RAFAEL PRIKLADNICKI

MuNDDoS
UM MODELO DE REFERÊNCIA PARA DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação, pelo Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Jorge Luis Nicolas Audy

PORTO ALEGRE, 2003.

Av. Ipiranga, 6681 - Prédio 16
CEP 90619-900 - Porto Alegre - RS - Brasil
http://www.pucrs.br/biblioteca

Fone: 0 (xx) 51 3320-3544
Fax: 0 (xx) 51 3320-3548
e-mail: bceadm@pucrs.br

"Para meus pais, David e Marlene, pelo exemplo de dedicação e perseverança e por sempre acreditarem e me incentivarem no alcance dos meus objetivos".

AGRADECIMENTOS

Escrever uma dissertação é uma tarefa desafiadora. Agora, o resultado de um processo de pesquisa e vivência pessoal e profissional está pronto. Mas isto não seria possível sem o apoio de muitas pessoas, as quais eu gostaria de agradecer publicamente neste espaço.

À minha família, por tudo o que eles me proporcionaram até aqui e por tudo o que representam na minha vida.

Ao grande amigo Jorge L. N. Audy, pela amizade construída ao longo destes dois anos.

Ao meu orientador Prof. Dr. Jorge L. N. Audy, pela sua forma única de conduzir um trabalho, pela paciência de me manter sempre no foco e pelo grande aprendizado que tive com o seu jeito "não dou o peixe, apenas ensino a pescar".

Aos professores Dra. Karin Becker e Dr. Ricardo Melo Bastos, pelo acompanhamento do trabalho, principalmente pelas críticas, comentários e sugestões visando sempre contribuir.

A PUCRS e Faculdade de Informática, pela excelente infra-estrutura disponível e por todos estes sete anos de convivência (desde a graduação).

A Dell *Inc.,* por ter financiado a pesquisa e o meu curso de mestrado. Muito obrigado!

Ao professor Dr. Roberto Evaristo, da *University of Illinois at Chicago*, pelas ótimas intervenções e discussões que tivemos e pela grande contribuição nos artigos que publicamos.

Às organizações participantes dos estudos de caso, por acreditar no valor do trabalho.

A todos os entrevistados, pelo tempo dispensado e pela rica contribuição.

Aos meus colegas de mestrado, pelas discussões, trabalhos e, pela grande amizade.

A todos os meus amigos, por entender a importância do trabalho e entender que às vezes as tardes e noites trabalhando eram necessárias.

A todos que contribuíram para que eu adquirisse o conhecimento necessário e, acima de tudo, para me qualificar como um profissional competente e ciente do meu compromisso.

”Não é por que as coisas são difíceis que não ousamos,
é por que não ousamos que elas são difíceis”.

Sêneca – filósofo latino

RESUMO

Atualmente, é cada vez mais significativo o número de empresas que estão distribuindo seus processos de desenvolvimento de software ao redor do mundo, visando ganhos de produtividade, redução de custos e melhorias na qualidade. Por isso, o desenvolvimento distribuído tem atraído um grande número de pesquisas na área de engenharia de software nos últimos anos. Os engenheiros de software têm reconhecido a grande influência desta nova forma de trabalho e estão em busca de modelos que facilitem o desenvolvimento de software com equipes geograficamente dispersas. Além dos engenheiros, os gerentes e os executivos têm enfrentado desafios e dificuldades em diferentes níveis, envolvendo fatores técnicos e não-técnicos. Neste sentido, esta dissertação de mestrado tem como objetivo propor um modelo de referência para desenvolvimento distribuído de software, contemplando as dimensões técnicas e não-técnica e os fatores envolvidos em cada uma. O principal método de pesquisa utilizado foi o estudo de caso e a base empírica da pesquisa envolveu duas unidades de desenvolvimento de software de duas empresas multinacionais de grande porte localizadas no Brasil. A pesquisa contribui no sentido de propor um modelo de referência para a área de desenvolvimento distribuído de software, além de apresentar dados empíricos e sistematizar parte da teoria recente da área. Também contribui ao propor um modelo de classificação do nível de dispersão de projetos distribuídos, considerando os atores envolvidos.

Palavras-chave: Engenharia de software, processo de desenvolvimento de software, desenvolvimento distribuído de software, *offshore outsourcing*

ABSTRACT

Software has become a vital component of almost every business. Success increasingly depends on using software as a competitive advantage. More than a decade ago, seeking lower costs and access to skilled resources, many organizations began to experiment with remotely located software development facilities and with outsourcing. Economic forces are relentlessly turning national markets into global markets and spawning new forms of competition and cooperation that reach across national boundaries. This change is having a profound impact not only on marketing and distribution but also on the way products are conceived, designed, constructed, tested, and delivered to customers. The number of organizations distributing their software development processes worldwide aiming at heightened profit and productivity as well as cost reduction and quality improvements keeps increasing. Software development is increasingly a multi-site, multicultural, globally distributed undertaking. Engineers, managers, and executives face formidable challenges on many levels, from the technical to the social and cultural. More recently, attention has turned toward trying to understand the factors that enable multinationals and virtual corporations to operate successfully across geographic and cultural boundaries. This way, the purpose of this dissertation is to propose a reference model for distributed software development. The research method is case study and the empirical base involves two software development centers from two multinational organizations located in Brazil. The research contributions are the reference model and a model to classify the levels of dispersion in distributed projects. Moreover, empirical data is presented, systemizing part of the theory in this recent area.

Keywords: Software engineering, software development process, distributed software development, offshore outsourcing

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CASE – *Computer-Aided Software Engineering*

CT-SE – *Collaborative Technology to Support Software Engineering*

CIO – *Chief Information Officer*

CMM – *Capability Maturity Model*

DCS – Desenvolvimento Centralizado de Software

DDS – Desenvolvimento Distribuído de Software

EC – Estudo de Caso

EC1 – Primeiro Estudo de Caso

EC2 – Segundo Estudo de Caso

ES – Engenharia de Software

EUA – Estados Unidos da América

FCS – Fatores Críticos de Sucesso

GC – Gestão do Conhecimento

GSD – *Global Software Development*

IEC – *International Electrotechnical Commission*

IPI – Imposto sobre Produtos Industrializados

ISO – *International Organization for Standardization*

MSF - *Microsoft Solutions Framework*

PDS – Processo de Desenvolvimento de Software

PMI – *Project Management Institute*

PPB – Processo Produtivo Básico

PSI – Planejamento de Sistemas de Informação

PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul

RUP – *Rational Unified Process*

SCM – *Software Configuration* Management

SEBRAE – Serviço Brasileiro de Apoio às Micro e Pequenas Empresas

SEPG – *Software Engineering Process Group*

SI – Sistemas de Informação

SPICE - *Software Process Improvement and Capability Determination*

SQA – *Software Quality Assurance*

SW-CMM – *Capability Maturity Model for Software*

TI – Tecnologia da Informação

UD – Unidade Distribuída

UD1 – Unidade Distribuída 1

UD2 – Unidade Distribuída 2

UML – *Unified Modeling Language*

WMRC – *World Market Research Council*

SUMÁRIO

1 INTRODUÇÃO

"Você vê coisas e diz: Por que? Mas eu sonho coisas que nunca existiram e digo: Por que não?" George Bernard Shaw

Nos últimos anos pode-se perceber um grande avanço em direção à globalização dos negócios. Na área de desenvolvimento de software não é diferente. O software tem se tornado um componente estratégico para diversas áreas de negócio. Neste sentido, para as organizações que buscam sucesso, é clara a necessidade do uso da Tecnologia da Informação (TI) como diferencial competitivo [HER 01a].

Na área de Engenharia de Software (ES), mercados nacionais têm se transformado em mercados globais, criando novas formas de competição e cooperação que vão além das fronteiras dos países. Tem sido cada vez mais difícil justificar o desenvolvimento de software tradicional, centralizado dentro de uma organização [KAR 98]. Isto se deve principalmente a falta de maturidade dos processos, a não existência de padronização, a comunicação ineficiente e a existência de ferramentas com pouca capacidade de integração. Tem se tornado cada vez mais custoso e menos competitivo desenvolver software no mesmo espaço físico, na mesma organização ou até mesmo no mesmo país.

Por outro lado, o avanço da economia, a sofisticação dos meios de comunicação e a pressão por custos e tecnologia têm incentivado o investimento maciço no desenvolvimento distribuído de software (DDS[1]). Embora a ES ainda esteja longe de ser uma disciplina madura [KAR 98], as melhorias nas ferramentas e métodos nas últimas décadas têm permitido que grupos de diferentes localidades e culturas, com diferentes expectativas e objetivos, possam formar uma equipe para trabalhar em projetos distribuídos. Por isso, visando a redução de custos, maior qualidade no processo

[1] Existem diversas formas de caracterizar o desenvolvimento de software com equipes dispersas, tais como DDS (Desenvolvimento Distribuído de Software) [LAY 00], GSD (*Global Software Development*) [CAR 99] e *Multi-site development* [KAR 98]. Neste estudo, adota-se o termo DDS.

de desenvolvimento de software e a possibilidade de obter recursos em âmbito global [COC 02], [HER 99], muitas organizações começaram a investir em DDS.

O DDS tem sido caracterizado principalmente pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, mas estão localizados em cidades ou países diferentes [KIE 03]. Esta área de pesquisa apresenta grandes desafios e muitas lacunas a serem preenchidas. Segundo [CAR 99], o estudo envolvendo o DDS transcende a Ciência da Computação, sendo uma área multidisciplinar envolvendo principalmente Sociologia, Psicologia, Administração e Educação. Esta pesquisa procurou estudar as características do DDS, sua operacionalização e os Fatores Críticos de Sucesso (FCS) envolvidos.

Desta forma, a **questão de pesquisa** que norteou este estudo foi:

Como os fatores envolvidos no DDS podem ser relacionados em um modelo de referência específico para esta área?

1.1 OBJETIVOS

O **objetivo geral** foi propor um modelo de referência para a área de DDS, contemplando as dimensões técnica e não-técnica e os fatores envolvidos em cada uma delas. Este modelo de referência é composto por um conjunto de variáveis críticas presentes no ambiente de DDS (em cada dimensão), identificando como estas variáveis se relacionam e como elas determinam o sucesso dos projetos neste tipo de ambiente.

De forma a complementar o objetivo geral proposto, apresentam-se os seguintes **objetivos específicos**:

- Descrever as características da área de DDS, considerando as dimensões existentes;
- Identificar os fatores críticos de sucesso presentes no DDS;
- Identificar as práticas utilizadas em ambientes de DDS;
- Propor um modelo de classificação do nível de dispersão de projetos distribuídos.

1.2 MOTIVAÇÃO

Atualmente, é cada vez mais significativo o número de empresas que estão distribuindo seus processos de desenvolvimento de software ao redor do mundo. Por isso, o desenvolvimento distribuído tem atraído um grande número de pesquisas na área

de ES [DAM 02c], [DAM 03a], [HER 01a], [HER 01b]. Os engenheiros de software têm reconhecido a grande influência desta nova forma de trabalho e estão em busca de modelos que facilitem o desenvolvimento de software com equipes geograficamente dispersas. Além dos engenheiros, gerentes e executivos têm enfrentado diversos desafios e dificuldades em diferentes níveis.

Recentemente, diversos autores têm escrito sobre dificuldades, desafios e práticas do desenvolvimento distribuído de software, principalmente no exterior [CAR 99], [CAR 01a], [DAM 02a], [DAM 02b], [ESP 03], [EVA 00], [HER 01a], [HER 99], [HER 01b], [KAR 98], [OPP 02], [VOG 01]. No Brasil, apesar de a pesquisa ser ainda incipiente e se concentrar na proposta de ferramentas para atuar em ambientes de desenvolvimento distribuído [NOT 00], também existem trabalhos sobre o tema. O trabalho realizado por [MAI 99] teve como objetivo a especificação de um modelo de gerência de processos para apoiar o desenvolvimento de software realizado por equipes de trabalho geograficamente dispersas.

Segundo [CAR 99], não existem dúvidas para qualquer profissional que trabalha na área de ES que tanto o desenvolvimento de software tradicional (desenvolvimento centralizado de software – DCS) quanto o distribuído possuem diversas dificuldades. As principais características que diferenciam o DDS do DCS são [CAR 99]: **dispersão geográfica** (a distância entre equipe de projeto, clientes e usuários, por exemplo); **dispersão temporal** (diferenças de fuso-horário); e **diferenças culturais** (incluindo idioma, tradições, costumes, normas e comportamento).

Estas diferenças refletem-se em diversos aspectos [HER 01a]. Entre eles, destacam-se as questões:

- Estratégicas: envolvem a decisão de distribuir ou não o projeto e para qual unidade distribuída o projeto será enviado, tendo por base análises de risco e custo-benefício envolvidos;

- Culturais: envolvem as diferenças culturais (valores, princípios, etc.) entre as equipes distribuídas, localizadas em diferentes regiões;

- Técnicas: envolvem aspectos relativos a compatibilização da infra-estrutura tecnológica (redes de comunicação de dados, plataformas de hardware, ambiente de software, etc.) e ao conhecimento técnico necessário (processo de desenvolvimento, etc.) para o desenvolvimento dos projetos distribuídos;

- Gestão do conhecimento: envolvem aspectos relativos à criação, armazenamento, processamento e compartilhamento de informações nos projetos distribuídos.

O DDS no Brasil é um tema que tem sido cada vez mais explorado, devido principalmente à vinda de grandes empresas multinacionais que estão implantando unidades *offshore* de desenvolvimento de software no país. Desta forma, torna-se relevante o desenvolvimento de pesquisas nesta área.

1.3 ORGANIZAÇÃO DO VOLUME

Este volume está organizado em 7 capítulos. Na seqüência, no capítulo 2 apresenta-se o referencial teórico deste desta pesquisa, envolvendo os principais conceitos e implicações das áreas do estudo: desenvolvimento de software, dificuldades e desafios do processo de desenvolvimento e desenvolvimento distribuído de software. A apresentação deste referencial teórico é feita de forma abrangente, em virtude da natureza exploratória desta pesquisa e do uso de métodos qualitativos, que determinam a necessidade de uma abrangente e consistente fundamentação teórica [YIN 01].

No capítulo 3, apresenta-se a metodologia de pesquisa, descrevendo cada uma das etapas do estudo, justificando a escolha e uso dos métodos apresentados. No capítulo 4 apresenta-se uma proposta de classificação do nível de dispersão das equipes em projetos distribuídos, considerando a distância física entre os atores envolvidos.

No capítulo 5, descreve-se detalhadamente o estudo de caso, desenvolvido em duas unidades de desenvolvimento de software de duas organizações multinacionais de grande porte. O estudo, alicerçado na fundamentação teórica desenvolvida, aborda as principais dificuldades, soluções, práticas adotadas e fatores críticos de sucesso das empresas que implementam o desenvolvimento distribuído de software.

O modelo proposto é apresentado no capítulo 6, como conseqüência do processo de pesquisa como um todo, apoiado na revisão teórica desenvolvida (Capítulo 2) e nos resultados obtidos com o estudo de caso (Capítulo 5).

Finalmente, no capítulo 7 apresentam-se as considerações finais sobre o tema e enfocam-se os aspectos relacionados às contribuições e limitações deste estudo. Conclui-se destacando rumos para futuras pesquisas na área.

2 REFERENCIAL TEÓRICO

> *"Há dois tipos de conhecimento: conhecemos um assunto pelo nosso próprio conhecimento, ou sabemos onde encontrar informações sobre ele". Samuel Johnson, 1775.*

Coerente com estudos de base qualitativa, o referencial teórico representa importante etapa da pesquisa [YIN 01]. Na seção 2.1 apresenta-se a evolução do desenvolvimento de software. Após, nas seções 2.2 e 2.3 identificam-se os principais problemas e desafios que o desenvolvimento de software apresenta atualmente. Nas seções 2.4 e 2.5 apresentam-se o estado da arte do DDS e os trabalhos relacionados. Por último, na seção 2.6 complementam-se os trabalhos relacionados com a visão de outros autores, desenvolvendo uma análise crítica sobre o DDS.

2.1 O DESENVOLVIMENTO DE SOFTWARE

Nos últimos anos, o software se tornou um componente vital nos negócios. O sucesso de uma organização cada vez mais depende da utilização do software como um diferencial competitivo. Neste sentido, o principal desafio na área de ES nas últimas duas décadas tem sido o estudo e a melhoria da qualidade e redução de custo do software produzido [PRE 01]. Pode-se entender a ES como um conjunto de disciplinas. A literatura apresenta diversas definições, e algumas delas são apresentadas a seguir:

> "Engenharia de Software pode ser definida pelo estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais [PRE 01]".

> "Engenharia de Software é uma disciplina que reúne metodologias, métodos e ferramentas a serem utilizados, desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolver problemas inerentes ao processo de desenvolvimento e ao produto de software [CAR 01b]".

Ainda que muitas definições abrangentes tenham sido propostas, todas elas convergem no sentido de apontar para necessidades de maior rigor no desenvolvimento de software. Sendo assim, a partir da revisão bibliográfica e baseando-se nas definições anteriormente citadas, encontrou-se na definição de [IEE 93] uma forma mais abrangente para definir a Engenharia de Software. Esta definição diz que:

> "Engenharia de Software é a aplicação de um ambiente sistemático, disciplinado e quantificável para o desenvolvimento, operacionalização e manutenção do software; ou seja, a aplicação da engenharia ao software [IEE 93] ".

Tendo por base estas definições, [PRE 01] entende a engenharia de software através de camadas. Estas camadas abrangem três elementos fundamentais: ferramentas, métodos e processo. De acordo com a figura 2.1, cada um destes elementos corresponde a uma camada, sendo que a camada base representa o foco na qualidade. Isto significa que as que representam os elementos fundamentais devem possibilitar ao gerente o controle do processo de desenvolvimento de software e oferecer ao desenvolvedor uma base para a construção de software de alta qualidade.

Figura 2.1 – As camadas da Engenharia de Software [PRE 01]

Os métodos de ES proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste, manutenção, entre outros.

As ferramentas de engenharia de software proporcionam apoio automatizado ou semi-automatizado aos métodos. Atualmente, existem ferramentas para sustentar cada um dos métodos citados anteriormente. Quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser usada em outra, é estabelecido um sistema de suporte ao desenvolvimento de software chamado Engenharia de Software Auxiliada por Computador (CASE – *Computer-Aided Software Engineering*).

Segundo [PRE 01], o processo é a camada mais importante da ES. Esta camada constitui o elo de ligação entre as ferramentas e os métodos, além de possibilitar um desenvolvimento racional do software. Um processo define a seqüência em que os métodos serão aplicados, como os produtos serão entregues, os controles que ajudam a

assegurar a qualidade e a coordenar as mudanças, e os marcos de referência que possibilitam aos gerentes de software avaliar o progresso do desenvolvimento [PRE 01].

Um processo de desenvolvimento de software é representado por um modelo, enquanto que o modelo é operacionalizado por meio de uma metodologia. Existem diversos modelos de processo de desenvolvimento de software, e cada modelo pode ter mais do que uma metodologia que o operacionaliza. A metodologia estabelece basicamente a seqüência das atividades e como elas se relacionam entre si, identificando o momento em que os métodos e as ferramentas serão utilizados.

Sendo assim, um processo de desenvolvimento de software deve ser implementado de acordo com um modelo previamente definido, seguindo uma metodologia que se sincronize as necessidades e objetivos existentes, servindo de guia para a correta utilização dos métodos e das ferramentas, tendo sempre destacado que a camada básica é o foco na qualidade (Figura 2.1).

Com relação à qualidade, existem alguns princípios da ES que descrevem de maneira geral as propriedades desejáveis para um produto de software [CAR 01b]. Entre estes princípios, pode-se citar:

- **Formalidade**: por ser uma atividade criativa, o desenvolvimento de software tende a ser não estruturado, pois depende da "inspiração do momento". Mas através de uma sistemática formal, é possível produzir produtos mais confiáveis, controlar o seu custo e ter mais confiança no seu desempenho. A formalidade não deve restringir a criatividade, mas deve melhorá-la.
- **Abstração**: é o processo de identificação dos aspectos importantes de um determinado fenômeno. Podem existir diferentes abstrações da mesma realidade, cada uma fornecendo uma visão diferente da realidade e servindo para diferentes objetivos.
- **Decomposição**: uma das maneiras de trabalhar com a complexidade é subdividir o processo em atividades específicas, atribuídas a especialistas de diferentes áreas. Esta separação permite o planejamento das atividades e diminui o tempo extra que seria gasto mudando de uma atividade para outra. Além do processo, o produto também pode ser desenvolvido através de sub-produtos, definidos de acordo com o sistema que está sendo desenvolvido. Entre as vantagens desta decomposição está a execução das atividades de forma paralela.
- **Generalização**: a generalização pode ser boa, mas ao mesmo tempo pode trazer algumas desvantagens no desenvolvimento de um produto de

software. Uma das vantagens é a possibilidade da reutilização de uma solução em diversos pontos do sistema. Mas uma solução genérica é bem mais custosa em termos de velocidade de execução ou tempo de desenvolvimento. O importante é avaliar a necessidade de se desenvolver uma solução generalizada, dependendo do sistema e dos custos envolvidos;

- **Flexibilidade**: quando se trata deste princípio fala-se na possibilidade de um produto ser modificado com facilidade. O processo deve ter flexibilidade suficiente para permitir que componentes do produto desenvolvido possam ser utilizados em outros sistemas, e deve-se avaliar também a sua portabilidade para diferentes sistemas computacionais.

Todos estes princípios isolados não são suficientes para guiar o desenvolvimento de software. Eles devem ser aplicados dentro de um contexto onde exista um processo de desenvolvimento de software bem definido, utilizando um modelo e uma metodologia adequados, com métodos e ferramentas de apoio.

