systems. In: INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURES FOR COLLABORATIVE ENTERPRISES, 6., 1996, Stanford. **WET ICE '96**. Stanford: IEEE, 1996.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of the IEEE,** v. 77, n. 4, p. 541-80, April, 1989.

NAKANO, Davi N. FLEURY, Afonso. Métodos de pesquisa na engenharia de produção. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO, 16. **ENEGEP**. Piracicaba: Unimep/Abepro/Multiview, 1996

NAVARRO, Leandro; PRINZ, Wolfgang; RODDEN, Tom A.. Towards open CSCW systems. In: PROC. IEEE 3RD WORKSHOP ON FUTURE TRENDS OF DISTRIBUTED COMPUTING SYSTEMS, 3., 1992, Taipei. Workshop on Future Trends of Distributed Computing Systems. Taipei: IEEE, 1992. p. 4 - 10.

NEAL Leavitt, The Changing World of Outsourcing, **Computer**, vol. 40, no. 12, pp. 13-16, Dec., 2007

O'DAY, V. L.; BOBROW, D. G.; SHIRLEY, M. The social-technical design circle. Ackerman, M. S. ed. PROCEEDINGS OF THE ACM 1996 CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK; **CSCW 96**; November 16-20; Boston, MA. New York: ACM; 1996; 160-169.

OHIRA, Masao et al. Accelerating Cross-Project Knowledge Collaboration Using Collaborative Filtering and Social Networks. In: INTERNATIONAL WORKSHOP ON MINING SOFTWARE REPOSITORIES, 27., 2005, Saint Louis. MSR'05. Saint Louis: ACM, 2005.

PASALA, Anjaneyulu et al. Context-aware mobile assistant agents in software project management. In: TENCON 2008 - 2008, TENCON 2008. **IEEE Region 10 Conference. Hyderabad**, India: IEEE, 2008. p. 1 - 6.

PETRI C.A., Kommunication mit Automaten. Schriften des IIM Nr.2, Institut für Instrumentelle Mathematik, Bonn, 1962. Traduzida para o inglês como: Communication with Automata, Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, Vol.1, Suppl.1, 1966.

PMI – Project Management Institute. **A guide to the project management body of knowledge**. 2 ed. Newton Square: PMI Publicatios, 2004, 390p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis Nicolas. Towards a Model of Software Development Process for a Physically Distributed Environment - Minimizing communication difficulties and adding planning and evaluation view to the software development life cycle. In: VIII CONGRESSO ARGENTINO DE CIENCIAS DE LA COMPUTACION, 2002, Buenos Aires: **CACIC 2002** - Anais del VIII Congresso Argentino de Ciencias de la Computacion. 2002. v. I, p. 798-809.

_____; _____. Uma Análise Comparativa de práticas de Desenvolvimento Distribuído de Software no Brasil e no exterior. In: XX SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE. Florianópolis: **SBES**, 2006. p. 255 - 270.

_____; _____; EVARISTO, Roberto. Distributed Software Development: Toward an Understanding of the Relationship between Project Team, Users and Customers. In: V ICEIS - INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 2003, Angers. Proceedings of **ICEIS** 2003 - the Fifth Conference on Enterprise Information Systems. Portugal: ICEIS Press, 2003. v. 3, p. 417-423.

RAPOSO A. B. **Coordenação em Ambientes Colaborativos Usando Redes de Petri**. (Tese de doutorado), Universidade Estadual de Campinas, Campinas – SP, Brasil, 2000.

REDDY, Ramana, Y. V.; SRINIVAS, K., JAGANNATHAN, V., and KARINTHI, R. 1993. Guest Editor's Introduction: COMPUTER SUPPORT FOR CONCURRENT ENGINEERING. **Computer** 26, 1 (Jan. 1993), 12-16.

RIBEIRO, L. . Métodos formais de especificação: Gramáticas de grafos. In: SBC Sul. (Org.). VIII Escola de Informática da SBC-Sul. Porto Alegre: Editora da UFRGS, 2000, v. , p. 1-33.

ROBINSON, M., Kalakota, **Offshore Outsourcing: Business Models, ROI and Best Practices**. EUA: Mivar Press.2004

ROBSON, Colin. **Real world research: a resource for social scientist and practitioner**. 2. ed. London: BLACKWELL-UK, 2002. 139 p.

RODDEN, Tom. A technological framework for CSCW. In: **CSCW**: SOME FUNDAMENTAL ISSUES, 1991. CSCW: Some Fundamental Issues. 1991: IEEE, 1991. p. 7/1 - 7/5.

SA, Jin; MASLOVA, Elena. A unified Process Support Framework for Global Software Development. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 26, Oxford. Proceedings Oxford: IEEE, 2002. p. 1 - 2.

SANCHEZ, Ligia. **Brasil tem potencial para ser pólo de nearshore: Sofisticação da indústria de TI e qualificação profissional são fatores que favorecem o País**. Disponível em: <http://www.itweb.com.br/shared/print_story.asp?id=107130>. Acesso em: 18 jan. 2006.

SANTOS, Antônio Raimundo Dos. **Metodologia científica: a construção do conhecimento**. Rio De Janeiro: DP&A Editora, 2001. 139 p.

SAYÃO, Luís Fernando. **Modelos teóricos em ciência da informação - abstração e método científico**. Ci. Inf., Abr 2001, vol.30, no.1, p.82-91. ISSN 0100-1965

SCHNEIDER, H. M. **Colaborando através da engenharia simultânea para a inovação**. In: SIMPÓSIO DE GESTÃO DA INOVAÇÃO TECNOLÓGICA, 18., São Paulo, 1994.