Cabe salientar que não existe um modelo de processo ideal. A escolha depende do tamanho da organização, da experiência da equipe, da natureza e da complexidade da aplicação, do prazo de entrega, entre outros fatores. Existem diversos modelos de processos disponíveis e prontos para serem utilizados. Entretanto, cada vez mais se busca adequar um modelo a um cenário específico. Além disso, deve existir um modelo de ciclo de vida fortemente conectado ao modelo de processo de desenvolvimento de software escolhido.

2.2 PROBLEMAS DO DESENVOLVIMENTO DE SOFTWARE

Os problemas que afligem o desenvolvimento de software podem ser caracterizados a partir de uma série de perspectivas diferentes. A partir da ampla revisão bibliográfica realizada [AUD 01], [CAR 01b], [PET 01], [PRE 01], [PRI 02a], [PRI 02c], [ROY 98], [SOM 03] identificaram-se diversos problemas inerentes ao processo de desenvolvimento de software. A seguir serão destacados os principais problemas identificados.

2.2.1 Especificação de Requisitos

Segundo [PET 01], existe um grande obstáculo no desenvolvimento de software que é o levantamento e a especificação de requisitos. Na maioria das empresas, onde não existe um processo formal definido, não existe um tempo suficiente para fazer

um profundo levantamento de requisitos sobre um determinado sistema, o que faz com que o software seja desenvolvido em um tempo muito maior do que o planejado. Além disso, muitas vezes não existe uma preocupação em formalizar determinados procedimentos, bem como uma discussão sobre os requisitos para se chegar em uma solução [SWE 01]. Tudo é realizado em paralelo, o que também impede a coleta de métricas e prejudica a produtividade. Na prática, algumas vezes o problema ocorre pois faltaram alguns requisitos, e em outras vezes os requisitos existiam mas foram especificados de forma insuficiente à contemplar todas as funcionalidades [PET 01].

2.2.2 Cumprimento dos Prazos

Freqüentemente as estimativas de prazos dos projetos de software são definidas de forma bastante imprecisas, e isto reflete diretamente na sua entrega, podendo ocorrer atrasos e problemas com o cliente [McC 96]. Isto ocorre devido principalmente à inexistência de um tempo adequado para coletar dados sobre o processo de desenvolvimento de software. Com poucos dados, as estimativas de prazos acabam não sendo reais, tendo como resultado uma estimativa bastante ruim. Além disso, muitas vezes os projetos são mal dimensionados, prevendo muito pouco tempo ou tempo em demasia [PRE 01]. E quando não existe uma indicação sólida de produtividade, não existe uma possibilidade precisa de avaliar a eficácia de novas ferramentas, métodos ou padrões.

2.2.3 Custo / Benefício

O custo de um projeto envolve a avaliação e definição de estimativas que contemplam o esforço que será gasto para o desenvolvimento de um determinado software. O custo pode ser expresso através de algumas variáveis (dias, horas, etc.), e sobre estas variáveis se estipula um valor que será cobrado pelo desenvolvimento [PMI 00]. Mas da mesma forma que ocorre na definição de prazos, muitas vezes não existem dados suficientes para se determinar um custo adequado. Sendo assim, muitas vezes os projetos acabam sendo mal avaliados e seu custo pode ser um obstáculo na hora de fechar um contrato com um cliente [PET 01]. Além disso, muitas vezes não se realiza uma análise de custo-benefício do desenvolvimento de um determinado projeto. Isto deve ser feito, identificando os custos e benefícios envolvidos.

2.2.4 Produtividade

A produtividade é uma peça chave no processo de desenvolvimento de software. Segundo [PRE 01], significa rendimento, facilidade de produzir. E, muitas

vezes, a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços. Como conseqüência, o cliente mostra-se freqüentemente insatisfeito com o sistema concluído. Os projetos normalmente são executados de qualquer maneira, com um vago indício das necessidades do cliente. A comunicação entre o cliente e o desenvolvedor muitas vezes é insuficiente. Falta uma sistemática onde seja possível interagir para captar as informações necessárias e para a produtividade se tornar um diferencial de uma área de desenvolvimento de software [McC 96].

2.2.5 Qualidade e Teste de Software

A qualidade e o teste de software são considerados problemas no processo de desenvolvimento de software pois muitas pessoas não têm uma noção clara destes conceitos [COR 01]. A prática de testes de software é recente, e apenas agora sua importância está sendo realmente entendida. O mesmo ocorre com a verificação da qualidade como um todo, do início ao fim do processo de desenvolvimento de software [SCH 99]. A globalização do processo de desenvolvimento de software tem destacado a importância da adoção de modelos de qualidade internacionalmente aceitos (SPICE, ISO, CMM), chamando a atenção para a adoção de padrões visando reduzir os problemas existentes na área de ES.

2.2.6 Trabalho em equipe

Equipes de desenvolvimento de software normalmente são compostas por pessoas que têm suas próprias idéias de como solucionar problemas técnicos específicos. Muitas vezes isto é prejudicial, causando problemas que poderiam ser solucionados se conduzidos em um ambiente de trabalho em grupo. Um grupo com personalidades e conhecimentos complementares pode trabalhar melhor como equipe do que um grupo formado considerando apenas as habilidades técnicas de seus integrantes [SCH 00]. Isto é importante, pois a existência de restrições de prazos, custos e escopo nem sempre permitem a implantação da melhor solução técnica. É importante existir uma interação positiva entre os membros de uma equipe, administrando conflitos, potencializando conhecimentos complementares visando obter melhores resultados [SOM 03]. Muitos problemas podem surgir da diferença e da comunicação entre as equipes, e principalmente, quando os interesses pessoais se sobrepõem aos interesses do grupo.

2.2.7 Capacitação de Pessoal

A maioria dos problemas do desenvolvimento de software são causados por falhas humanas do dia-a-dia [PRE 01]. Para ilustrar, pode-se citar alguns exemplos do

que acontece na prática, onde gerentes de nível médio e superior sem nenhum conhecimento em software recebem a responsabilidade pelo seu desenvolvimento. Além disso, os profissionais da área de software têm recebido pouco treinamento formal em novas técnicas de desenvolvimento [McC 96]. Algumas pessoas desenvolvem através de tentativa e erro, enquanto outras desenvolvem práticas inadequadas que se refletem diretamente na qualidade e manutenção do software.

Todos resistem a mudanças, mas ao mesmo tempo todos deveriam ter a oportunidade para parar e refletir que, enquanto o potencial de computação (hardware) experimenta novas e enormes mudanças em um curto espaço de tempo, as pessoas da área de software responsáveis pelo aproveitamento deste potencial muitas vezes se opõem à mudança quando ela é discutida e resistem à mudança quando ela é introduzida [SOM 03].

2.2.8 Planejamento

O ponto central de qualquer esforço para o desenvolvimento de um software é especificar, projetar e testar a construção conceitual do software que está sendo criado. O planejamento deve ser o ponto de partida de qualquer atividade relacionada ao desenvolvimento de software. Muitas vezes o planejamento é esquecido, o que aumenta consideravelmente as chances de ocorrer algum problema [AUD 01]. As principais dificuldades estão relacionadas diretamente com a gestão do projeto e sua organização. A ausência de uma etapa formal de planejamento pode ser apontada como um dos principais problemas no processo de desenvolvimento de software, diluindo decisões críticas ao processo como um todo em etapas subseqüentes perdendo-se a dimensão sistêmica do problema em análise [AUD 01], [REP 98].

James Martin, em 1991, já apontava para a necessidade de inserção desta etapa no processo de desenvolvimento de software [MAR 91]. O modelo proposto por este autor considerava a existência de quatro grandes etapas no processo de desenvolvimento de software: planejamento, análise do negócio, projeto do software e construção. Seus estudos apontavam que a ausência de maior rigor nesta etapa de planejamento acarretava um grande número de problemas nas etapas subseqüentes. Abordagens mais recentes [AUD 01], [BOA 97], [ROY 98] incorporam a etapa de planejamento no processo de desenvolvimento de software, visando se antecipar a possíveis problemas que a ausência desta etapa podem causar. Entretanto, ênfases de base mais tecnicista tendem a abandonar esta perspectiva, se concentrando em aspectos mais intrínsecos às etapas mais técnicas do desenvolvimento de software em si. Mais recentemente, [PRE 01] retoma este problema ao afirmar a importância e a necessidade da etapa de planejamento no processo de desenvolvimento de software.

2.2.9 Motivação

Motivação, segundo [SCH 00], é o ato de dar motivo, despertar o interesse por alguma atividade. Estar motivado significa que, além de ter um conhecimento e uma habilidade técnica, é necessário ter interesse em desempenhar determinada função. Uma pessoa motivada pode ter muita facilidade em produzir. Por outro lado, uma pessoa desmotivada pode gerar muitos problemas no desenvolvimento de software.

Em 1954, A. Maslow disse que as pessoas eram motivadas por satisfazer suas necessidades, e estas poderiam ser organizadas em um conjunto de níveis, como ilustra a figura 2.2. As prioridades humanas são primeiramente de satisfazer as necessidades dos níveis mais baixos antes das necessidades mais abstratas nos níveis superiores [SOM 03].

Figura 2.2 – Modelo de motivação em um conjunto de níveis [SOM 03]

Para as pessoas que trabalham com desenvolvimento de software, assegurar as necessidades de satisfação social, de reconhecimento e de realização pessoal são considerados desafios sob um ponto de vista gerencial. Por isto, a falta de motivação pode ocasionar diversos problemas dentro de um contexto de desenvolvimento de software, refletindo diretamente na produtividade, na qualidade e principalmente na responsabilidade [McC 96], [SCH 00].

2.2.10 Manutenção

A manutenção de software é um conceito antigo e extremamente importante e que, muitas vezes, não tem sua importância reconhecida. Por definição, a manutenção é muito mais do que consertar erros [PRE 01]. Por muito tempo era uma fase negligenciada do processo de desenvolvimento de software. Na prática, a falta de controle e disciplina nas atividades de desenvolvimento quase sempre se traduz em problemas durante a manutenção do software. Entre os diversos problemas, [PRE 01] destaca os principais, entre eles:

- É difícil e às vezes impossível rastrear a evolução através de muitas versões ou lançamentos. As mudanças não estão corretamente documentadas.
- É difícil rastrear o processo através do qual o software foi desenvolvido.
- A documentação não existe ou é de má qualidade. Reconhecer que um software deve ser documentado é o primeiro passo, mas toda a documentação gerada deve estar de acordo com a lógica adotada.
- A maioria dos softwares não é projetada para sofrer mudanças.
- A manutenção não é considerada importante. Grande parte desta percepção vem do elevado nível de frustração associado ao trabalho de manutenção.

Todos estes problemas podem ser atribuídos ao grande número de sistemas existentes que foram desenvolvidos sem levar em consideração alguma metodologia, regras, etc. De um modo geral, a engenharia de software atualmente oferece algumas soluções para os problemas associados à manutenção [PRE 01].

2.2.11 Gerência de Projeto

Segundo [PMI 00] a gerência de projetos é a aplicação de conhecimentos, habilidades e técnicas para projetar atividades que visam atingir as necessidades e expectativas das partes envolvidas em um projeto [PMI 00]. Gerenciar projetos de software de uma maneira disciplinada, utilizando-se de princípios de engenharia, de forma a obter um sistema economicamente viável, seguro e eficiente não é uma tarefa fácil [SCH 00]. Exige do profissional desta área não apenas o conhecimento de linguagens de programação, métodos de análise, etc., mas também de aspectos gerenciais. O domínio de técnicas, métodos e ferramentas gerenciais que conduzem projetos de desenvolvimento de software ao resultado esperado são uma exigência para a evolução profissional [PMI 00].

A gerência de projeto tem uma importância fundamental para o sucesso do desenvolvimento de software. Todos os problemas citados até o momento podem ser tratados, ou ao menos identificados, se o projeto for bem gerenciado. Uma má gerência de projeto pode significar a perda do projeto e dos recursos envolvidos [ROY 98].

2.2.12 Análise Crítica

Além dos problemas citados anteriormente, foram identificados na literatura outros problemas, tais como aceitação do software pelo usuário, falta de comunicação,

tempo de desenvolvimento muito longo (hardware e software obsoletos), documentação inexistente, incompleta ou desatualizada. Devido ao grande número de problemas, identificou-se um conjunto de categorias para agrupá-los, apresentando-os de uma forma sintetizada. Para isto, utilizou-se a classificação proposta por [GLA 98], onde este identifica as principais causas de falhas no processo de desenvolvimento de software:

- Objetivos do projeto não estão totalmente claros e especificados;
- Má elaboração do planejamento e da estimativa;
- A organização incorporou uma nova tecnologia;
- As técnicas de gerência de projeto não existem ou são inadequadas;
- A equipe não possui experiência suficiente para desenvolver o projeto;
- O equipamento existente na organização apresenta uma baixa performance de hardware e software;
- Outros problemas de performance e/ou eficiência.

Buscando sintetizar a categorização proposta por [GLA 98] com os problemas identificados na revisão teórica desenvolvida (seções 2.2.1 a 2.2.12), consolidou-se um esquema de representação, agrupando os problemas por temas convergentes. A figura 2.3 apresenta a proposta de agrupamento, classificando-os em categorias mais amplas.

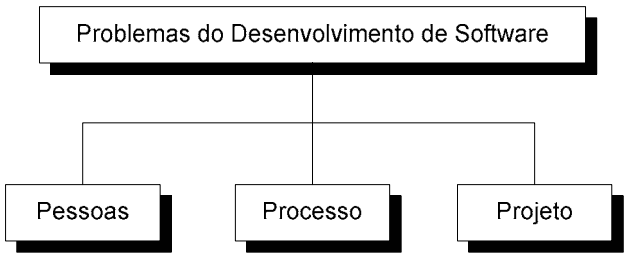

Figura 2.3 – Categorias de problemas do desenvolvimento de software

De acordo com os principais problemas anteriormente apresentados, as categorias propostas agrupam os seguintes problemas (Quadro 2.1):

Quadro 2.1 – Problemas agrupados por categorias

Categoria	Problemas	Autores relacionados
Pessoas	Capacitação de Pessoal Motivação Produtividade Trabalho em Equipe	[McC 96], [PRE 01], [SOM 03] [McC 96], [SCH 00] [McC 96] [SCH 00], [SOM 03]
Processo	Especificação de Requisitos Qualidade do Software Manutenibilidade	[PET 01], [PRE 01] [COR 01], [PRE 01], [SCH 99] [PRE 01], [SOM 03]
Projeto	Custo Gerência de Projeto Planejamento Prazo	[PET 01], [PMI 00] [PMI 00], [ROY 98] [AUD 01], [PRE 01], [REP 98] [McC 96], [PRE 01]

O quadro 2.1 propõe o agrupamento dos problemas nas categorias pessoas, processo e projeto. Os problemas relacionados às pessoas dizem respeito a características que afetam diretamente os recursos humanos envolvidos no desenvolvimento de software. Os problemas relacionados ao processo dizem respeito à forma com que o projeto está sendo desenvolvido. Por último, a categoria os problemas relacionados ao projeto dizem respeito a características necessárias em nível de projeto.

2.3 DESAFIOS DO DESENVOLVIMENTO DE SOFTWARE

Os principais desafios do desenvolvimento de software estão fortemente ligados com os problemas relatados na literatura, os quais foram tratados na seção anterior. Esta seção aborda os principais desafios identificados.

2.3.1 Gerência de riscos

Segundo [McC 96], gerenciar riscos significa identificar, tratar e eliminar fontes de riscos antes que eles se tornem uma ameaça concreta para o término de um projeto de software. Riscos podem ser tratados em diferentes níveis e a gerência de risco compreende algumas categorias e subcategorias, como pode ser visto na figura 2.4.

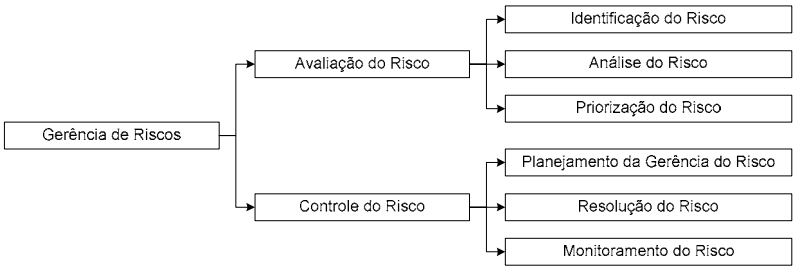

Figura 2.4 – Gerência de risco segundo Boehm, 1989 [SOM 03]

A gerência de riscos nunca foi uma tarefa simples [BER 97], e é tida como desejável dentro do processo de desenvolvimento de software. Atualmente um dos maiores desafios é a avaliação de riscos, prevendo possíveis danos que possam ser causados no futuro. A não existência da gerência de riscos pode acarretar em muitos problemas indesejáveis para a empresa e para o cliente, tais como o atraso no projeto, custos maiores, entre outros.

2.3.2 Planejamento

Muitas organizações não possuem um planejamento adequado no que diz respeito ao desenvolvimento de software. Antes de se iniciar o desenvolvimento de qualquer sistema um passo obrigatório deve ser o planejamento. Planejar significa definir

as estratégias que deverão conduzir o processo de desenvolvimento como um todo, ao longo do tempo. E esta etapa deve ser uma etapa preliminar a um conjunto de ciclos de projetos derivados deste processo de planejamento [PRI 02c]. Muitas vezes este planejamento não ocorre, o que faz com que a empresa não tenha uma visão geral de todos os projetos que estão sendo desenvolvidos, nem dos projetos que estão por vir, e muito menos da prioridade de cada um deles. Além disso, informações podem ser perdidas sem que haja um planejamento efetivo e uma visão de futuro adequada. Definir as estratégias da empresa na área de Sistemas de Informação (SI), a partir de um processo formal de planejamento é um grande desafio para as organizações [AUD 01].

2.3.3 Padrões de Desenvolvimento de Software

Desenvolver software respeitando padrões é muito importante quando se busca qualidade, facilidades de manutenção, reutilização de código e clareza no entendimento do que está sendo desenvolvido [COR 01]. Muitas empresas não investem na padronização de suas atividades, o que acaba deixando cada integrante da equipe de desenvolvimento livre para implantar o seu próprio método de desenvolvimento. A não utilização de uma padronização, tanto em código quanto em processo e documentação, acaba por transformar um sistema num conjunto de módulos, sendo que cada módulo só é entendido pelo seu próprio autor. Além disso, a não padronização no processo de desenvolvimento de software compromete aspectos relacionados com qualidade e confiabilidade do projeto.

2.3.4 Melhoria de Processos de Software

Muitas organizações têm buscado modelos de verificação e reconhecimentos de níveis de maturidade do seu processo de desenvolvimento de software, tais como CMM[2] (*Capability Maturity Model*). Isto se deve à necessidade de as organizações contratantes ter um mínimo de garantia sobre a qualidade do processo utilizado pela organização ou laboratório de desenvolvimento de sistemas parceiros.

Desenvolver software de qualidade, dentro dos prazos estabelecidos e sem necessitar de mais recursos do que os alocados tem sido o grande desafio da ES [PRE 01]. A principal causa dos problemas, apontada pelos especialistas, é a falta de um processo de desenvolvimento claramente definido e efetivo. Conhecer o processo significa conhecer como os produtos e serviços são planejados, produzidos e entregues.

[2] CMM – Modelo de qualidade utilizado pelas organizações como guia para definir e orientar um trabalho de melhoria e maturidade em seus processos de software [PAU 93], [PRE 01].

Neste contexto surgem os modelos de qualidade, nos seus diversos níveis. O reconhecimento em modelos de referência demonstra a conformidade da empresa em relação a requisitos de padrões normativos nacionais e internacionais. No caso específico do CMM, no processo de desenvolvimento de software, se enfatiza a documentação dos processos, seguindo a premissa de que para construir ou realizar alguma melhoria do processo é preciso conhecê-lo e entendê-lo, e que a qualidade de um produto é reflexo da qualidade e gerenciamento do processo utilizado no seu desenvolvimento.

2.3.5 Reutilização

A política de reutilização de software de uma empresa não pode ser vista como uma questão puramente operacional, mas como uma estratégia de alto-impacto que garanta aos grupos dedicados ao desenvolvimento de software o aumento de produtividade e redução de prazos, a retenção de conhecimento, a otimização de recursos e redução de custos e a melhoria da qualidade. Para tanto, segundo [DOU 97], é necessário mobilizar a alta-direção da empresa e empreender medidas que contribuam para estabelecimento de um processo gradativo e coordenado de reuso, não só de código de software, mas também de especificações, arquiteturas, casos de teste, dados, protótipos, planos, documentação e *frameworks*, denominados de artefatos.

Um repositório de artefatos reutilizáveis deve ser otimizado, composto tanto de itens genéricos como (ex: componentes para segurança de acesso), quanto especializados numa aplicação de mercado como (ex: marketing). Os critérios para eleger o desenvolvimento de um artefato como reutilizável está estreitamente ligado ao retorno financeiro esperado pelo reuso em relação ao investimento para torná-lo genérico, padronizado, documentado, certificado e classificado.

Além disso, um repositório deverá ser construído dentro de uma estratégia que privilegie artefatos (1) pertencentes a domínios estratégicos para a empresa e com potencial para atender a uma família de sistemas, (2) que tenham um alto valor de retorno sobre o investimento devido a uma expectativa de reutilização freqüente, e (3) que tenham sido demandados pelos desenvolvedores segundo necessidades reincidentes. De todo modo, a reutilização requer uma grande mudança de cultura da empresa, e este é um grande desafio, principalmente àquelas empresas desacostumadas a utilizar e obter vantagens destas técnicas [DOU 97].

2.3.6 Gestão do Conhecimento (GC)

Gestão do conhecimento (GC) é o nome dado ao conjunto de práticas que visam à manutenção do conhecimento nas organizações [NON 94]. Para [NON 94], a

gestão do conhecimento é "uma estratégia que transforma bens intelectuais da organização - informações registradas e o talento dos seus membros - em maior produtividade, novos valores e aumento de competitividade". Pode-se ainda tratar a GC como um conjunto de processos e técnicas que coordena a criação, disseminação, armazenamento e utilização do conhecimento buscando atingir plenamente os objetivos da organização. Em termos práticos, isto significa garantir que todos dentro da organização tenham acesso ao conhecimento quando, onde e na forma que eles necessitam. Além disso, significa ajudar e motivar que detentores de conhecimentos importantes compartilhem seu conhecimento.

Gerir conhecimento não é um conceito novo, ele apenas está sendo esquematizado e disponibilizado de uma forma nova pelas novas tecnologias, mídia, dispositivos e técnicas. Qualquer que seja sua forma (tácito ou explícito), as empresas estão percebendo cada vez mais que o recurso de conhecimento se tornou a chave para estabelecer vantagens competitivas e duradouras. Presencia-se atualmente apenas o começo do desenvolvimento de novos modelos de negócio entre empresas. A drástica redução nos custos de troca de informações e a ampla possibilidade de se trabalhar de maneira síncrona e assíncrona com pessoas em diferentes localidades estão permitindo que o conhecimento e o capital intelectual sejam efetivamente compartilhados. Verifica-se um número crescente de alianças entre empresas de todos os tamanhos e na emergência de redes colaborativas de empresas essencialmente virtuais. Esta tendência tende a se acelerar, incentivando cada vez mais a interligação entre locais e culturas distintas.

Em relação ao desenvolvimento de software, diversos autores abordam a questão da GC de forma integrada com a gerência de projetos e o conhecimento organizacional [DES 02], [KAT 02], [NON 94]. Segundo [KAT 02], considerando o contexto de projetos de desenvolvimento de software, o conhecimento deve estar sempre presente, muito além do ciclo de vida de um único projeto, existindo a necessidade de se gerenciar o conhecimento ao longo do tempo. Dentro dos projetos, o conhecimento é gerado enquanto as atividades ocorrem, e o contexto sócio-cultural que gerou este conhecimento é essencial. Sendo assim, o desafio é encontrar formas de prover uma eficiente gestão de conhecimento em diversas situações.

2.3.7 Novos Ambientes de Desenvolvimento

2.3.7.1 *E-Business* e Desenvolvimento *Web*

Velocidade, mudanças contínuas e competição global estão forçando as empresas a redefinirem radicalmente seus processos para sobreviverem. Os clientes estão ficando cada vez mais exigentes, sofisticados e seletivos. A conseqüência é o

investimento em novas tecnologias e formas de fazer negócios [STE 00]. Dada a relevância cada vez maior da internet na vida cotidiana e a tendência de uma sociedade cada vez mais voltada para o mundo digital, as empresas estão se reestruturando de maneira a adequar-se a esta nova realidade. E estes novos ambientes de desenvolvimento têm sido um dos grandes desafios das empresas. Segundo [ALB 00], o *e-business* propicia várias vantagens competitivas às empresas, dentre elas a conveniência (como a cadeia de valores está conectada via *extranet*, as empresas não precisam emitir pedidos e esperar alguns dias para a confirmação dos mesmos); o custo; a fidelidade (trabalhando em tempo real, a empresa pode responder de forma mais eficaz às exigências de seus clientes, aumentando a chance de tornar o cliente fiel); e a negociação direta (eliminação dos intermediários nas negociações trazem economias entre 30% e 80%).

Apesar das vantagens obtidas e da inquestionável tendência de digitalização da economia e dos negócios, segundo [ALB 00], o *e-business* oferece algumas desvantagens tais como o congestionamento (novas tecnologias estão em desenvolvimento para tornar a internet mais veloz); segurança (apesar dos avanços, a internet não é totalmente segura e está sujeita a diversos problemas de invasão e roubo de informações confidenciais); limitações (para que o *e-business* realmente dê vantagem competitiva à empresa, é preciso que esta esteja preparada para tal, munida de pessoal capacitado e muito bem treinado, provendo-os de tecnologia da informação adequada).

Em suma, o mais importante é a necessidade de as empresas trabalharem para integrar, primeiramente, seus processos internos e organizá-los de maneira que possam trabalhar em tempo real, sem burocracia. Deve existir também um ambiente de desenvolvimento que permita a empresa alcançar os objetivos propostos. Além disso, os executivos deverão apoiar e difundir essa nova idéia. Tudo isso requer tempo e dinheiro.

2.3.7.2 Outsourcing

Outsourcing é a prática de contratar uma organização externa para desenvolver um sistema, ao invés de desenvolver na sua própria sede (*in-house*) [McC 96]. As organizações que utilizam *outsourcing* podem se especializar em uma determinada área, ter mais desenvolvedores para trabalhar em um determinado projeto, e ter uma grande biblioteca de código reutilizável. A combinação destes fatores pode resultar numa significativa redução no tempo necessário para desenvolver um produto. Muitas vezes os custos de desenvolvimento também podem diminuir. Os maiores desafios deste tipo de ambiente de desenvolvimento são a necessidade de se avaliar muito bem a real necessidade do *outsourcing* e como ele será estruturado. Os desafios estão totalmente ligados aos riscos que uma estratégia destas envolve. Entre eles, pode-se citar [McC 96]:

- Transferência da experiência e da especialidade em um determinado assunto para fora da organização;
- Perda do controle sobre os desenvolvimentos futuros;
- Deve haver um compromisso com a informação considerada confidencial;
- Perda de visibilidade de progresso e controle dos projetos.

Para minimizar os riscos existentes é necessário um gerenciamento bastante eficaz. Como regra [McC 96], *outsourcing* necessita de muito mais gerenciamento do que um desenvolvimento *in-house*, e geralmente é um grande desafio para as organizações.

Duas opções de *outsourcing* que vem se tornando bastante popular ao longo dos últimos anos é o *offshore outsourcing* (contratação de uma organização externa, localizada em um outro país) e o *offshore insourcing* (contratação de uma subsidiária da própria organização, também localizada em um outro país). Organizações *offshore* são empresas que estão localizadas em algum outro país, e que oferecem custos mais baixos de desenvolvimento. Além disso, também oferecem uma qualidade que é, no mínimo, equivalente à qualidade das organizações localizadas no próprio país [McC 96].

Em suma, uma organização pode transferir todo o seu desenvolvimento para fora da empresa ou até mesmo para fora do seu país, desde que o local seja bem selecionado e todos os aspectos sejam analisados e bem planejados. Este tipo de solução requer um adequado processo de gestão.

2.3.7.3 Ambientes fisicamente distribuídos

Ao optar pelo *outsourcing* ou *insourcing*, uma empresa não configura um ambiente de desenvolvimento de software fisicamente distribuído. Isto ocorre, pois a empresa externa que foi contratada pode exercer suas atividades na própria sede da empresa contratante. A distribuição do processo de desenvolvimento de software ocorre somente quando parte dos envolvidos no processo estão fisicamente distantes. Este talvez seja um dos mais desafiadores contextos que o atual ambiente de negócios apresenta. Muitas empresas estão distribuindo seu processo de desenvolvimento de software em países como Índia, Irlanda, Brasil, China, Singapura, entre outros [CAR 99], [KAR 98]. Muitas vezes este processo se dá dentro de um mesmo país, em regiões com incentivos fiscais ou de concentração de massa crítica em determinadas áreas.

As empresas buscam vantagens competitivas em termos de custos, qualidade ou flexibilidade na área de desenvolvimento de sistemas [PRI 02c], além de ganhos de produtividade e diluição de riscos [McC 96]. Neste caso, ao optar por instanciar um ambiente de desenvolvimento distante fisicamente da sua sede, uma organização começa a encarar diversos desafios de adaptação, diferenças culturais, planejamento do

trabalho, treinamento da equipe, entre outros. Sendo assim, os desafios existentes no *outsourcing* ganham proporções maiores, mas o resultado pode ser muito significativo. E desenvolver ambientes tecnológicos, modelos e ferramentas para atuar neste tipo de ambiente é um desafio cada vez mais importante, tanto para a teoria na área de ES, como para as empresas que enfrentam as dificuldades criadas por este ambiente.

2.3.8 Análise Crítica

O processo de desenvolvimento de software tem avançado muito nos últimos anos, mas muitos desafios devem ser vencidos e muitas barreiras precisam ser quebradas. Ainda, além dos desafios citados anteriormente, identificou-se na literatura capacitação de pessoal, aprendizagem organizacional, produtividade, motivação e condução do processo de mudança.

Identificou-se uma convergência na direção de três desafios identificados na literatura: o planejamento no contexto do processo de desenvolvimento de software [AUD 01], [REP 98], os ambientes de desenvolvimento fisicamente distribuídos [McC 96], [PRI 02c] e a gestão do conhecimento [DES 02], [NON 94]. A evolução dos ambientes fisicamente distribuídos mostra uma nova tendência em desenvolvimento de software no âmbito mundial. Até pouco tempo atrás não se desenvolviam projetos com equipes dispersas globalmente da forma que tem ocorrido hoje.

Além disso, o desenvolvimento de software possui diversos desafios relacionados ao planejamento dos projetos e a gestão do conhecimento. Em um contexto de ambiente fisicamente distribuído estes desafios se acentuam. Um dos fatores determinantes para o sucesso do desenvolvimento de software em uma empresa é a capacidade de implantar um processo de planejamento integrado com o processo de desenvolvimento de software e possibilitar formas de criar, armazenar e disseminar o conhecimento gerado neste contexto.

2.4 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE (DDS)

Na última década observou-se um grande investimento na conversão de mercados nacionais em mercados globais, criando novas formas de competição e colaboração [HER 01a]. Entretanto, o mercado global de software vinha passando por diversas crises. Por um lado, um grande número de falhas em projetos. De outro, uma crescente demanda, atingida pela escassez de recursos capacitados. Nesse ambiente, muitas organizações encontraram no DDS uma alternativa, experimentando o desenvolvimento em locais remotos. Atualmente um grande número de organizações

realiza desenvolvimento distribuído de software, e o assunto está cada vez mais presente em congressos e workshops internacionais [DAM 02c], [DAM 03a], além de livros específicos sobre o tema [CAR 99], [KAR 98].

O DDS, tendo em vista sua recente aplicação e grande abrangência, possui uma diversidade de conceitos que auxiliam sua caracterização. Dentro do contexto de DDS podem ser aplicados conceitos de equipes globais [MAR 01] e organizações virtuais [KAR 98], por exemplo, ampliando a extensão do conhecimento envolvido. Segundo [MAR 01], uma equipe global refere-se a um grupo de pessoas de diferentes nacionalidades que trabalham unidas em um projeto comum, através de culturas e fusos-horários distintos, por um extenso período de tempo. Com relação às organizações virtuais, [KAR 98] as define como entidades caracterizadas por realizar partes de um projeto em locais distintos, comportando-se como se estivesse no mesmo local.

Quando a distância física entre os atores em um ambiente de DDS envolve mais de um país, [KAR 98] define uma instância do DDS chamada de desenvolvimento global de software (*Global Software Development* – GSD). O GSD é instanciado através de equipes globais de desenvolvimento de software (*Global Software Teams*), que [CAR 99] define como sendo um grupo de pessoas distribuída em dois ou mais países, colaborando ativamente em um projeto comum.

Neste sentido, o DDS tem sido caracterizado principalmente pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, mas estão localizados em cidades ou países diferentes, distantes temporal e fisicamente. As figuras 2.5 e 2.6 ilustram a mudança para este novo cenário.

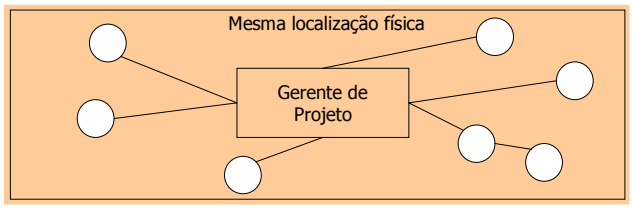

Figura 2.5 – Desenvolvimento centralizado

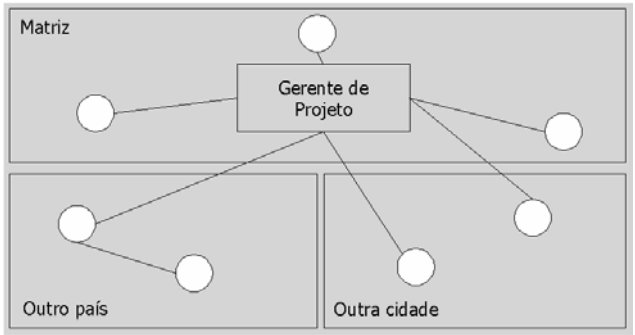

Figura 2.6 – Desenvolvimento distribuído

Apesar de muitas vezes este processo ocorrer em um mesmo país, em regiões com incentivos fiscais ou de concentração de massa crítica em determinadas áreas, algumas empresas, visando maiores vantagens competitivas, buscam soluções externas até mesmo em outros países. Muitas empresas estão distribuindo seu processo de desenvolvimento de software em diversos países. Neste contexto, surgem os conceitos de *outsourcing*, *offshore outsourcing* e *offshore insourcing*, o que potencializa os problemas e os desafios existentes no DDS.

Diversos fatores têm contribuído para isto, entre eles:

- A necessidade de recursos globais para serem utilizados a qualquer hora;
- As vantagens de estar perto do mercado local, incluindo o conhecimento dos clientes e as condições locais;
- A rápida formação de organizações e equipes virtuais para explorar as oportunidades de mercado;
- A grande pressão para o desenvolvimento *time-to-market*, utilizando as vantagens proporcionadas pelo fuso horário diferente, no desenvolvimento conhecido como *follow-the-sun*, ou seja, o desenvolvimento quase que contínuo (24 horas contínuas, contando com as equipes fisicamente distantes em países com fusos horários diferentes).

Estas mudanças estão causando um grande impacto não apenas no mercado propriamente dito, mas na maneira como os produtos de software estão sendo criados, modelados, construídos, testados e entregues para os clientes. Além disso, [HER 01a] destaca que trabalhar com DDS é um dos maiores desafios que o atual ambiente de negócios apresenta do ponto de vista do processo de desenvolvimento de software.

Várias ferramentas e ambientes têm sido desenvolvidos ao longo dos últimos anos para ajudar no controle e coordenação das equipes de desenvolvimento que estão inseridas neste tipo de ambiente. Muitas destas ferramentas estão focadas no suporte aos procedimentos de comunicação formal tais como elaboração de documentos, processos automatizados e canais de comunicação não interativos [ALT 98], [BIU 02].

Alguns estudos [CAR 01a], [KIE 03], [VOG 01] observam uma dificuldade de implantar, executar e monitorar projetos em ambientes de DDS devido a fatores não-técnicos tais como fatores sociais, culturais, comportamentais, psicológicos, lingüísticos e políticos (confiança, diferenças culturais, idioma, entre outros). Enquanto isso, outros estudos [ALT 98], [BIU 02], [HER 99], [PRI 03b], observam estas mesmas dificuldades devido a fatores técnicos (processo de desenvolvimento de software, gerência de projeto, complexidade e tamanho de projetos, tecnologia de comunicação disponível, entre

outros). Mas são poucos os trabalhos [EVA 00], [KAR 98], [CAR 99] que têm abordado ambos os conjuntos de fatores em um modelo híbrido, procurando relações entre eles.

2.4.1 Razões que levam ao DDS

O desenvolvimento de software costumava ser realizado apenas por pessoas com altíssimo grau de especialização, trabalhando em centros de processamento de dados de países avançados [CAR 99]. Hoje, entretanto, ele ocorre cada vez mais de forma distribuída. Em 2000, 203 das empresas americanas na *Fortune 500*[3] realizavam *offshore outsourcing* [CAR 01a]. Neste sentido, existem diversas razões para a aplicação do DDS. Essas razões, ou um subconjunto delas motivam um crescente número de organizações a desenvolverem software de forma distribuída. Nesta subseção são apresentados em detalhe alguns fatores que motivam o uso de DDS nas organizações. A figura 2.7 identifica as principais razões envolvidas.

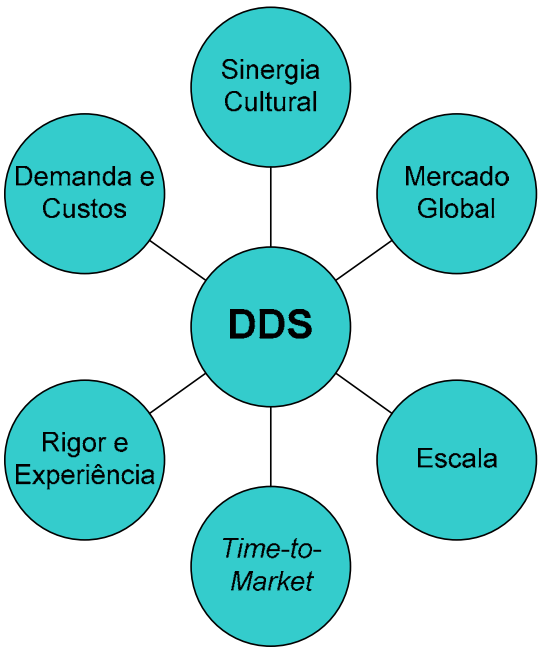

Figura 2.7 – Principais razões envolvidas no DDS

Demanda e Custos: a demanda por serviços de software tem superado historicamente a disponibilidade de pessoas que os realizam (Figura 2.8). Como conseqüência, o custo dos profissionais cresceu, conforme as empresas competiam por suas contratações [KAR 98]. No mercado americano, responsável por grande parte da demanda, as cotas de vistos de trabalho temporário para área de tecnologia se esgotavam antes do final do ano fiscal, gerando uma maior escassez de profissionais [CAR 99]. Dessa forma, com um número de funcionários insuficiente e custos de

[3] *Fortune* 500 é o ranking anual das 500 maiores empresas nos Estados Unidos, promovido pela revista *Fortune* (http://www.fortune.com/fortune/).

contratação altíssimos, a disponibilidade de recursos equivalentes em outras localidades a um custo mais baixo tornou-se um grande atrativo para as empresas de software, motivando o DDS.

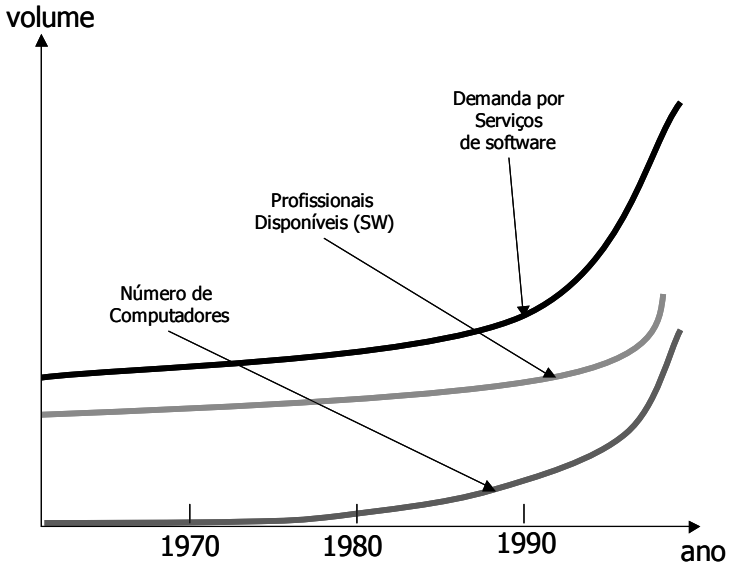

Figura 2.8 – Demanda por software X profissionais disponíveis [KAR 98]

Time-to-market: as pressões para reduzir o tempo necessário para colocar um produto em mercado (*time-to-market*) também auxiliam no crescimento do DDS [HER 01a]. A possibilidade de desenvolvimento *follow-the-sun* é um grande atrativo para empresas que visam reduzir o *time-to-market*. No desenvolvimento *follow-the-sun* equipes são distribuídas ao redor do globo, de forma que há sempre pelo menos uma equipe disponível realizando o trabalho [MAR 01]. Dessa forma, a distância geográfica transforma-se em uma vantagem competitiva, permitindo trabalhar 24 horas por dia [CAR 99].

Mercado e presença global: outro fator importante é o crescente mercado de software externo à América do Norte e Europa. Conforme os custos se reduzem e o poder computacional aumenta, o mercado global de informática cresce, apresentando grande demanda por soluções de software [KAR 98]. Em conjunto, para melhor satisfazer o mercado consumidor, a presença das corporações se torna necessária para venda, projeto e manutenção do software. Dessa forma, muitas empresas optam pelo DDS para atingir o mercado global e ficarem próximas a seus consumidores. Além disso, obtém-se o diferencial de se tornar uma empresa global, um bom atrativo de marketing [CAR 99].

Rigor e experiência no desenvolvimento: equipes de desenvolvimento co-localizadas tendem a utilizar mais de mecanismos informais, descuidando no uso de metodologias formais de desenvolvimento e práticas de qualidade. Equipes de DDS, por outro lado, na tentativa de melhorar a comunicação entre os locais de desenvolvimento,

tendem a melhorar significativamente a documentação utilizada [CAR 99]. Além disso, em muitos casos, determinados locais desenvolvem experiência e habilidade em áreas pouco difundidas em outros pontos da organização. Nestes casos, a experiência de um local específico pode ser um diferencial para que o desenvolvimento seja realizado nele.