SCHOUERI, Ricardo. **Ambiente para desenvolvimento de algoritmos genéricos para programação da produção intermitente**. 1998. 119 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Universidade de São Paulo, São Paulo, 1998.

SHAMI, N. Sadat et al. An experimental simulation of multi-site software development. In: CONFERENCE OF THE CENTRE FOR ADVANCED STUDIES ON COLLABORATIVE RESEARCH, 10, **CASCON** 2004. 2004: ACM, 2004. p. 255 - 266.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Operating System Concepts**. 8. ed.: wiley, 2008.

SINHA, V.S.; SENGUPTA, B.; GHOSAL, S., An Adaptive Tool Integration Framework to Enable Coordination in Distributed Software Development, GLOBAL SOFTWARE ENGINEERING*, 2007. **ICGSE** 2007. Second IEEE International Conference on* , vol., no., pp.151-155, 27-30 Aug. 2007

SIQUEIRA, Fábio L. SILVA, Paulo S. M. As características do desenvolvimento distribuído de software. In: SIMPÓSIO BRASILEIRO DE SISTEMA DE INFORMAÇÃO, 1., 2004, Porto Alegre. Simpósio. Porto Alegre: **SBSI**, 2004. p. 170 - 178.

SLACK, Nigel; CHAMBERS, Stuart; JOHNSTON,Robert. **Administração da Produção**. ATLAS, 7ª edição, 2002

SOUZA, Cleidson R. B. de et al. Management of Interdependencies in Collaborative Software Development. In: PROCEEDINGS OF THE 2003 INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING, 2003. **ISESE'03**. 2003: IEEE, 2003. p. 255 - 266.

STACHOWIAC, Herbert. Models. In: SCIENTIFIC thought: concepts, methods andprocedures. Paris : Unesco, 1972, p. 145-166

STACHOWIAK, Herbert. **Allgemeine Modelltheorie**. Springer-Verlag, Wien and New York, 1973

SUZUKI, Junichi; YAMAMOTO, Yoshikazu. Leveraging Distributed Software Development. **IEEE Computer**, New York, v. 32, n. 9, p.59-65, setembro 1999.

TIWANA, Amrit. Beyond the black box: knowledge overlaps in software outsourcing. **IEEE Software**, n. 21. jan/fev 2004. p. 51-58.

TRINDADE, André L P ; et al . Desenvolvimento e Replicação de uma Fábrica de Software. In: IV JORNADAS IBEROAMERICANAS DE INGENIERÍA DEL SOFTWARE E INGENIERÍA DEL CONOCIMIENTO, Madri. **Anais del IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento**, 2004

_____. **Uma Contribuição para o entendimento da Ensinagem na Preservação do Conhecimento em Ambientes de Fábrica de Software**. 2006. Tese (Doutorado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

TURNER, Marlene E.; FULLER, Sally Riggs. **Groups at Work: Theory and Research**. London: Lawrence Erlbaum Assoc Inc, 2000.

TYRREL, Sebastián. The Many Dimensions of the Software Process. Crossovers of ACM - http://www.acm.org/crossroads/xrds6-4/software.html. ACM, 2002.

VANZIN, Mariangela; BLOIS, Marcelo; PRIKLADNICKI, Rafael; CECCATO, Ilmari; ANTUNES, Dante. Global Software Processes Definition in a Distributed Environment. In: PROCEEDINGS OF THE 29TH ANNUAL NASA/IEEE SOFTWARE ENGINEERING WORKSHOP, Washington DC, USA, 2005.

WESTBROOK, R.. Action Research: a new paradigm for research in production and operations management. **International Journal of Operations & Production Management**, vol.15, no. 12. 1.995

WOMACK, J. P., JONES, D. T., ROSS, D. **A máquina que mudou o mundo**. São Paulo, 1991.

WU, Haihong; YU, Chunyan; WU, Minghui. Using Extended Fuzzy-Timing Petri Nets to Model Role-Based and Agent Oriented Collaborative Virtual Environment. In: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 5., 2004, Hangzhou. Hangzhou: IEEE, 2004.

WU, Lei; SAHRAOUI, Houari. Supporting Web Collaboration for Cooperative Software Development. In: INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, 5., 2004, Beijing. **WI'04**. Beijing: IEEE, 2004.

YACOUB, Sherif; AMMAR, Hany; MILI, Ali. **Characterizing a Software Component**. Disponível em: <http://www.sei.cmu.edu/cbs/icse99/papers/34/34.htm>. Acesso em: 25 out. 2005.

YIN, Robert K.. **Estudo de caso: planejamento e métodos**. 3. ed. Porto Alegre: Bookman, 2005. 212 p.

ZANONI, R., AUDY, Jorge Luis Nicolas. Gerência de Projeto de Software em ambiente fisicamente distribuído In: VIII CONGRESSO ARGENTINO DE CIÊNCIAS DE LA COMPUTACION, 2002, Buenos Aires. Anaies del VIII Congresso Argentino de Ciencias de Computacion - **CACIC**. Buenos Aires: Depto Computacion - FCEN - UBA, 2002. v.1. p.810 – 820.

ZHANG, H. C. AND ZHANG, D. Concurrent engineering: an overview from manufacturing engineering perspectives. **Concurrent Engineering: research and application**, v.3, n.3, p.221-36, Sep, 1995.

ZURUWASKI, R.; ZHOU M. Petri nets and industrial applications: a tutorial. **IEEE Transactions on Industrial Electronics**, v. 41, n. 6, p. 567-583, December, 1994.