Sinergia cultural: a diversidade amplia a criatividade e a inspiração na organização de desenvolvimento de software. Uma equipe global cria uma sinergia cultural, encontrando novas formas de resolver problemas, projetar produtos, ou pensar sobre os processos [CAR 99]. Além disso, a sinergia cultural, ligada à demanda e aos desafios envolvidos, amplia a capacidade de aprendizado da organização [MAR 01].

Escala: centros de desenvolvimento de software conforme aumentam, chegam a um ponto onde ficam muito grandes e difíceis de gerenciar. Com isso, torna-se necessário distribuir o desenvolvimento para atender a demanda necessária [CAR 99].

2.5 TRABALHOS RELACIONADOS

O DDS apresenta um grande impacto na forma como os produtos são concebidos, desenvolvidos, testados e entregues aos clientes [HER 01a]. Assim, a estrutura necessária para o suporte desse tipo de desenvolvimento se diferencia da utilizada em ambientes centralizados. Diferentes características e tecnologias se fazem necessárias, crescendo a importância de alguns detalhes antes não percebidos. Atualmente encontram-se disponíveis na literatura da área alguns estudos que propõem modelos de referência para o DDS, abordando fatores técnicos e não-técnicos. A seguir apresentam-se três abordagens existentes [KAR 98], [CAR 99], [EVA 00].

2.5.1 A abordagem de [KAR 98]

O trabalho de [KAR 98] aborda o DDS seguindo, segundo o autor, o ciclo de vida tradicional de um projeto de desenvolvimento de software (Visão e Escopo, Requisitos, Modelagem, Implementação, Teste, Entrega e Manutenção). O autor propõe um modelo para desenvolver projetos de DDS abrangendo um conjunto de atividades que devem ocorrer ao longo do ciclo de vida de um projeto e as características essenciais de cada uma. O modelo considera atividades desde a estratégia a ser adotada para atuar em DDS (empresas globais, parcerias estratégicas, entre outros) até a manutenção do software desenvolvido, mas não se aprofunda na análise de como implementar cada uma. Discutem-se os conceitos envolvidos, sugestões de atividades e dificuldades envolvidas. A figura 2.9 ilustra o modelo proposto:

Figura 2.9 – O modelo proposto por [KAR 98]

A seguir apresenta-se detalhadamente cada atividade proposta no modelo:

1. Alinhar o negócio: segundo autor, a primeira atividade necessária para desenvolver projetos de DDS é a identificação e alinhamento do tipo de estrutura que será utilizada. Isto envolve definir se será uma interação com outras empresas (parcerias estratégicas), ou a criação de unidades da empresa em outras localidades (empresas globais). Isto deve ser feito levando-se em consideração o negócio da organização e seus objetivos.

2. Identificar a equipe distribuída: uma vez definida a forma de interação em projetos de DDS, a próxima atividade diz respeito à identificação da estrutura da equipe que irá atuar nos projetos distribuídos de desenvolvimento de software. Devem ser definidos os integrantes da equipe, seus respectivos papéis e responsabilidades. A formação da equipe deve considerar aspectos tais como a aquisição de confiança, diferenças culturais, relacionamento, entre outros.

3. Identificar as tecnologias: projetos de DDS envolvem uma grande quantidade de comunicação. Conseqüentemente, isto necessita de um suporte tecnológico considerável. Por isto, esta atividade prevê a identificação da infra-estrutura disponível para os membros da equipe se comunicarem (vídeo conferência, e-mail, *groupware*[4], entre outros), considerando a distância física existente e a característica da equipe distribuída.

4. Definir o contrato: independente da forma de interação sendo realizada (parcerias estratégicas, empresas globais ou outras), deve-se sempre criar um documento (conhecido como *Statement of Work*) para identificar as responsabilidades e expectativas entre o cliente e a equipe de projeto no desenvolvimento do software. Isto é crítico à medida que será feito de forma dispersa e, da mesma forma que em projetos co-localizados, um contrato é o documento que define o escopo do que deve ser feito.

[4] Tecnologia de colaboração freqüentemente utilizada para suportar a interação de pessoas fisicamente dispersas.

5. Dividir o trabalho: uma vez identificada equipe, tecnologias e definido o contrato, o autor propõe uma atividade onde o esforço de trabalho deve ser divivido entre os membros da equipe. Isto deve ser realizado através de alguns critérios tais como nível de experiência, modularidade do projeto, etc., e pode ter como base, a arquitetura do produto que será desenvolvido.

6. Identificar ferramentas e métodos: a identificação de ferramentas e métodos diz respeito principalmente aos recursos técnicos que serão utilizados nas atividades de modelagem e implementação do projeto. Esta definição deve considerar o nível de dispersão das equipes (o quão distantes elas estão), o processo de desenvolvimento existente e outros fatores que a equipe considerar relevante.

7. Estabelecer responsáveis por SCM: a gerência de configuração de software (SCM – *Software Configuration Management*) tem como objetivo controlar modificações nos artefatos, dando suporte ao controle de versões. Sendo assim, [KAR 98] sugere a existência de um grupo responsável pelo controle de configuração e versões do sistema. Por este motivo, esta atividade visa identificar os membros deste grupo, bem como as ferramentas que eles utilizarão e a freqüência necessária de reuniões para discutir o andamento do trabalho.

8. Identificar e gerenciar riscos: gerenciar riscos faz parte de qualquer projeto. Segundo [KAR 98], os riscos em projetos de DDS tendem a estar mais centrados em aspectos não tão visíveis. Devem existir atividades de identificação de riscos e planejamento de estratégias de mitigação. De acordo com a figura 2.9, esta atividade deve ocorrer em todas as fases do desenvolvimento, exceto entrega e manutenção. De acordo com [KAR 98], podem existir três categorias de riscos em projetos de DDS: organizacional, técnico e de comunicação. Além disso, podem existir riscos presentes em mais de uma categoria, e estes devem estar no topo da lista de prioridades.

9. Controlar a documentação: em projeto de DDS, uma documentação pobre pode ser uma das causas de ineficiência na colaboração. A resistência à documentação entre as equipes desenvolvimento é conhecida, mas deve-se identificar um método de controle de toda a documentação do projeto, organizando-a e permitindo um fácil acesso à mesma. A existência de uma boa documentação no DDS é muito importante, pois pode evitar ambigüidades e facilitar futuras manutenções.

10. Desenvolver plano e casos de teste: segundo [KAR 98], qualquer projeto distribuído necessita de pelo menos dois artefatos relacionados ao teste. O primeiro é um plano de teste, documentando as estratégias, métodos e ambientes que serão utilizados. O segundo é um conjunto de casos de teste, identificando, durante a modelagem e a implementação, funcionalidades que deverão ser testadas. Além disso,

durante a fase de Teste, deve-se incorporar atividades de validação e verificação do sistema desenvolvido.

11. Criar matriz de rastreabilidade: uma matriz de rastreabilidade é um artefato que identifica as funcionalidades do projeto e os módulos que as implementam. Segundo [KAR 98], deve-se criar esta matriz durante a fase de Requisitos e alterar durante a modelagem e a implementação, rastreando as funcionalidades e identificando possíveis inconsistências.

12. Criar matriz de versão de módulos: uma matriz de versão de módulos é um artefato que identifica qual versão de um módulo foi utilizada na compilação do código de um projeto. Este artefato é essencial principalmente para a coordenação das atividades e divisão do trabalho entre os membros da equipe do projeto.

13. Criar grupo de manutenção: apesar do DDS ser uma tarefa desafiadora, alguns desafios podem continuar na fase de manutenção. O modelo de [KAR 98] sugere a criação de um grupo responsável por revisar solicitações de alterações após o produto ser entregue ao cliente.

14. Controlar a qualidade do software: segundo [KAR 98], a qualidade deve ser parte obrigatória de qualquer projeto de DDS. A qualidade é influenciada pelo processo de desenvolvimento e pela forma como a solução foi modelada (a estratégia de modelagem influencia principalmente no teste, na manutenção e na confiabilidade). Além disso, a qualidade deve ser medida e deve estar relacionada com a satisfação do cliente. Por tudo isto, devem existir atividades que melhoram a qualidade do software a ser desenvolvido, tais como revisões de modelagem, inspeções de código e teste.

15. Gerenciar a propriedade intelectual: ao lidar com o desenvolvimento de software em um mercado competitivo, as organizações buscam muitas vezes desenvolver produtos com características únicas. Por isso, determinados projetos podem conter idéias geradas no decorrer do desenvolvimento que necessitam de meios de proteção da propriedade intelectual. Neste sentido, [KAR 98] prevê uma atividade onde busca-se a devida proteção, levando-se em consideração leis e restrições do local onde o projeto foi desenvolvido (alguns locais fisicamente dispersos podem ter leis muitas vezes desconhecidas pelas organizações).

2.5.2 A abordagem de [CAR 99]

Em seu trabalho, [CAR 99] aborda a formação de equipes globais de desenvolvimento de software e os principais fatores que devem ser considerados ao montar uma equipe para um projeto distribuído. O trabalho do autor sugere a existência de cinco categorias de fatores que podem levar uma equipe distribuída ao fracasso

(comunicação ineficiente, falta de coordenação, dispersão geográfica, perda do espírito de equipe e diferenças culturais), as quais são chamadas de forças centrífugas. Além disso, o autor sugere a existência de seis fatores que podem levar a equipe ao sucesso (infra-estrutura de comunicação, arquitetura do produto, construção de uma equipe, metodologia de desenvolvimento, tecnologia de colaboração e técnicas de gerência), os quais são chamados de forças centrípetas. A Figura 2.10 apresenta o modelo proposto.

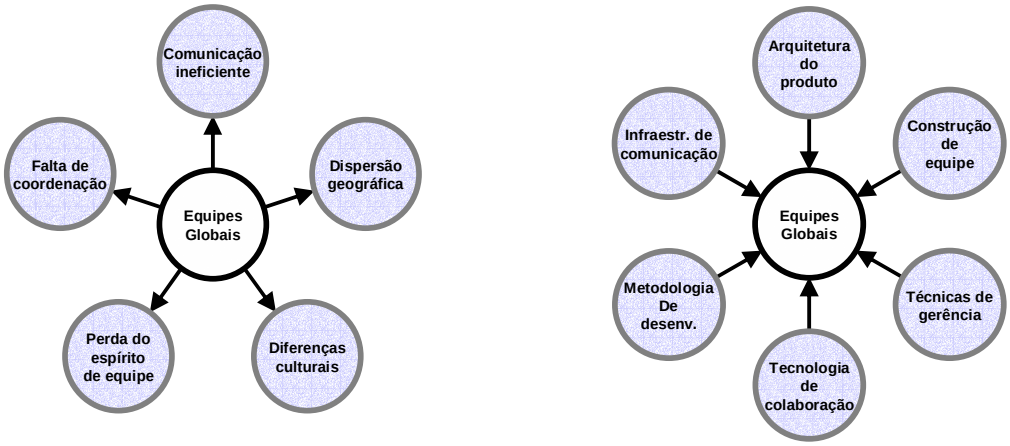

Figura 2.10 – Forças centrífugas e forças centrípetas de equipes globais [CAR 99]

A seguir apresenta-se detalhadamente cada uma das forças centrífugas e centrípetas definidas por [CAR 99].

2.5.2.1 Forças centrífugas

Diferenças culturais: o gerenciamento da diversidade cultural é fundamental para efetividade de uma equipe distribuída, principalmente em âmbito global. Segundo [CAR 99], as culturas diferem em muitas dimensões críticas, como a necessidade de estrutura, atitudes com relação à hierarquia, senso de tempo e estilos de comunicação. Além disso, reflete diretamente em questões como formas de liderança, de comunicação e de resolução de problemas.

Dispersão: segundo [CAR 99] existem dois tipos de dispersão. O primeiro diz respeito à distância geográfica, baseada na distância física entre os atores envolvidos em projetos distribuídos. O segundo diz respeito à dispersão temporal, onde membros de uma equipe estão dispersos no tempo pela diferença nos horários de trabalho, fusos horários e/ou ritmos de trabalho que diminuam o tempo disponível para interação síncrona (isto é, ao mesmo tempo). A dispersão temporal também afeta a comunicação devido aos estados físicos e mentais dos participantes. Elementos em um local podem estar iniciando o dia, enquanto outros estão no final do expediente.

Falta de coordenação: equipes distribuídas apresentam dificuldades nos mecanismos de coordenação (integração das tarefas e unidades organizacionais de forma que o esforço da equipe contribua para o objetivo geral) e controle (o processo de

adesão a metas, políticas e padrões), principalmente os informais. As dificuldades e os desafios são ampliados devido aos problemas de cultura, língua e tecnologia. O grau de dependência das tarefas exerce um papel fundamental na coordenação. O gerenciamento da configuração de software (SCM) também apresenta novos desafios. Controlar modificações nos artefatos em cada uma das localidades e coordenar o processo de modificação com o processo de desenvolvimento de todo o produto pode ser bastante complexo. A manutenção da consistência de padrões e formato de documentação também é mais difícil em ambientes distribuídos.

Comunicação Ineficiente: a dispersão tem um impacto direto em todas as formas de comunicação, principalmente na redução da comunicação informal. As pessoas deixam de se comunicar devido às dificuldades impostas, o número de reuniões aumenta e a complexidade de coordená-las torna-se maior. Segundo [CAR 99], comunicação clara e efetiva é essencial para o sucesso de equipes distribuídas, mas em muitos casos é necessário comunicar-se indiretamente (devido à distância temporal), prejudicando a riqueza de contexto. Dados os diferentes estágios do ciclo de desenvolvimento de software, algumas tarefas necessitam de comunicação mais rica que outras. De forma geral, qualquer tarefa que necessite de intensa cooperação requer mais comunicação, e quanto mais rica, melhor. Determinadas tarefas exigem muito cuidado na escolha do meio de comunicação a ser utilizado. Alguns dos impactos da dispersão na comunicação são a falta de comprometimento e o desconforto ao utilizar alguns meios [LAY 00].

Perda do espírito de equipe: equipes são unidades sociais frágeis que podem facilmente ser quebradas [CAR 99]. Quando apresentadas a dificuldades como distância e diferença culturais, o espírito de equipe geralmente desaparece. O espírito de equipe é o efeito sinérgico que torna a equipe uma unidade coesa. Em ambientes distribuídos e multiculturais é difícil afirmar que todos os membros entendem o que é uma equipe. A confiança, fundamental para o espírito de equipes, torna-se de difícil manutenção e o tamanho do grupo é importante para assegurar que a comunicação entre todos ocorra de forma efetiva.

2.5.2.2 Forças centrípetas

Infra-estrutura de comunicação: segundo o autor, ambientes de DDS necessitam de conexões confiáveis, de alta velocidade para todas as formas de comunicação e dados. Embora em alguns países a estrutura de telecomunicações não permita que se consiga nada além de linhas telefônicas comuns, a maioria das localidades dispõe tecnologias que permitem a transmissão confiável de voz e dados com boa performance.

Arquitetura do produto: segundo [CAR 99], a arquitetura do software é um fator determinante na efetividade e redução das dificuldades do DDS e deve se basear no princípio da modularidade, considerada a principal forma de resolver e alocar tarefas complexas de forma distribuída. Um projeto modular reduz a complexidade e permite um desenvolvimento com uma menor interdependência entre os locais, podendo reduzir custos adicionais de coordenação.

Tecnologia de colaboração: uma das forças centrípetas que [CAR 99] define como fundamental é a existência de tecnologias de colaboração. Estas são utilizadas para ampliar a comunicação informal e possibilitar novas formas de comunicação formal entre as equipes dispersas. Podem ser divididas tecnologias genéricas de colaboração (correio eletrônico, correio de voz, entre outros) e tecnologias de colaboração para suporte das atividades de engenharia de software (*Collaborative Tecnology to Support Software Engineering* – CT-SE). Esta última compreende ferramentas de gerência de configuração de software, gerência de projetos, ferramentas CASE, entre outros. Segundo [CAR 99], a ferramenta de CT-SE mais importante para equipes dispersas é a de gerência de configuração de software, cuja função principal é controlar as múltiplas peças de um projeto distribuído.

Técnicas de gerência de projetos: a gerência de projetos de DDS exige uma adaptação de algumas técnicas utilizadas em projetos co-localizados, de forma a suportar e reduzir as dificuldades impostas pela dispersão das equipes. A estrutura utilizada em equipes co-localizadas pode perder a efetividade em ambientes distribuídos e necessita de adaptações. Uma equipe de DDS deve possuir uma estrutura flexível para suportar a distribuição de tarefas e a tomada de decisões de forma efetiva. Os principais aspectos enfocados por [CAR 99] são a gerência de conflitos, utilização de métricas, formas de reconhecimento e bonificação e escolha de um gerente com perfil para atuar em projetos distribuídos.

Processo de desenvolvimento: a existência de um processo de desenvolvimento comum às equipes distribuídas é fundamental segundo [CAR 99]. Quando equipes distribuem o processo em diversas localidades, a falta de sincronização pode se tornar crítica. Dessa forma, um processo fornece um conjunto comum de expectativas aos elementos envolvidos, impondo rigor à equipe.

Formação de Equipes: equipes de DDS necessitam se relacionar, ter mecanismos de comunicação eficientes e possuir uma visão compartilhada. A interação efetiva entre membros de uma equipe tem a confiança como base, que é fundamental quando algumas pessoas dependem de outras para realização de seus objetivos. O conhecimento dos papéis de cada um e da estrutura da equipe por todos os envolvidos é necessário para que o contato com os responsáveis por cada tarefa seja facilitado.

Segundo [CAR 99], o DDS, principalmente em âmbito global, usualmente envolve equipes de localidades cujo idioma nativo é diferente. As dificuldades de idioma são sutis e podem causar diversos problemas. Profissionais expostos a um idioma diferente do nativo usualmente lêem mais lentamente, não percebem algumas idéias, e encontram dificuldades de concentração e expressão.

2.5.3 A abordagem de [EVA 00]

A pesquisa realizada por [EVA 00] propõe a existência de algumas dimensões no contexto de equipes de projetos distribuídos. Estas dimensões (Figura 2.11) auxiliam no entendimento dos problemas, vantagens e desvantagens deste tipo de ambiente e afetam diretamente na performance destes projetos. Cabe salientar que o trabalho de [EVA 00] é aplicável a qualquer tipo de projeto, não apenas a projetos de desenvolvimento de software.

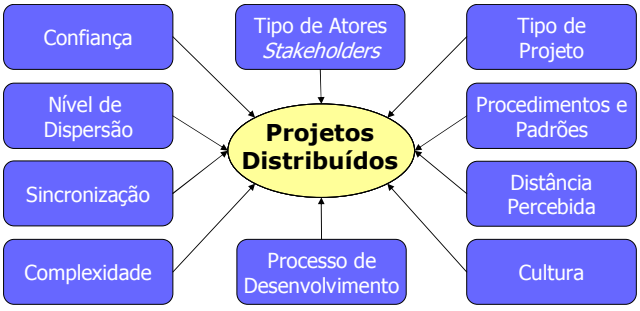

Figura 2.11 – As dimensões do DDS segundo [EVA 00]

A seguir apresenta-se detalhadamente cada dimensão definida pelo autor.

Confiança: a confiança, fundamental para manter o espírito de equipes, pode-se tornar de difícil manutenção no DDS. A construção desse tipo de relacionamento é muitas vezes impedida pela distância. Por isso, confiar nas equipes geograficamente distribuídas é essencial para desenvolver um projeto de DDS.

Distância Percebida: existem diversas possibilidades entre a habilidade de encontrar uma pessoa face a face freqüentemente (fisicamente próximo), e de nunca ser capaz de encontrá-la (fisicamente distante). Esta dimensão considera todos os participantes de um projeto, mas desconsidera papéis, que são relevantes apenas em níveis de dispersão. A distância percebida afeta as atividades de coordenação.

Nível de Dispersão: a distância física entre os membros de um determinado grupo de *stakeholders*[5] (equipe de projeto, por exemplo) é chamado de nível de dispersão. O trabalho de [PRI 03a], [PRI 03c] explora algumas possibilidades de

[5] Um *stakeholder* é qualquer pessoa ou representante de uma organização que possua um *stake* – um grande interesse – no resultado de um projeto [PRE 01].

caracterização do nível de dispersão considerando a existência de três atores no desenvolvimento de um projeto (equipe de projeto, clientes e usuários). Segundo [EVA 00], quanto maior o nível de dispersão, maior é a dificuldade de monitorar o comportamento de diferentes grupos.

Sincronização: esta dimensão diz respeito à capacidade das pessoas trabalharem no mesmo projeto de forma concorrente. Uma das formas que possibilitam uma total sincronização é ter todos *stakeholders* no mesmo fuso-horário ou trabalhando no mesmo momento em um determinado projeto.

Tipos de Atores (*stakeholders*): diferentes tipos de grupos de *stakeholders* existem em cada projeto, cada um com diferentes percepções sobre o projeto. Quanto maior o número de *stakeholders* envolvidos, maior pode ser a distribuição do projeto.

Complexidade: o nível de complexidade de um projeto pode afetar a performance de projetos distribuídos. Existem diversos fatores de complexidade, entre eles o nível de tecnologia utilizada em um projeto e o nível de definição dos objetivos e escopo do mesmo.

Cultura: cultura é um fator multidimensional que afeta a performance de projetos distribuídos de diversas maneiras. Podem ser considerados aspectos da cultura nacional, cultura organizacional, entre outros. Estes aspectos afetam muitos dos comportamentos de membros de equipes de projetos que pertencem a diversas culturas.

Tipo de Projeto: o tipo de projeto (manufatura, hardware, software) pode afetar a forma como o mesmo será gerenciado. Além disso, as técnicas de gerência de projeto devem permitir o gerenciamento não apenas de projetos desenvolvidos em locais fisicamente dispersos, mas também projetos que envolvam um grande número de *stakeholders*.

Processo de Desenvolvimento: no caso de um projeto de desenvolvimento de software, este deve considerar um processo bem definido e conhecido pelas equipes geograficamente distribuídas. Não é interessante que um gerente coordene um projeto tendo um processo sendo executado por cada equipe, utilizando metodologias diferentes. Existem diversas metodologias disponíveis, tais como Orientação a Objeto e Programação Estruturada. Cada metodologia utiliza-se de algumas técnicas para especificar e modelar sistemas, e dependendo da metodologia existente a tarefa de gerência do projeto pode ser complexa. Considerando o processo de desenvolvimento de software, [EVA 00] sugere que verificação do nível de maturidade dos processos de software dos locais dispersos de acordo com o modelo CMM. Isto pode facilitar as tarefas de coordenação e o entendimento das variações em questões referentes à comunicação, qualidade e planos de um projeto de DDS.

Existência de Procedimentos e Padrões: segundo [EVA 00], a idéia desta dimensão é criar procedimentos e padrões (técnicas de estimativa, padrões de comunicação, padrões de implementação, entre outros) e torná-los parte da cultura da organização. Com isto, busca-se principalmente um padrão entre os grupos geograficamente distribuídos no âmbito organizacional.

2.5.4 Análise Crítica

As abordagens descritas anteriormente destacam os projetos distribuídos sob três perspectivas diferentes. Enquanto que [KAR 98] propõe um modelo específico para projetos de DDS, tendo como guia um processo tradicional de desenvolvimento de software, [CAR 99] aborda a questão da formação de equipes globais de software, suas características e os principais desafios inerentes a formação de equipes de sucesso. Por sua vez, [EVA 00] também aborda a questão da formação de equipes, mas relacionando com equipes distribuídas para qualquer tipo de projeto, não necessariamente projetos de DDS. Todas as abordagens descritas, inclusive [EVA 00], apresentam características essenciais do DDS. Diversos outros autores abordam estas características, ampliando os conceitos relacionados e aprofundando o estudo em pesquisas mais específicas. A seguir (Quadro 2.2), listam-se as características apresentadas nas abordagens descritas anteriormente, identificando outros autores que realizam estudos sobre cada uma.

Quadro 2.2 – Complementando as abordagens apresentadas

Características	Abordagem	Trabalhos relacionados
Alinhamento e planejamento do negócio	[KAR 98]	[CAR 02], [DEL 03], [HER 01a], [PRI 02d]
Arquitetura do produto e alocação de tarefa	[CAR 99] [KAR 98]	[BAT 01], [CAR 01a], [HER 01a], [RAP 01]
Complexidade do projeto	[EVA 00]	[ESP 02]
Comunicação	[KAR 98]	[HER 99], [HER 00], [HER 01a], [HER 03], [KIE 03], [LAY 00], [MAR 01], [PAA 03]
Confiança	[EVA 00]	[JAR 98], [KIE 03], [KRA 99], [MAR 01], [PYY 03], [BER 00]
Construção de equipe e espírito de equipe	[CAR 99] [KAR 98]	[BAT 01], [HER 03], [KIE 03], [KOB 03], [KRA 99], [MAR 01], [PAA 03], [PYY 03], [SAB 99]
Contrato	[KAR 98]	[KHA 03]
Coordenação, controle e sincronização	[CAR 99] [EVA 00]	[DAM 03b], [ESP 03], [GRI 99], [HER 01a], [KOB 03], [MAR 01]
Diferenças culturais	[CAR 99] [EVA 00] [KAR 98]	[FAV 01], [HER 01a], [KIE 03], [MAR 01], [VOG 01],
Dispersão geográfica e temporal	[CAR 99]	[ESP 03], [HER 00], [HER 03], [KIE 03], [LAY 00], [MAR 01],

	[EVA 00]	[PAA 03], [PRI 03a]
Documentação	[KAR 98]	[HER 99], [HER 01a], [OPP 02]
Engenharia de requisitos	[KAR 98]	[DAM 02a], [DAM 02b], [DAM 03b], [LAN 03], [PRI 02c]
Gerência de riscos	[KAR 98]	[HER 01a], [OPP 02],
Infra-estrutura	[KAR 98]	[BIU 02], [KHA 03], [LAY 00]
Métricas de software	[CAR 99]	[IFP 02]
Procedimentos e padrões	[EVA 00]	[KIE 03], [LAY 00]
Processo de desenvolvimento	[CAR 99] [EVA 00]	[CAR 01a], [HER 99], [HER 01a], [MAI 99]
Propriedade Intelectual	[KAR 98]	[KHA 03]
Qualidade de software e testes	[KAR 98]	[BIA 03], [KHA 03], [LAN 02], [OPP 02]
Técnicas de gerência de projeto	[CAR 99]	[EVA 99], [EVA 03], [HER 01a], [KHA 03], [OPP 02]
Tecnologias de colaboração	[CAR 99] [KAR 98]	[ALT 98], [HER 02], [LAN 02] [LAN 03], [MAR 01], [SAR 02]
Tipo de atores	[EVA 00]	[PRI 03a]
Tipo de projeto	[EVA 00]	[BIA 03], [ESP 02], [EVA 99]

O quadro apresentado anteriormente não possui todos os fatores existentes em projetos de DDS. Procurou-se nesta análise identificar a existência de outros autores que abordam os fatores listados, considerando as três abordagens apresentadas. Procurou-se também manter a mesma classificação adotada nas abordagens (seções 2.5.1, 2.5.2 e 2.5.3). Neste sentido, pode existir na literatura trabalhos utilizando termos similares para os fatores, ou ainda categorias agrupando um conjunto destes, de acordo com a visão de outros autores.

Percebe-se no quadro 2.2 que diversos modelos e métodos têm sido desenvolvidos para a área de DDS, tendo por objetivo identificar as principais dificuldades e propor soluções para minimizá-las. Diferentes modelos buscam formular estratégias de atuação na área. Mas nem todos compreendem os diversos fatores envolvidos ou atuam apenas como suportes conceituais às atividades executadas nos projetos com equipes geograficamente distribuídas. Neste sentido, tem sido necessária a criação de processos formais para atuar neste cenário específico, procurando considerar os fatores que interferem no sucesso dos projetos de DDS.

2.6 CONSOLIDAÇÃO DA BASE TEÓRICA

A área de desenvolvimento de software está em um constante processo de amadurecimento [CAR 99]. Por este motivo, ao longo da revisão teórica realizada neste capítulo identificaram-se diversos problemas e desafios inerentes ao processo de

desenvolvimento de software e, mais especificamente, o desenvolvimento distribuído de software, foco deste estudo. O DDS, ao lidar com dispersão geográfica, dispersão temporal e diferenças culturais, amplia as dificuldades e desafios presentes no desenvolvimento tradicional.

Os trabalhos relacionados a este estudo identificados [CAR 99], [KAR 98], [EVA 00] contribuem no sentido de minimizar as dificuldades apontadas pela literatura e fazem frente aos desafios do ambiente de DDS. As dimensões propostas por [EVA 00] e as forças centrífugas e centrípetas propostas por [CAR 99] se concentram nos principais fatores presentes nas equipes em projetos distribuídos e projetos de desenvolvimento distribuído de software respectivamente. Além disso, [KAR 98] identifica um conjunto de atividades necessárias ao longo do ciclo de vida de um projeto distribuído. Adicionalmente, outros estudos ilustrados no quadro 2.2 aprofundam a análise destes fatores e acrescentam outros presentes não apenas nas equipes, mas em projetos de DDS como um todo, muitos deles relacionados com os problemas e desafios identificados na literatura de desenvolvimento de software como um todo.

Apesar de diversos estudos estarem sendo desenvolvidos na área de DDS nos últimos quatro anos, a carência de uma base teórica mais estável e consistente na área direcionou o estudo de modo a obter dados empíricos relacionados ao tema de estudo. Sendo assim, nos capítulos a seguir serão explorados os problemas e desafios encontrados na literatura, relacionando-os à nova classe de problemas que surge com o DDS.

3 METODOLOGIA DE PESQUISA

"Eu sempre digo que, quando você pode medir aquilo sobre o que está falando, e expressá-lo em números, significa que você sabe algo a respeito dele". Lord Kelvin, 1883.

Neste capítulo apresenta-se a metodologia de pesquisa utilizada no estudo. Na seção 3.1 apresenta-se o desenho de pesquisa e as suas etapas. Na seção 3.2 identificam-se os aspectos metodológicos do estudo. Na seção 3.3 apresenta-se a operacionalização das variáveis. Por último, na seção 3.4, é apresentada a base metodológica do estudo de caso.

Muito embora a ampla revisão teórica desenvolvida, não se tem conhecimento de que o problema apresentado tenha sido abordado sob a mesma perspectiva. Assim, esta pesquisa se caracteriza como um estudo predominantemente exploratório, sendo que o método de pesquisa principal foi o estudo de caso.

Neste estudo, de natureza exploratória, pode-se justificar o uso de métodos qualitativos pelo fato de ele envolver o estudo de DDS no seu contexto real, com a descrição e a compreensão do estado da arte naquelas situações em que a prática se antecipa à teoria [HOP 97]. Com relação à natureza do estudo, a pesquisa exploratória tem como principal finalidade desenvolver, esclarecer e modificar conceitos e idéias, com vistas à formulação de novas teorias, modelos e hipóteses pesquisáveis em estudos posteriores [GIL 95]. Ainda segundo este autor, a pesquisa exploratória muitas vezes constitui-se na primeira etapa de uma investigação mais ampla. Freqüentemente, o tema em foco é genérico, tornando-se necessário seu esclarecimento e delimitação, exigindo uma consistente revisão da literatura, discussão com especialistas e outros procedimentos.

Quanto ao uso de métodos de pesquisa qualitativa na área de SI, apesar de alguns autores adotarem uma abordagem mais positivista [LEE 89], a maior parte das pesquisas adota uma linha interpretativista, na linha do construtivismo, afastando-se da abordagem positivista [BEN 87]. Diversos autores destacam a emergência da abordagem interpretativista no estudo de Sistemas de Informação. Neste sentido, quanto à posição epistemológica, os métodos de estudo de caso inviabilizam a visão positivista, pois são caracterizados por um maior envolvimento do pesquisador no contexto da pesquisa e entrevista semi-estruturada em profundidade. Por isso, são considerados de caráter construtivista e interpretativista. O estudo de caso pode ter uma visão mais neutra e isenta, conforme propõe [LEE 89], porém a maioria dos autores tem uma posição crítica quanto a esta possibilidade, remetendo novamente a uma posição interpretativista. Com relação ao papel do pesquisador, na experiência vivenciada, a interferência do pesquisador é maior no processo, enquanto, no estudo de caso existe um distanciamento maior, sem haver a interferência direta do pesquisador no processo.

O método utilizado foi o estudo de caso, adotado conforme proposto por [YIN 01]. Neste estudo de caso foram identificadas dificuldades, soluções (práticas), e fatores críticos de sucesso do desenvolvimento distribuído de software. Utilizou-se como instrumento de pesquisa um roteiro para uma entrevista semi-estruturada, com questões abertas, caracterizando uma pesquisa do tipo trans-seccional.

Por tratar-se de uma pesquisa qualitativa, devem-se ter claras as limitações deste tipo de pesquisa, principalmente no que se refere ao número de empresas estudadas, restringindo a generalização dos resultados obtidos. No capítulo 6 deste estudo, aborda-se com mais profundidade a questão dos limites da pesquisa.

3.1 DESENHO E ETAPAS DA PESQUISA

O desenho de pesquisa contempla os componentes básicos de uma pesquisa qualitativa, quais sejam: questão de pesquisa, unidade de análise e critérios para interpretar os resultados (protocolo de análise). Embora o desenho de pesquisa contemple três etapas, esta pesquisa se concentrou apenas nas atividades da etapa um. Isto se deveu ao tempo disponível para realizar tal pesquisa e pela qualidade que se desejava buscar. As etapas dois e três serão desenvolvidas em estudos futuros, como continuidade do que será desenvolvido neste. Sendo assim, o objetivo principal foi à criação de um modelo de referência para o desenvolvimento distribuído de software, respondendo à seguinte questão de pesquisa: *Como os fatores envolvidos no DDS podem ser relacionados em um modelo de referência específico para esta área?*

A seguir apresenta-se o desenho de pesquisa (Figura 3.1), identificando suas principais etapas:

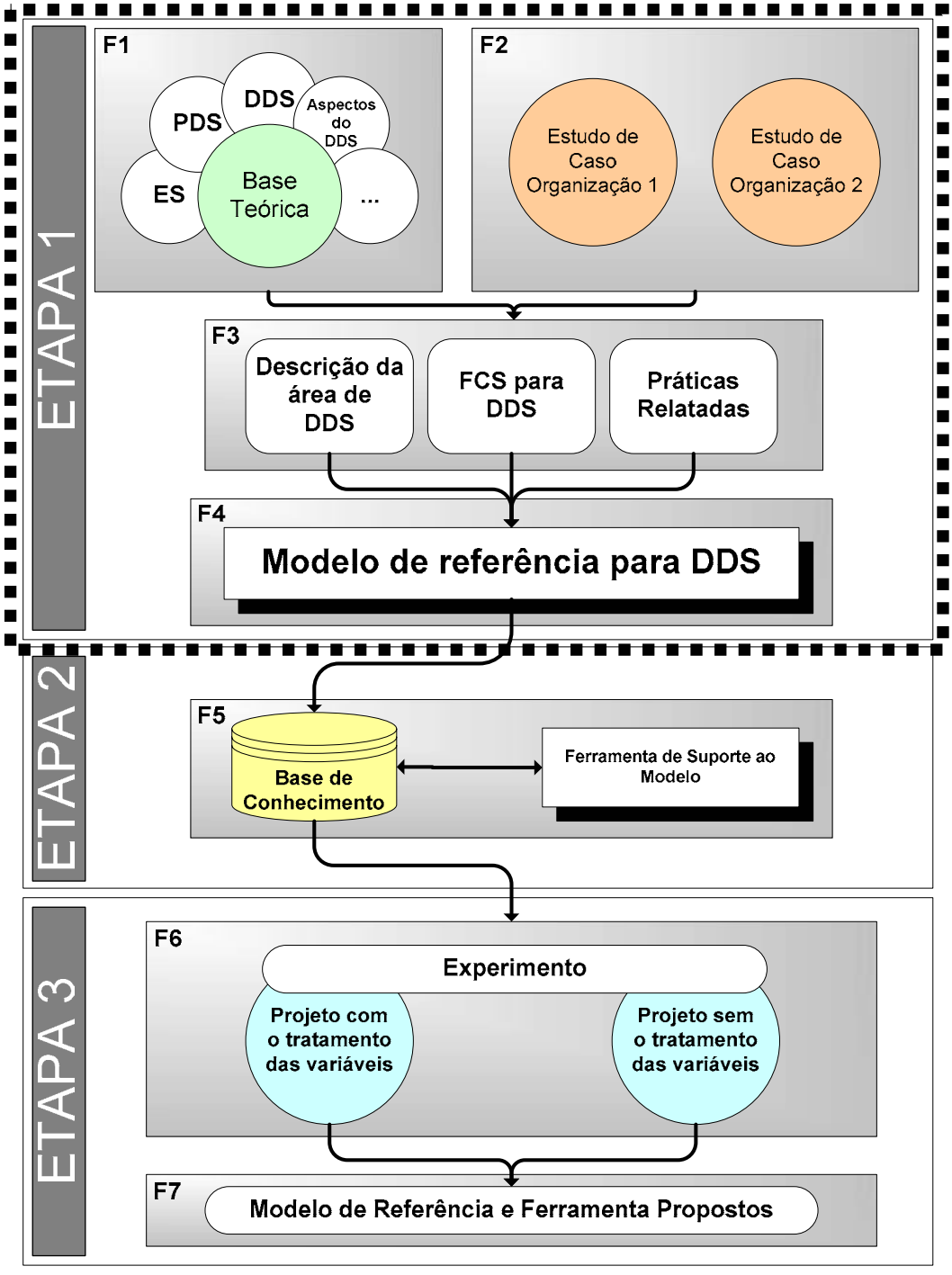

Figura 3.1 – Desenho de Pesquisa

A seguir, são descritas as quatro fases que compreendem a etapa um:

Fase 1: foi estudada a base teórica da área. Fazem parte da base teórica a engenharia de software, o processo de desenvolvimento de software (PDS), o DDS e os aspectos do DDS. Esta fase foi importante na medida em que formou uma base teórica consistente para a continuidade do estudo. Apesar de o estudo estar concentrado

principalmente em PDS e DDS, fez-se necessário o estudo de diversos outros conceitos relacionados, tais como qualidade de software, melhoria contínua, gerência de projetos, entre outros, visando complementar o referencial teórico e buscar conceitos inter-relacionados.

Fase 2: desenvolveram-se dois estudos de caso em duas organizações que possuem desenvolvimento distribuído de software, preferencialmente desenvolvimento *offshore*. Estes estudos de caso visaram levantar dados sobre as variáveis presentes em cada uma das organizações.

Fase 3: foram realizadas análises críticas sobre os dados coletados nas fases anteriores (teórica e empírica). Nesta fase gerou-se uma descrição da área de DDS, a avaliação da importância dos aspectos encontrados e as práticas identificadas nos estudos de caso.

Fase 4: na última fase da Etapa um, que é também a última fase da pesquisa, construi-se um modelo de referência para DDS contemplando as informações obtidas da base teórica e a consolidação dos dados extraídos do estudo de caso desenvolvido.

Conforme dito anteriormente, as etapas dois e três não fizeram parte desta pesquisa, sendo parte de estudos futuros. O objetivo da etapa dois será construir uma ferramenta para suportar o modelo gerado na etapa um. Por sua vez, a etapa três será dividida em duas fases: na primeira fase busca-se a validação dos artefatos (modelo de referência e ferramenta) construídos. Para isso, deverá ser realizado um experimento em uma das empresas participantes do estudo de caso da etapa um, onde o objetivo será verificar a aplicação do modelo e da ferramenta propostos e analisar o seu comportamento sob diversos prismas. Já na segunda fase pretende-se consolidar o modelo e a ferramenta propostos, considerando os resultados do experimento realizado na fase anterior. A seguir detalham-se um pouco mais as fases que compõe a etapa um deste processo de pesquisa, objeto desta dissertação de mestrado.

Suporte à Concepção

Pesquisas qualitativas pressupõem uma consistente experiência prática e conceitual do pesquisador na área [STR 89]. Assim, neste segmento do desenho de pesquisa (F1, F2 e F3), ao qual denomina-se de suporte à concepção, busca-se apresentar a experiência na área de estudo através da revisão da base teórica, do desenvolvimento do estudo de caso e da consolidação dos resultados encontrados.

F1: Base Teórica: A base teórica estudada contemplou as áreas que de alguma forma se relacionam com o DDS. Desta forma, buscou-se a partir da revisão

teórica o suporte necessário para a aquisição de um conhecimento mais aprofundado da área e o desenvolvimento do estudo de caso.

F2: Estudo de Caso: A definição do método foi decorrente do fato de que o estudo de caso permite o estudo aprofundado de uma ou mais organizações e, internamente, de diferentes segmentos e áreas vinculadas a um determinado projeto ou processo, permitindo o conhecimento mais aprofundado de seus impactos e conseqüências. Neste caso, as organizações estudadas estavam desenvolvendo projetos dentro de ambientes de DDS.

F3: Consolidação dos resultados: A consolidação dos resultados envolveu três atividades essenciais para a riqueza dos dados coletados: descrição da área de DDS, fatores críticos de sucesso e práticas relatadas.

Descrição da área de DDS: foi desenvolvida tendo como base o referencial teórico e os resultados do estudo de caso. Esta é tida como a primeira atividade em direção a consolidação de um modelo de referência para a área, entendendo modelo como sendo uma representação externa e explícita de parte da realidade vista pela pessoa que deseja usar aquele modelo para entender, mudar, gerenciar e controlar parte daquela atividade [PID 98].

Fatores Críticos de Sucesso (FCS) para o DDS: envolveu a identificação, através de perguntas e respostas diretas, dos principais aspectos que contribuem para o sucesso de projetos em ambientes de DDS, entendendo FCS como um conjunto de variáveis consideradas essenciais e que, apesar de diferirem de organização para organização, são de fundamental importância para o sucesso de projetos em ambientes de DDS.

Práticas relatadas: envolveu a identificação das principais dificuldades relacionadas com o DDS e as práticas que vêm sendo utilizadas para resolver estas dificuldades, não se limitando apenas à parte técnica do desenvolvimento de software.

Modelo Proposto

F4: Modelo de referência para DDS: Esta fase representa a criação do modelo proposto, agregando ao DDS as contribuições de áreas como psicologia, educação e sociologia, além da identificação de dificuldades, soluções e fatores críticos de sucesso, bem como a descrição da área de DDS.

A pesquisa desdobrou-se em diversas etapas, como pode ser visualizado na figura 3.2.

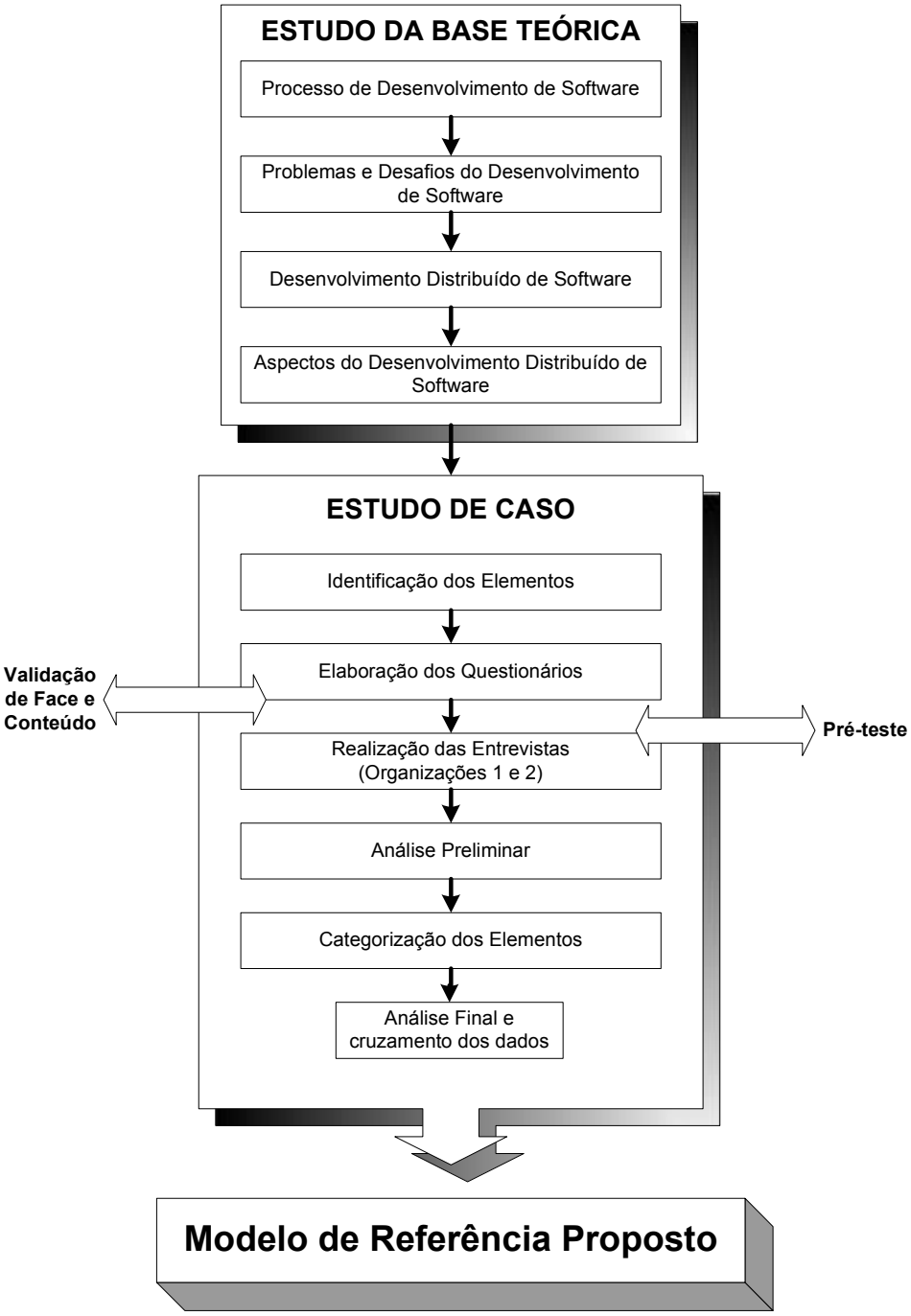

Figura 3.2 – Etapas da pesquisa

3.2 ASPECTOS METODOLÓGICOS

A definição e a utilização de protocolos para desenvolvimento e formalização dos estudos de caso, que se baseiam em um consistente referencial teórico, tiveram por objetivo uniformizar e sistematizar a tarefa de observação e análise, aumentando a confiabilidade do estudo. Finalmente, quanto à questão da generalização, acredita-se que seja possível adotar o método de generalização analítica [YIN 01], no qual uma teoria

previamente desenvolvida serve de quadro de referência na comparação entre resultados empíricos obtidos.

Desta forma, acredita-se ser possível garantir algum grau de generalização sobre os efeitos (resultado) e percepções gerais dos efeitos do modelo e sua aplicação. Esta estratégia tem sido utilizada na área de SI, em campos orientados pela prática e em pesquisas de teses e dissertações. Segundo [YIN 01], a generalização analítica pode ser defendida tanto se a pesquisa envolva um caso quanto múltiplos casos.

3.3 OPERACIONALIZAÇÃO DAS VARIÁVEIS

O modelo de pesquisa proposto envolve o detalhamento da consolidação dos elementos de pesquisa identificados a partir da revisão teórica desenvolvida no capítulo 2. O anexo 1 apresenta o protocolo de análise desenvolvido para suportar a geração, aplicação e análise dos resultados do questionário (Anexo 1). A seguir, no quadro 3.1, identificam-se os construtos teóricos utilizados na elaboração do questionário (coluna 1), as variáveis (coluna 2), as questões associadas ao construto (coluna 3) e o (s) principal (is) autor (es) que abordam a questão (coluna 4).

Quadro 3.1 – Construtos teóricos

Construtos Teóricos	Variáveis	Questões	Autores
Referenciais Estratégicos	Negócio	1. Qual é o negócio da empresa?	[AUD 01]
	Missão	2. Qual é a missão da empresa?	
	Políticas	3. Quais são as políticas da empresa?	
		4. Por que a empresa adota uma estratégia de desenvolvimento de software *offshore*?	
	Principais Clientes	5. Quais são os principais clientes da empresa?	
Processos de Negócio	Processos de Negócio	6. Quais os principais processos de negócios existentes na empresa?	[SPR 99]
Estrutura Organizacional	Estrutura Organizacional	7. Qual é a estrutura organizacional da empresa?	[SPR 99]
		8. Qual é a estrutura da empresa na área de Desenvolvimento de Software?	[SPR 99]
Processo de Desenvolvimento de Software	Processo de Desenvolvimento de Software	9. Existe algum processo formal de desenvolvimento de software utilizado pela empresa (metodologia, ciclo de vida e atividades)?	[PRE 01] [SOM 03]

	Projetos Suportados	10. Quais são os tipos de projetos suportados pelo processo de desenvolvimento de software?	[PRE 01]
	Estrutura das equipes	11. Qual é a estrutura da equipe de um projeto de software (papéis, quantidade)?	[PRE 01]
Dificuldades Encontradas	Dificuldades Encontradas	12. No seu projeto quais foram as principais dificuldades enfrentadas com relação ao desenvolvimento distribuído de software?	[KAR 98] [CAR 99] [EVA 00]
		13. Na sua experiência em outros projetos similares na empresa, quais foram as principais dificuldades enfrentadas com relação ao desenvolvimento distribuído de software?	
		14. Em quais atividades do processo de desenvolvimento de software cada dificuldade foi encontrada?	
Soluções	Soluções	15. Para cada dificuldade identificada, qual foi a solução adotada?	[KAR 98] [CAR 99] [EVA 00]
		16. Como esta solução foi identificada?	
	Impacto da solução	17. Na sua análise, qual foi o impacto da solução adotada, tanto para a empresa quanto para o projeto?	
Fatores Críticos de Sucesso	Fatores Críticos de Sucesso	18. Tendo por base este projeto e a sua experiência em outros projetos similares na empresa, na sua opinião, quais são os fatores críticos de sucesso principais para o desenvolvimento de projetos de software em ambientes de desenvolvimento distribuído de software?	[TUR 95]
Desenvolvimento Distribuído x Desenvolvimento Centralizado	Desenvolvimento Distribuído x Desenvolvimento Centralizado	19. Tendo por base a sua experiência em desenvolvimento de software, como você compara o desenvolvimento de software centralizado com o distribuído?	[KAR 98] [CAR 99] [EVA 00]

3.4 BASE METODOLÓGICA DO ESTUDO DE CASO

Segundo [HOP 97], o estudo de caso é particularmente adequado ao exame exploratório dos fenômenos ainda pouco estudados e que precisam ser investigados em seu ambiente de ocorrência. Diversos autores, como [BEN 87] destacam o estudo de

caso, conforme proposto por [YIN 01], particularmente adequado quando se tem por objetivo aprender sobre o estado da arte e gerar novas teorias apoiadas na prática, entender a natureza e complexidade do processo, enquanto este acontece, e trazer novos fatos e informações, evidenciados durante a execução de processo estudado.

O estudo de caso foi desenvolvido em duas unidades de desenvolvimento de software, cada uma pertencendo a uma organização multinacional com filiais no Brasil e no exterior, caracterizando-se assim um estudo com múltiplos casos. A primeira organização é da área de prestação de serviços de software, incluindo projetos de desenvolvimento, consultoria e treinamento. Ela possui diversas unidades de desenvolvimento de software no Brasil, responsáveis por atender a demanda de clientes externos, além de escritórios no Brasil e no exterior. É uma organização de grande porte para o seu segmento e a sua matriz está localizada no Brasil. A segunda organização trabalha com manufatura e suporte de computadores, possuindo três unidades de desenvolvimento de software localizadas em dois continentes, responsáveis por atender a demanda de projetos internos da mesma em âmbito mundial. Também é considerada uma empresa de grande porte para o seu segmento e a sua matriz está localizada nos Estados Unidos.

3.4.1 Seleção das Organizações e Unidade de Análise

A unidade de análise do estudo foi projetos de DDS (foram analisados dois projetos em cada organização), escolhidos de acordo com a caracterização distribuída dos mesmos. Conforme descrito anteriormente, foram escolhidas empresas dos setores de prestação de serviços de software e manufatura e suporte de computadores. O que justificou a escolha das organizações, dentro de um conjunto de cinco candidatas, foram o porte (grande porte pelo critério de faturamento e número de funcionários, segundo padrão SEBRAE/RS), a adoção de um processo de desenvolvimento formal e documentado, além da certificação mínima no nível dois de maturidade do modelo SW-CMM (*Capability Maturity Model for Software*). Todas as organizações disponibilizaram acesso irrestrito aos procedimentos deste estudo, tanto no acompanhamento do processo, quanto no que se refere à documentação. No capítulo 5 apresentam-se detalhadamente as organizações nas quais foi desenvolvido o estudo, culminando com as lições aprendidas em cada uma e a análise da aderência às abordagens de DDS identificadas na base teórica.

3.4.2 Fonte dos Dados e Seleção dos Participantes

A coleta de dados foi constituída por fontes primárias e secundárias. As fontes primárias foram constituídas de entrevistas. Foram realizadas 22 entrevistas semi-

estruturadas individuais em profundidade (11 em cada organização). Partiu-se de um roteiro básico com questões formuladas aos entrevistados e adequadas conforme seu desenvolvimento. Como complemento, foram utilizadas fontes secundárias, tais como a consulta e a coleta de documentos das organizações, tais como a missão, os referenciais estratégicos, os processos de negócios, as atas de reuniões, a descrição de processos de desenvolvimento de software, além de acesso ilimitado a *home-page* de ambas as organizações, entre outros.

Logo no início do estudo, realizou-se uma imersão em cada organização estudada, envolvendo uma visita guiada, durante um turno, percorrendo todos os setores de cada uma. Cada organização dedicou um responsável para acompanhar o estudo de caso e atuar como facilitador do processo e dar suporte nas entrevistas. Todas as entrevistas foram gravadas e transcritas para o papel em paralelo no decorrer da realização das mesmas. As respostas foram associadas à função e à área de atuação do respondente.

O critério inicial para a definição dos entrevistados centrou-se na unidade de análise e nos objetivos do estudo. Neste sentido, a população envolvida direta ou indiretamente constituía-se por integrantes das equipes dos projetos, gerentes de desenvolvimento, responsáveis pela área de qualidade e implantação dos processos de desenvolvimento de software na organização, além de um gestor de nível estratégico na organização. A amostra foi não probabilística, por conveniência, e procurou-se, em conjunto com a organização, uma representatividade dos diversos grupos envolvidos.

3.4.3 Análise de Dados

A técnica de análise de dados utilizada foi a de análise de conteúdos. Associada a ela foi utilizada a análise documental, para compreender as informações fornecidas pelos entrevistados. Embora as formas de análise tenham técnicas semelhantes (codificação de informação e estabelecimento de categorias), a análise documental tem no método histórico a sua forma mais conhecida, de modo que se buscou encontrar as relações entre os fatos sociais e sua ocorrência no tempo. A utilização de mais de uma técnica e fontes de informações possibilitou ampliar ao máximo a descrição, a explicação e a compreensão do objeto de estudo.

3.4.4 Fases e Operacionalização do Estudo de Caso

A pesquisa desenvolvida nas duas organizações permitiu desenvolver o estudo proposto plenamente, com forte apoio diretivo e acesso completo às informações correntes ou históricas dos projetos em análise. Isto envolveu não somente os contatos

visando desenvolver as entrevistas em profundidade, como também a coleta de documentos. Ambas as organizações forneceram todas as condições de espaço físico e apoio de material quando solicitado.

Para a organização que atua na área de manufatura e suporte de computadores, o contato ocorreu entre o pesquisador e o diretor da unidade de desenvolvimento de software da organização localizada em Porto Alegre. Após este contato inicial, foi realizada uma reunião para apresentar o protocolo do estudo de caso e obter a aprovação para realizar o estudo. Com a aprovação obtida, uma outra reunião foi realizada para definir os projetos que iriam constituir a unidade de análise do estudo, bem como os respondentes das entrevistas.

Com relação à organização que atua na área de prestação de serviços de software, localizada em São Paulo (SP), o contato inicial ocorreu através de uma representante da organização que trabalhava em Porto Alegre. Após este contato houve diversos contatos telefônicos para verificar a viabilidade de se realizar este estudo de caso. Neste sentido, o protocolo do estudo de caso foi enviado por e-mail para avaliação dos responsáveis. Com a aprovação da organização, foi possível agendar uma visita de dois dias à unidade de desenvolvimento de software em SP, para conhecer o ambiente físico da organização e realizar as entrevistas. A seleção dos projetos participantes e dos respondentes constituiu-se na primeira atividade do pesquisador em SP, juntamente com o responsável da organização.

O instrumento de pesquisa (questionário semi-estruturado) foi desenvolvido tomando-se por base um roteiro inicial de questões, a partir da teoria estudada e representada pelo protocolo de pesquisa desenvolvido para o estudo de caso (Anexo 1). Foram desenvolvidos dois questionários, cada um explorando uma dimensão de informações que deveriam ser capturadas: dimensão organizacional, contendo informações da organização como um todo e dimensão de projetos, contendo informações relacionadas aos projetos selecionados para participar do estudo. Foram realizados refinamentos sucessivos até a versão parcial do roteiro. Foi então realizada a validação de face e conteúdo por parte de um pesquisador sênior (doutor), professor da PUCRS.

A atividade seguinte foi a realização do pré-teste. Foram realizadas duas entrevistas, respectivamente com um gerente de projeto diretamente envolvido nas interações com equipes distribuídas e com um membro da equipe de qualidade de processo de software de uma das organizações envolvidas no estudo de caso. Com sua aplicação foi possível descobrir os inconvenientes, eliminar equívocos e ambigüidades e escolher a formulação mais adequada das perguntas para a finalidade da pesquisa. Após as entrevistas, foram realizadas novas alterações no roteiro e, finalmente, definido o

roteiro final. Depois de adequar o instrumento à luz do resultado do pré-teste e valendo-se de sugestões decorrentes das modificações realizadas, iniciou-se a fase de entrevistas.

Foram definidas entrevistas com onze profissionais em cada organização, selecionados em conjunto com os responsáveis das mesmas. Todas as entrevistas foram agendadas previamente e transcritas logo após a sua realização. Em virtude de algumas dificuldades no processo de transcrição, alguns respondentes foram contatados novamente para esclarecer alguns pontos.

Valendo-se das transcrições, desenvolveu-se a análise qualitativa destes dados. Inicialmente foi realizada uma análise de conteúdo em que se identificaram as categorias preliminares. Este processo foi desenvolvido independentemente pelo pesquisador e depois consolidado com o orientador, definindo um conjunto de categorias a serem consideradas. A figura 3.3 mostra as fases do estudo, descritas anteriormente.

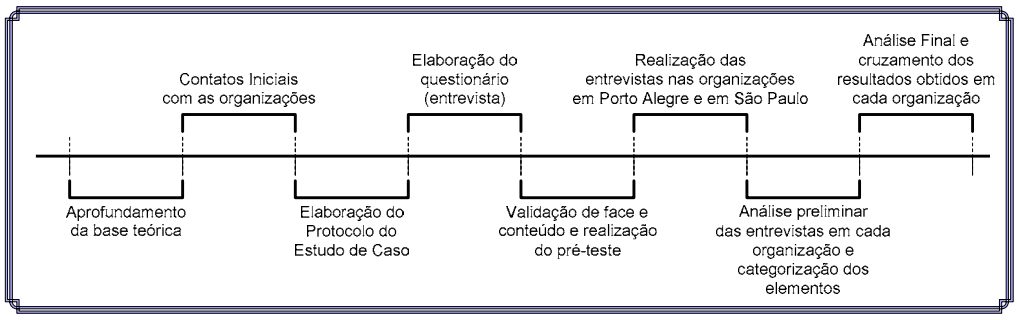

Figura 3.3 – Fases do estudo

A análise de conteúdo seguiu uma série de etapas, que iniciaram pela definição do universo estudado, delimitando o que estaria envolvido. Em seguida, iniciou-se sua categorização, representando tópicos significativos em função das quais o conteúdo foi classificado. A definição destas categorias é o procedimento mais importante da análise de conteúdo, visto que elas fazem a conexão entre os objetivos de pesquisa e os resultados. Seu valor fica sujeito à legitimidade das categorias de análise e depende da qualidade da elaboração conceitual feita *a priori* pelo pesquisador e da exatidão com a qual ele consiga traduzir os textos em categorias, permitindo, desta forma, formular conclusões e obter novas informações por meio do exame detalhado dos dados.

4 NÍVEIS DE DISPERSÃO EM DDS

"O importante não é o quanto você sabe, mas o quão bem você pode aplicar o que você sabe". Anônimo

Neste capítulo apresenta-se uma proposta de classificação de níveis de dispersão em desenvolvimento distribuído de software, considerando apenas a distância física. Para isso, nas seções 4.1 e 4.2 apresentam-se as distâncias físicas inter-atores e intra-atores respectivamente. Por último, na seção 4.3 apresenta-se a representação da classificação dos níveis de dispersão, considerando os atores envolvidos.

Ao longo desta pesquisa foram encontrados alguns modelos para caracterizar o nível dispersão geográfica em projetos de DDS [ESP 03], [LAY 00], [KAR 98]. Todos os modelos caracterizavam apenas o nível de dispersão da equipe de projeto, não considerando outros atores envolvidos no processo. Sendo assim, verificou-se uma oportunidade para refinar os modelos existentes, contribuindo no sentido de considerar a existência de três atores principais no processo [PRI 02b], [PRI 03a], [PRI 03c]:

A **equipe de projeto (P)**, que representa todos os envolvidos no desenvolvimento de um determinado projeto, podendo também ser formada por um conjunto de sub-equipes. Esta equipe pode envolver responsáveis pela área de negócios, gerência de projetos, desenvolvimento, testes, controle de qualidade, responsáveis pelo suporte de ferramentas dentro do projeto, entre outros. O **cliente (C)**, pessoa física ou jurídica que solicitou e contratou o desenvolvimento de um determinado projeto. O **usuário (U)**, que representa os responsáveis por fornecer as informações necessárias (requisitos) para o correto desenvolvimento do projeto e responsáveis por utilizar o produto gerado. Às vezes, clientes e usuários podem ser as mesmas pessoas, representando os dois papéis simultaneamente.

Visando uma representação fiel do nível de dispersão entre os atores, foram definidos dois critérios para a sua classificação: a distância física inter-atores e intra-atores.

4.1 DISTÂNCIA FÍSICA INTER-ATORES

Considerando os três atores existentes (equipe de projeto, clientes e usuários), este critério define a distância física existente entre eles. Para isto foram definidas quatro situações que verificam o tipo de distância e suas características principais. As situações diferem da proposta inicial feita em [PRI 02b], [PRI 03a] devido ao estudo e aprofundamento de outros modelos propostos para a mesma finalidade, gerando alterações no modelo inicial. Sendo assim, as situações para a distância física dos atores estão definidas da seguinte forma:

Mesma localização física: esta é a situação onde a empresa possui toda a sua equipe instalada em um mesmo local, considerando todos os atores envolvidos. Nesta situação, reuniões ocorrem sem dificuldades e a equipe pode interagir estando fisicamente presente. Não existe diferença de fuso-horário e as diferença culturais raramente envolvem a dimensão nacional. Os obstáculos são os já existentes no desenvolvimento centralizado de software (Figura 4.1).

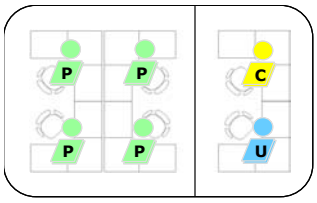

Figura 4.1 – Mesma localização física

Distância nacional: esta situação caracteriza-se por ter a equipe localizada dentro de um mesmo país, podendo se reunir em curtos intervalos de tempo. Dependendo do país, pode haver diferenças em relação ao fuso-horário e as diferenças culturais podem ocorrer em maior escala do que na situação anterior (Figura 4.2).

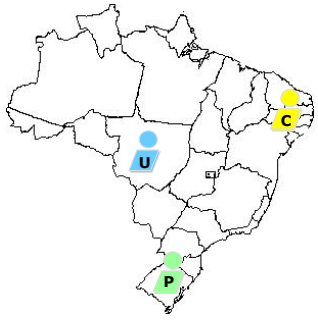

Figura 4.2 – Distância nacional

Distância continental: esta situação caracteriza-se por ter a equipe localizada em países diferentes, necessariamente dentro do mesmo continente. Nesta situação, as reuniões ficam um pouco mais difíceis de serem realizadas face a face, devida a distância física. O fuso-horário exerce um papel importante na equipe, podendo dificultar algumas interações (Figura 4.3).

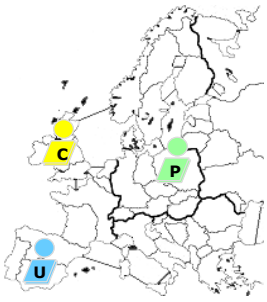

Figura 4.3 – Distância continental

Distância global: está situação caracteriza-se por ter as equipes localizadas em países diferentes e em continentes diferentes, formando muitas vezes uma distribuição global. Nesta situação, reuniões face a face ocorrem geralmente no início dos projetos e, entre outros fatores, a comunicação e as diferenças culturais podem ser barreiras para o trabalho. O fuso-horário exerce um papel fundamental, podendo impedir interações entre as equipes (Figura 4.4).

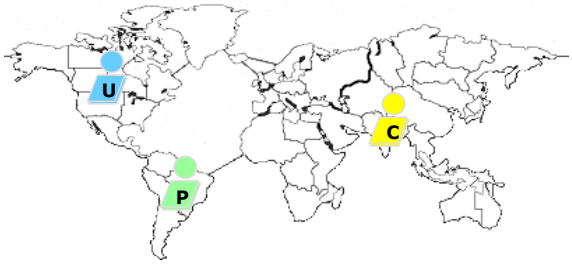

Figura 4.4 – Distância global

4.2 DISTÂNCIA FÍSICA INTRA-ATORES

Ter toda a equipe participante do processo de desenvolvimento de software (equipe de projeto, clientes e usuários) distante fisicamente de acordo com alguma das quatro situações previstas no item anterior não indica que um determinado grupo de atores esteja distribuído. Sendo assim, em um ambiente de desenvolvimento distribuído de software pode haver a distribuição interna dos atores. A distância física intra-atores é o critério que define a distância física existente dentro de cada equipe de atores (por exemplo, dentro da equipe de projeto ou do conjunto de usuários). As equipes podem estar centralizadas ou distribuídas. No caso de distribuição, a distância física pode

assumir qualquer uma das quatro possibilidades citadas anteriormente. Cabe salientar que a distribuição intra-atores não leva em conta a distribuição inter-atores.

4.3 REPRESENTAÇÃO DO NÍVEL DE DISPERSÃO

A partir dos critérios definidos, pode-se dizer então que a caracterização de um ambiente de Desenvolvimento Distribuído de Software (DDS) ocorre sempre que pelo menos um dos atores envolvidos (equipe de projeto, clientes, usuários) estiver fisicamente distante dos demais. Ao considerar esta definição para DDS, verificou-se a necessidade de utilizar uma escala para definir o nível (grau) da distribuição existente. Buscando-se uma representação para os critérios que definem o nível de distribuição do DDS, utilizando os atores identificados, a figura 4.5 representa graficamente os critérios propostos para definir o nível de distribuição do DDS (de distância física inter-atores e distância física intra-atores) e a relação entre eles.

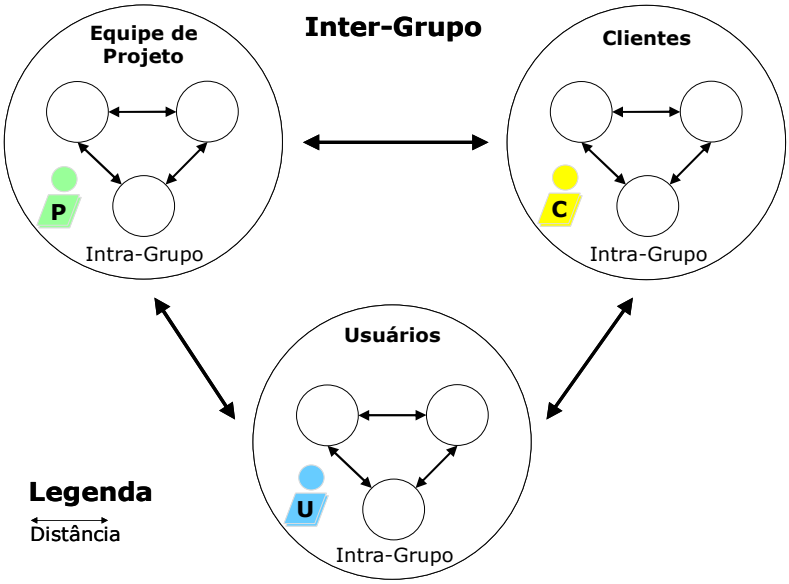

Figura 4.5 – Definição do nível de distribuição / dispersão no DDS.

Caso existam distâncias diferentes, deve ser considerada sempre a maior distribuição, tanto intra-atores quanto inter-atores. Como exemplo, considera-se integrantes de uma equipe de projeto localizados tanto no continente X como no continente Y. Se os clientes estiverem no continente Y, mesmo que parte da equipe de projeto esteja no mesmo continente, outra parte estará globalmente distante (continente X). Neste caso, a distância existente entre a equipe de projeto e os clientes usuários é a maior distância existente, ou seja, entre o continente X e o Y, caracterizando uma distância global.

5 RESULTADOS DO ESTUDO DE CASO

"Um artista, ao pintar, seleciona algumas coisas, recusa outras, e organiza todas". Ruskin, 1858.

Neste capítulo apresentam-se os resultados dos estudos de caso. Nas seções 5.1 e 5.2 apresentam-se os resultados encontrados em cada organização estudada. Na seção 5.3 são consolidados os resultados encontrados em cada organização e a seção 5.4 apresenta as lições extraídas da pesquisa que serviram de base para o modelo de referência proposto.

Elementos de Análise

Uma das mais importantes contribuições deste estudo envolve a análise das principais dificuldades vivenciadas pelas organizações com relação ao desenvolvimento distribuído de software e as soluções encontradas e implantadas. Neste sentido, a própria categorização resultante desta análise de conteúdos já é, por si só, parte relevante dos resultados desta pesquisa.

Esta análise permitiu relatar os resultados de forma a traduzir a realidade estudada e sua relação com os objetivos desta pesquisa. Buscou-se, entre outros objetivos, identificar as diferenças de percepção dos diferentes níveis gerenciais e operacionais das organizações participantes, levando-se em consideração o papel de cada respondente e o seu contato com as equipes distribuídas. A seguir apresentam-se os elementos analisados e as categorias obtidas (em separado para cada estudo de caso), buscando, ao final, direcionar para um conjunto de lições relevantes (abrangendo os resultados dos dois estudos de caso e da base teórica estudada) visando contemplar o objetivo principal deste estudo, relacionado com a busca de um modelo de referência para DDS.

5.1 ESTUDO DE CASO 1: *ORGANIZAÇÃO 1*

5.1.1 Caracterização da Organização

O primeiro estudo de caso (EC1) foi desenvolvido em uma unidade de desenvolvimento de software de uma organização de grande porte, chamada aqui de *Organização 1*, com sede em uma cidade do interior do Estado de São Paulo, no Brasil. A organização é nacional e possui escritórios nas principais capitais brasileiras e em alguns países no exterior, incluindo Estados Unidos (EUA), Espanha e Argentina, com mais de 350 clientes em todo o mundo e em torno de 2.500 colaboradores. Segundo dados fornecidos pela própria organização, um dos seus principais diferenciais é a abrangência geográfica. Sua estratégia de crescimento está centrada principalmente no investimento contínuo em gestão e na expansão focando novos mercados (principalmente no exterior).

O estudo de caso foi desenvolvido na matriz da empresa, onde funciona a principal unidade de desenvolvimento de software. A organização utiliza-se o conceito de *fábrica virtual*, no qual cada unidade é especializada em um ambiente operacional, tendo os projetos direcionados sempre ao centro de competência especializado. A distribuição ocorre quando os projetos são desenvolvidos utilizando-se recursos de mais de uma unidade ou quando são desenvolvidos distantes de clientes e usuários, aproveitando a capacitação de uma determinada unidade. A unidade possui certificação ISO[6] 9001 desde 1996 e nível dois de maturidade de desenvolvimento de software reconhecida pelo padrão SW-CMM desde 2002, com 80 colaboradores atuando no desenvolvimento de software. O estudo não considerou as outras unidades existentes. Com relação aos clientes, a unidade trabalha apenas com clientes externos à organização. A figura 5.1 apresenta a localização das unidades de desenvolvimento de software da organização.

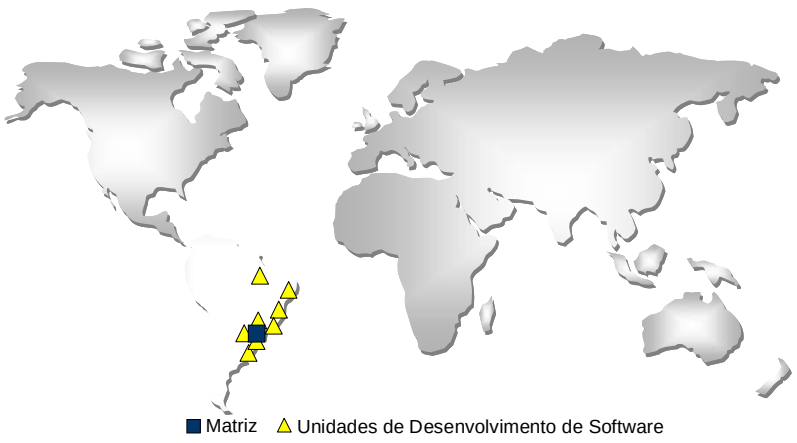

Figura 5.1 – Organização 1

[6] ISO – *International Organization for Standardization*, forma junta com a IEC (*International Eletrotechnical Commission*) um sistema unificado para padronização em âmbito mundial.

O quadro 5.1 apresenta os referenciais estratégicos e os processos de negócio da organização (questões 1 a 8 da entrevista):

Quadro 5.1 – Referenciais Estratégicos

Referenciais Estratégicos
Negócio
- Prover serviços completos e segmentados para todas as necessidades de tecnologia de informática, incluindo consultoria, fábrica de projetos, fábrica de programas.
Missão
- Atender as necessidades e superar as expectativas dos clientes através da inovação e dinamismo, provendo soluções de tecnologia da informação com alto valor agregado.
Valores
- Ética; - Responsabilidade social; - Honestidade; comprometimento; - Qualidade; perseverança; - Melhoria contínua e inovação.
Políticas Organizacionais
- Investimento contínuo em gestão; - Expansão para novos mercados; pesquisa e desenvolvimento de produtos e serviços com maior valor agregado; - Plano para aquisição de empresas com recursos próprios; agregar valor aos serviços e soluções oferecidos; - Procurar a excelência no relacionamento com os clientes e compromisso com responsabilidade social.
Principais Clientes
- Varig, Bradesco, TAM, Itaú, Volkswagen, 3M, McDonald's, Citibank, Nextel, Pirelli, Tim, Microsoft.
Motivos que levam a empresa a adotar uma estratégia de DDS
- Redução de custo; - Foco em qualidade; - Especialização de cada unidade distribuída (centros de competência); - Competitividade; - Necessidade de se manter o padrão da empresa em todos os locais; - A possibilidade de ter um banco de projetos; - A estratégia da empresa é de manter custos competitivos independente da localização.
Processos de Negócio
- Desenvolvimento de Software e Consultoria em informática.

5.1.1.1 Estrutura Organizacional

A figura 5.2 apresenta como está estruturada a organização e como a unidade de desenvolvimento de software está contextualizada dentro dela (questões 7 e 8) .

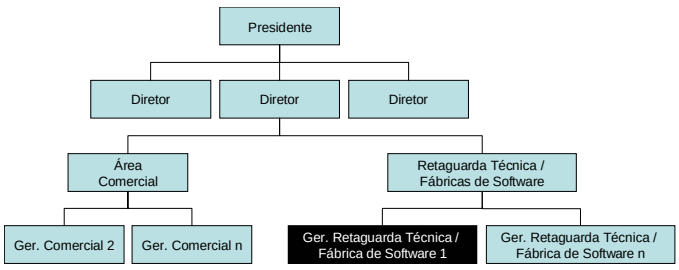

Figura 5.2 – Estrutura organizacional

A organização possui um presidente (CIO - *Chief Information Officer*) e diversos diretores. Segundo o entrevistado, um diretor pode ser responsável por uma ou mais áreas da empresa. Abaixo dos diretores existem duas áreas distintas, representadas

pela área comercial (equipes responsáveis pelo atendimento ao cliente e contratos), e pelas retaguardas técnicas (células responsáveis pelo desenvolvimento dos projetos). As equipes comerciais atuam dentro das unidades de desenvolvimento de software, enquanto que outras atuam apenas nos escritórios da organização. Com relação às retaguardas técnicas, elas são células responsáveis pelo desenvolvimento dos projetos. Cada retaguarda técnica possui uma fábrica de software envolvida e pessoas ou grupos de pessoas (geralmente de empresas parceiras) tecnicamente preparados para fornecer suporte à qualquer solução. As retaguardas técnicas são muito utilizadas quando não existem pessoas capacitadas tecnicamente para analisar uma solução para algum tipo de projeto na unidade. É uma espécie de suporte para ser utilizado quando necessário.

A unidade estudada possui alguns diretores responsáveis, envolvidos com uma ou mais áreas específicas (consultoria, área comercial, entre outros), dependendo da sua experiência. Esta unidade é coordenada por um gerente de retaguarda técnica / fábrica de software. Este gerente é responsável por todos os projetos em que a unidade está envolvida e pelas equipes subordinadas à ele (equipes de projeto, equipes de desenvolvimento de programas e SEPG – *Software Engineering Process Group*) além de também ser responsável pelo contato com os diretores da organização. A equipe de SQA (*Software Quality Assurance*) está subordinada à diretoria e não ao gerente da fábrica, devido à independência das ações a serem tomadas. Além disso, a unidade estudada possui algumas equipes comerciais, cada uma coordenada pelos respectivos gerentes.

Segundo a entrevista realizada, todas as unidades devem possuir a mesma estrutura, podendo variar em relação às equipes comerciais e ao número de diretores envolvidos. Eventualmente, uma unidade pode exercer apenas a função de fábrica de projetos, ou fábrica de programas. Nem todos os centros possuem o SEPG ou até mesmo a equipe de SQA, mesmo sendo fortemente recomendado. Além disso, a unidade estudada foi a primeira da empresa a ser oficialmente reconhecido como uma organização nível dois de maturidade de software segundo o modelo SW-CMM. Com relação à estrutura do centro na área de desenvolvimento de software, a figura 5.3 apresenta o organograma existente.

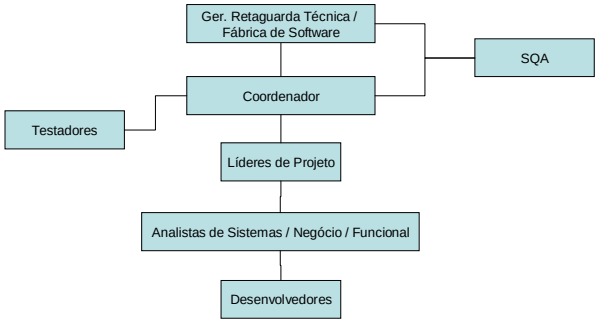

Figura 5.3 – A unidade estudada

Este organograma contempla apenas a fábrica de projetos. A fábrica de programas possui uma estrutura semelhante, mas não é objeto deste estudo. Todos os projetos desenvolvidos são monitorados em um nível mais gerencial pelo gerente de retaguarda técnica / fábrica de software. Além disso, cada projeto possui um coordenador, responsável pelas atividades de gerência do projeto. Apesar de a equipe de SQA estar subordinada à diretoria, no desenvolvimento dos projetos ele interage diretamente com o gerente da fábrica e com o coordenador do projeto.

O coordenador também é responsável pela equipe de projeto. Os membros da equipe de projeto podem representar papéis de líder do projeto (líder técnico), analistas de sistemas, de negócio ou funcionais, desenvolvedores, testadores e analistas de qualidade (SQA). Os papéis de analistas de sistemas, de negócio e funcionais são alocados dependendo das características do projeto, enquanto que os outros papéis devem estar presentes em todos os projetos desenvolvidos. Os papéis citados podem ser representados por mais de uma pessoa (isto é definido no início do projeto) de acordo com as características do mesmo.

5.1.2 Caracterização dos Projetos Analisados

Como critérios de seleção, optou-se por projetos caracterizados pelo desenvolvimento distribuído, tendo como suporte a definição de DDS fornecida no capítulo 2. O **Projeto 1** tinha como objetivo desenvolver uma aplicação para uma grande empresa cuja matriz estava localizada nos Estados Unidos. Apesar de o projeto ter sido requisitado pela matriz, era também gerenciado pela filial localizada no Brasil. De acordo com a classificação proposta no capítulo 4, a equipe de projeto, clientes e usuários estavam assim relacionados (Figura 5.4):

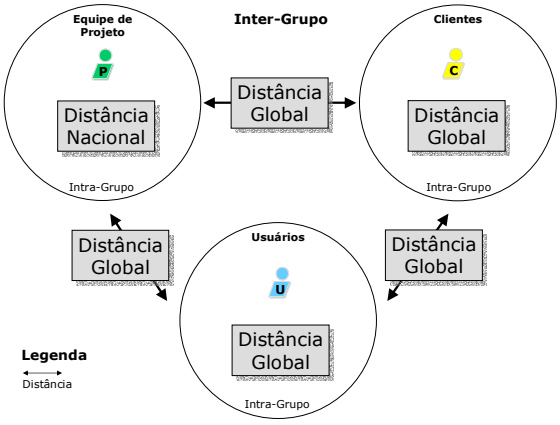

Figura 5.4 – Projeto 1

A equipe de projeto estava localizada no Brasil, em dois locais diferentes, enquanto que os clientes estavam localizados tanto no Brasil quanto nos Estados Unidos.

Além disso, os usuários eram todos os funcionários da empresa, mas no projeto eles estavam sendo representados por pessoas tanto dos Estados Unidos quanto do Brasil.

O **Projeto 2** tinha por objetivo desenvolver um sistema para um banco de grande porte do Estado de São Paulo. Neste sentido, o banco contratou uma empresa que contratou a unidade de desenvolvimento estudada, caracterizando assim o banco como sendo o usuário, a empresa contratante como sendo o cliente e atuando também como parte da equipe de projeto e a empresa contratada como sendo o restante da equipe de projeto. De acordo com a classificação proposta no capítulo 4, a equipe de projeto, clientes e usuários estavam assim relacionados (Figura 5.5):

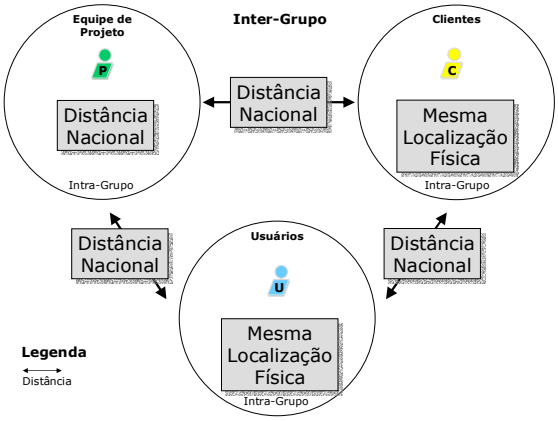

Figura 5.5 – Projeto 2

Tanto a equipe de projeto, quanto os clientes e usuários estavam localizados no Brasil. No entanto, a equipe de projeto estava distribuída em duas cidades, enquanto que o cliente e os usuários atuavam na mesma localização física, distantes fisicamente apenas de parte da equipe de projeto. A outra parte da equipe de projeto atuava dentro do banco. Neste sentido, o cliente e o usuário possuíam uma distância nacional em relação à equipe de projeto devido à dispersão física da mesma (distância intra-atores).

5.1.3 Processo de Desenvolvimento de Software

A unidade de desenvolvimento de software analisada possui um processo de desenvolvimento de software bem definido, seguindo as normas e padrões da ISO 9001[7] e do modelo de maturidade SW-CMM (nível 2). O processo é baseado em parte de outros processos conhecidos na comunidade acadêmica e empresarial, tais como RUP (*Rational Unified Process*) para o desenvolvimento de software e a metodologia do PMI (*Project Management Institute*) para o gerenciamento de projetos. Estes processos não são aplicadas na sua totalidade, sendo adaptados à realidade da organização e à outras partes do processo.

[7] ISO 9001 é a certificação concedida a empresas que atendam a rigorosas exigências de gestão da qualidade, sendo emitida por órgãos de reconhecimento internacional (http://www.iso.ch).

Segundo informado na entrevista, o processo é igual para todas as outras unidades de desenvolvimento, e é seguido o processo da unidade que possui o maior nível de maturidade de acordo com o modelo SW-CMM. Neste caso, todas as outras unidades deveriam seguir o processo de desenvolvimento de software existente na unidade estudada. O processo está dividido em 6 fases e segue rigorosamente as práticas definidas no modelo SW-CMM. Cada fase deriva algumas etapas e cada etapa deriva diversas atividades.

Por trabalhar com clientes externos à organização, todo o planejamento dos projetos é realizado em tempo de proposta. Ou seja, a análise preliminar dos requisitos e estimativas são elaboradas detalhadamente juntamente com o orçamento do projeto (fase especificar). As fases do processo de desenvolvimento de software são: controlar, especificar, prototipar, desenvolver, validar e implantar. Nestas fases estão incluídas as atividades de desenvolvimento e de gerência dos projetos. Como o objetivo não é detalhar o processo existente, a seguir explica-se brevemente cada uma das fases.

Controlar: fase onde é realizado o acompanhamento e o controle do projeto.

Especificar: fase onde é realizado o levantamento de informações sobre o projeto, a elaboração do desenho lógico, modelo de dados e projeto conceitual. O marco de final de fase é a apresentação e aprovação do projeto conceitual.

Prototipar: fase onde são desenvolvidas as telas funcionais e de relatórios do sistema. Dependendo do projeto podem ser desenvolvidos protótipos navegáveis. O marco de final de fase é a apresentação e aprovação dos protótipos desenvolvidos.

Desenvolver: fase onde são realizados a especificação e o desenvolvimento do projeto, além da execução dos testes. Para isto, existem atividades tais como preparação do ambiente, especificação, codificação e testes individuais, testes integrados e ajustes.

Validar: fase onde são executadas as validações junto ao cliente. O marco de final de fase é a apresentação e aprovação da validação.

Implantar: fase onde é elaborado o manual do usuário, treinamento do cliente no projeto e implantação do projeto em produção. O marco de final de fase é o aceite do projeto.

5.1.4 Caracterização dos Respondentes e sua Participação

A pesquisa foi desenvolvida de acordo com a abordagem metodológica apresentada no capítulo 3. Foram realizadas entrevistas com 8 profissionais (a previsão inicial era 10). Os participantes foram selecionados em função de seu papel na

organização. Foram entrevistados gerentes de desenvolvimento, gerentes de projeto, líderes técnicos, desenvolvedores, responsáveis pela qualidade e integrantes do SEPG. Todos os participantes entrevistados possuem pelo menos 3 anos de experiência na área de Informática, sendo o tempo médio de experiência de 10,4 anos. A média de idade dos entrevistados é de 28,3 anos, sendo a idade mínima de 18 anos e a idade máxima de 36 anos. De todos os entrevistados, 87.5% possuem um tempo de atuação na organização entre 2 e 5 anos e o restante possui um tempo de atuação de até 6 meses. As entrevistas tiveram uma duração média de 50,1 minutos (entre um mínimo de 20 minutos e um máximo de 81 minutos de duração) e contaram com total disponibilidade e atenção dos participantes. Foram fornecidas todas as informações solicitadas, sempre respeitando a política de privacidade e confidencialidade da organização.

De todos os entrevistados, sete pessoas faziam parte das equipes dos dois projetos selecionados e uma pessoa era integrante do SEPG. Com relação ao nível de formação, o grupo representa adequadamente o alto nível de qualificação dos funcionários da organização, sendo que 5 possuem no mínimo Curso Superior completo. Com relação à formação acadêmica, 6 vêm das áreas da Ciência da Computação e 2 da Administração. A função dos respondentes se distribui conforme a figura 5.6.

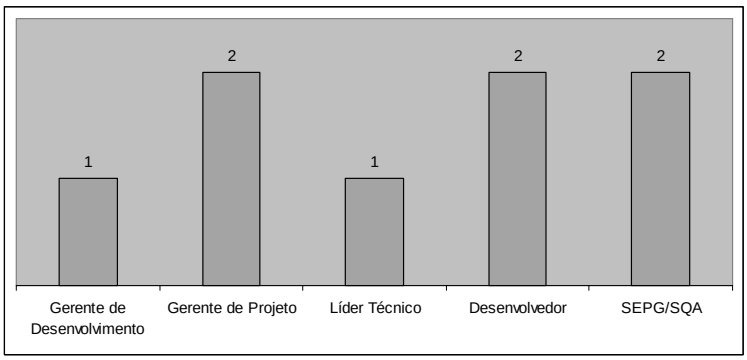

Figura 5.6 – Função dos respondentes

5.1.5 Elementos de Análise

A seguir apresentam-se os elementos analisados e as categorias obtidas.

5.1.5.1 Dificuldades do DDS

Esta questão buscava explorar a percepção e a vivência dos respondentes com relação às principais dificuldades vivenciadas em projetos de desenvolvimento distribuído de software. Foram citadas dificuldades relacionadas com comunicação, definição de requisitos, gestão de conhecimento, experiência em gestão de projetos, sincronização de ambientes de desenvolvimento, gerência de configuração, confiança entre as equipes distribuídas, definição de padrões e a diferença de idioma. Após a

análise das respostas obtidas, foram geradas algumas categorias. Neste sentido, após as discussões entre os pesquisadores envolvidos (orientador e pesquisador), ficaram definidas as seguintes categorias (Quadro 5.2):

Quadro 5.2 – Dificuldades encontradas

Dificuldades do DDS
Gestão do Conhecimento
Gerência de Configuração de Software
Gerência de Requisitos
Comunicação e idioma
Confiança
Falta de definição de padrões

Alguns trechos das entrevistas transcritas permitem ilustrar estes resultados, como, por exemplo, a citação de um dos coordenadores de projetos:

> "Apesar de existir uma base histórica de conhecimento sobre os projetos, ainda não existe uma cultura na organização para fazer um uso eficiente da informação nela contida. Muitas vezes o que fazemos é enviar um e-mail para uma lista específica de funcionários perguntando se alguém sabe algo sobre um determinado assunto. Apesar disso, já existe um sistema que objetiva a gestão do conhecimento na empresa. A idéia para a sua construção foi dada em um dos eventos de integração, realizado no final do ano passado. Hoje em dia é tudo feito por e-mail global (para toda a empresa), ou através do telefone. O conceito já existe, mas falta implantar".

5.1.5.2 Soluções Encontradas para as dificuldades do DDS

Esta questão buscava explorar a percepção e a vivência dos entrevistados com relação às principais soluções encontradas para as dificuldades apontadas. As soluções se concentraram principalmente na definição de padrões, gerência dos riscos, realização de reuniões periódicas com as equipes distribuídas, revisões e avaliações constantes, geração de atas com todos os problemas ocorridos, investimento em treinamento, investimento em planejamento e avaliação da produtividade das equipes. O quadro 5.3 apresenta as categorias identificadas, indicando forte ênfase da necessidade de haver um processo de desenvolvimento único e bem definido, na definição de padrões, treinamentos e avaliações constantes.

Quadro 5.3 – Soluções encontradas

Soluções Encontradas
Definição de padrões
Gerência de Riscos
Integração das equipes
Avaliação constante da produtividade das equipes
Investimento em planejamento
Documentação das atividades e dos problemas
Treinamento

A seguir, apresenta-se a opinião do gerente da fábrica de software a respeito das soluções encontradas:

> "A utilização de UML[8] no projeto 2 como linguagem padrão de modelagem do sistema facilitou o trabalho entre as equipes distribuídas, pois este foi um projeto onde o cliente exigiu algumas adaptações do nosso processo, o que avaliamos como sendo um risco para o projeto. Mas no final, houve um treinamento para toda a equipe para entender todos os padrões a serem utilizados e todas as modificações que seriam feitas para o desenvolvimento deste projeto, e o grupo respondeu positivamente. Mas eu acho que construir diagramas de casos de uso para trabalhar em projetos distribuídos não é suficiente. Para projetos internos, ou ainda projetos centralizados eu acho interessante, mas em um modelo distribuído acredito que muita informação se perde ao utilizar casos de uso, principalmente por que o nível de detalhamento da documentação deve ser bem maior, sem contar que deve ser perfeitamente entendido pela equipe distribuída".

5.1.5.3 Fatores Críticos de Sucesso do DDS

Nesta questão, buscava-se identificar a percepção e a vivência dos participantes com relação aos principais fatores críticos de sucesso para atuar em projetos de desenvolvimento distribuído de software. Diversos aspectos foram citados, concentrando-se na necessidade de haver uma comunicação eficaz, uma sintonia e sincronização entre as equipes, a definição e formalização de um processo claro, bem definido e único, a existência de condições de acompanhamento dos projetos por todos, seja através de ferramentas ou de reuniões periódicas. Também foi citada a importância da correta especificação dos requisitos dos projetos, pois, de acordo com os entrevistados, a especificação e a modelagem são o coração de qualquer projeto de desenvolvimento de software. Identificou-se que, nos projetos distribuídos, existe a sensação de que os requisitos recebidos estão todos especificados e estáveis, mas muitas vezes o que ocorre é justamente o contrário, devido aos desafios que o DDS apresenta.

Um aspecto relevante foi a necessidade de haver uma integração inicial entre as equipes distribuídas visando um entrosamento, além da definição clara do que se espera de cada pessoa envolvida no projeto, a necessidade de haver um treinamento adequado de acordo com as exigências dos projetos e a necessidade de se manter uma hierarquia e saber a quem recorrer. O quadro 5.4 apresenta as categorias finais identificadas relacionadas aos principais fatores críticos de sucesso encontrados:

Quadro 5.4 – Fatores Críticos de Sucesso

Fatores Críticos de Sucesso
Integração entre as equipes
Comunicação
Sincronização
Processos de desenvolvimento
Engenharia de Requisitos

Alguns trechos das entrevistas transcritas permitem ilustrar estes resultados, como, por exemplo, a citação de um dos desenvolvedores com relação à sincronização:

[8] UML (*Unified Modeling Language*) é a linguagem padrão para visualizar, especificar, construir e documentar os artefatos de um sistema intensamente baseado em software, podendo ser utilizada durante todo o ciclo de desenvolvimento e com diferentes tecnologias de implementação [SOM 03].

> "É muito importante que a comunicação entre as equipes seja freqüentemente refinada. Além disso, todo o projeto possui um cronograma, e muitas vezes ele não é seguido à risca. Quando estamos desenvolvendo de forma distribuída, aumenta ainda mais a dependência entre as equipes e as atividades envolvidas, e um atraso em alguma atividade pode significar em tempo ocioso para determinadas pessoas, devido a espera por atividades atrasadas".

5.1.5.4 Comparação do Desenvolvimento Distribuído com o Centralizado

Nesta questão, buscava-se identificar a percepção dos respondentes com relação à comparação entre o desenvolvimento distribuído e o desenvolvimento centralizado de software. Identificou-se uma diversidade de opiniões, destacando-se a existência de uma maior flexibilidade e facilidade na alocação (distribuição e redistribuição) e explicação das tarefas em um ambiente centralizado.

Já em um ambiente distribuído a dificuldade de se explicar um fenômeno aumenta, e com isso aumenta também a dificuldade de integração. Além disso, também se comentou da necessidade de se realizar alguns passos a mais em projetos de DDS, caracterizando um esforço adicional em gestão, principalmente nas tarefas que objetivam a sincronização do trabalho da equipe como um todo.

Alguns entrevistados comentaram o fato de que o desenvolvimento distribuído é mais formal, não podendo haver margem para fugir do processo previamente definido, e mais propício a conflito de papéis. Também foi comentado que, por ser um desafio, o DDS pode ser um incentivo a mais para a equipe, mesmo com a existência de dificuldades extras. Por último, a gerência de risco foi repetidamente citada como sendo um diferencial e um processo que não deve ser subestimado. O quadro 5.5 apresenta as categorias finais identificadas:

Quadro 5.5 – Comparação do DDS com o DCS

DDS X DCS
Gerência de projeto diferenciada
Incentivo
Rigor na execução processos
Gerência de Risco

Ao aprofundar a questão da confiança necessária entre as equipes distribuídas, um integrante do SEPG afirmou:

> "É realmente muito mais fácil administrar quem está perto. Existe a idéia do líder e/ou coordenador como uma pessoa de confiança. Além disso, se você não possuir alguém de confiança que esteja do lado da equipe distribuída, podem surgir problemas. O cliente estar distribuído dificulta bastante a engenharia de requisitos e aumentam as chances de haver um desentendimento com relação aos requisitos do sistema".

5.1.5.5 Comentários Adicionais

A última parte da entrevista foi caracterizada por um espaço onde os entrevistados foram convidados a acrescentar comentários que julgassem pertinentes.

Verificou-se neste caso que a maioria dos participantes efetivamente utilizou este espaço, concentrando os comentários em sugestões para melhorar os projetos desenvolvidos de forma distribuída com idéias que não foram citadas quando perguntados sobre as soluções encontradas para contornar as dificuldades.

Foram citadas outras dificuldades em relação às diferenças culturais e à diferença de idioma. Como soluções adicionais, citou-se que a implantação do modelo CMM acabou ajudando na definição de um processo que pudesse ser utilizado pelas equipes distribuídas, além de poder controlar e analisar diversas variáveis dos projetos e comparar com o desenvolvimento centralizado. Também se argumentou que nem sempre as soluções irão resolver todos os problemas, mas que se deve trabalhar para minimizar ao máximo seu impacto, investindo em planejamento.

Entre os comentários realizados, destaca-se o do gerente da fábrica de software:

> "As questões culturais (diferenças culturais principalmente) podem ser um empecilho para a engenharia de requisitos. Deve-se a todo custo evitar mal entendido, pois a possibilidade é grande, considerada um risco para o projeto. Além disso, a falta de entendimento da língua nativa pode ocasionar sérios problemas de comunicação, e isto deve ser verificado logo no início do projeto".

Um integrante do SEPG fez um comentário bastante interessante com relação ao que a implantação do SW-CMM ajudou no desenvolvimento distribuído:

> "A implantação do CMM no nosso centro ajudou no desenvolvimento distribuído, pois facilitou a definição de um processo único entre as equipes. Além disso, também possibilitou colocar em prática alguns controles e análise constante dos projetos".

Finalizando, um dos coordenadores de projeto completou:

> "Querendo ou não, sempre existiram e existirão problemas de relacionamento e confiança entre as pessoas e as equipes, seja em ambientes de desenvolvimento centralizado ou distribuído. As pessoas são diferentes, e o que podemos fazer é minimizar ao máximo os impactos destas diferenças, e o quanto antes fizermos isto, melhor para o projeto".

5.2 ESTUDO DE CASO 2: *ORGANIZAÇÃO 2*

5.2.1 Caracterização da Organização

O segundo estudo de caso (EC2) foi desenvolvido em uma unidade de desenvolvimento de software de uma organização de grande porte, chamada aqui de *Organização 2*, com sede em uma cidade no Estado do *Texas*, nos Estados Unidos. A organização é americana e possui escritórios em mais de 34 países em todo o mundo, inclusive o Brasil. Segundo dados fornecidos pela própria organização, sua estratégia de crescimento está centrada principalmente no marketing direto e no uso da internet. Possui em torno de 40.000 colaboradores em todo o mundo.

A unidade onde o estudo foi aplicado está localizada em uma cidade do Estado do Rio Grande do Sul, no Brasil. Ela possui mais de 110 colaboradores trabalhando em projetos que atendem as necessidades da área de TI da empresa. Esta unidade foi criada com os incentivos da Lei de Informática Brasileira (Lei Federal número 8.248/91) que estimula as organizações localizadas no país a investir parte dos seus ganhos em pesquisa e desenvolvimento, fornecendo isenção ou redução de taxas (IPI – Imposto sobre Produtos Industrializados) em produtos manufaturados de acordo com o Processo Produtivo Básico (PPB) brasileiro. Atuando em um ambiente de DDS, a maior interação é com a matriz, responsável pela demanda dos projetos.

A unidade possui nível dois de maturidade de desenvolvimento de software reconhecida pelo padrão SW-CMM desde 2003. Cabe salientar que o estudo foi aplicado apenas no contexto desta unidade, não sendo consideradas outras unidades de desenvolvimento da organização. Com relação aos clientes, trabalha-se apenas com clientes internos à organização. A figura 5.7 apresenta a localização das unidades de desenvolvimento de software e dos países onde a organização está presente.

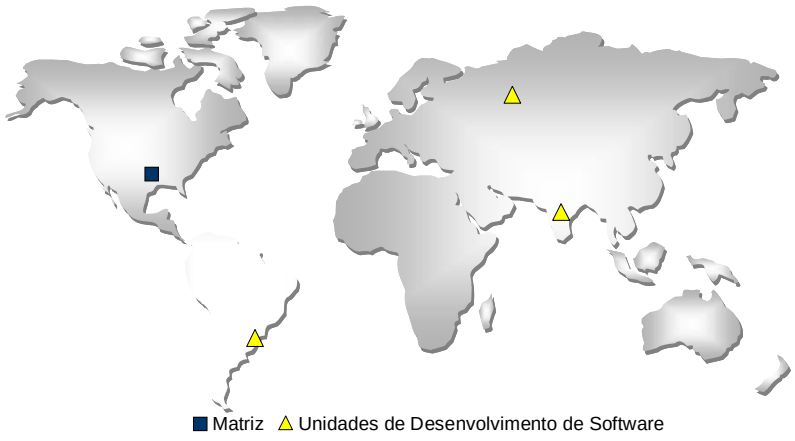

Figura 5.7 – Organização 2

O quadro 4.6 apresenta os referenciais estratégicos e os processos de negócio da organização (questões 1 a 8 da entrevista):

Quadro 5.6 – Referenciais Estratégicos

Referenciais Estratégicos
Negócio
- Desenvolvimento e aperfeiçoamento das aplicações de software e serviços da organização em um âmbito mundial.
Missão
- Desenvolver e aperfeiçoar as aplicações de software e serviços, conforme padrões de qualidade internacional, visando a melhoria e a criação de novos processos de negócios da organização em um âmbito mundial.
Valores
- Zelo pela imagem e cultura da organização; - Respeito à ética; - Agregação de valor ao negócio da organização em um âmbito mundial; - Valorização e respeito às individualidades e diferenças culturais; - Valorização da iniciativa, criatividade e do trabalho em equipe; - Qualidade no processo de desenvolvimento e aperfeiçoamento de software, bem como nos serviços prestados, disponibilizar informações e *feedback* constantes aos colaboradores (empregados, estagiários, terceiros, etc.); - Eqüidade no tratamento e igualdade de oportunidades para os colaboradores.
Políticas Organizacionais
- Desenvolvimento de software em conformidade com as expectativas dos clientes, em relação a prazo, escopo, qualidade e custo; - Estabelecimento de processos e políticas gerenciais consistentes, visando níveis de competitividade crescentes, propiciando trabalho em equipe com alto nível de participação e informação; - Desenvolvimento de programas de capacitação e oportunidades de crescimento pessoal e profissional dos colaboradores, proporcionando um ambiente de aprendizagem contínua, técnica e comportamental; - Geração de uma base de conhecimentos que possibilite altos níveis de aprendizagem organizacional a partir das capacidades individuais dos colaboradores e dos conhecimentos acumulados na organização; - Busca constante dos mais altos níveis de qualidade, por meio de certificação em padrões internacionais na área de software e gerência de projetos; - Direcionamento de todos os esforços no sentido de buscar soluções na área de software que agreguem valor aos processos de negócio da organização em um âmbito mundial; - Busca de uma integração e alinhamento constante e dinâmico com os referenciais estratégicos de negócio da organização.
Principais Clientes
- Áreas de TI da organização em âmbito mundial, que especificam seus sistemas de informação e subcontratam sua execução, teste ou suporte; - Áreas de negócio da organização, que são usuárias de sistemas de informação desenvolvidos internamente: vendas, suporte, pós-venda, RH, entre outras.
Motivos que levam a empresa a adotar uma estratégia de DDS
- Redução de custo; - Expansão para mercados globais; - Consolidação da marca da empresa fora dos Estados Unidos; - Necessidade de se manter o padrão da empresa em todos os locais.
Processos de Negócio
- Desenvolvimento de Software.

5.2.1.1 Estrutura Organizacional

A figura 5.8 apresenta como está estruturada a organização (organograma) e como a unidade de desenvolvimento de software estudada está contextualizada dentro dela (questões 7 e 8).

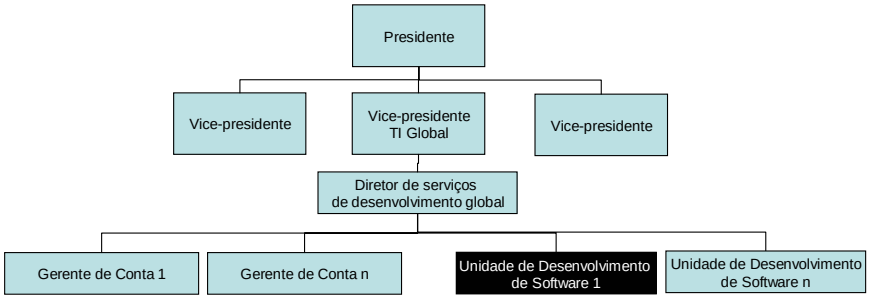

Figura 5.8 – Estrutura organizacional

A organização possui um presidente (CIO) e diversos vice-presidentes. Cada vice-presidente é responsável por uma determinada área ou departamento da empresa, sendo um responsável pela parte de serviços globais de TI. Uma das áreas controladas por este vice-presidente é a parte de operações de desenvolvimento de software global. Esta área envolve todas as células de desenvolvimento de software da organização em todo o mundo. Abaixo deste vice-presidente existe um diretor que controla as unidades de desenvolvimento de software da empresa (diretor de serviços de desenvolvimento de software global). Logo abaixo, estão as unidades propriamente ditas. Existe também a figura dos gerentes de conta, responsáveis pelo gerenciamento das contas dos clientes de cada unidade. Como os principais clientes são as áreas de TI da organização (clientes internos), os gerentes de conta têm como função principal auxiliar no controle dos projetos e na administração das contas (projetos em execução, projetos a serem desenvolvidos, entre outros).

A unidade estudada é coordenada por um diretor responsável por toda a parte administrativa e operacional do centro (desde recursos humanos até o desenvolvimento dos projetos). Ele é o responsável pelo contato com os diretores da organização como um todo. Abaixo do diretor existe um departamento responsável pelo suporte administrativo / gerencial e um consultor de RH. Existe ainda a área de qualidade, coordenada pelo SEPG. Com relação aos projetos, existe uma estrutura organizacional mista, com três áreas de desenvolvimento de software (corporativa, comercial e industrial). Cada área possui um gerente de desenvolvimento responsável pela alocação dos integrantes das equipes de projetos e responsável também pela gerência da equipe. Os gerentes de desenvolvimento também são responsáveis pela negociação dos projetos e pelo contato com os gerentes de conta e clientes.

Segundo a entrevista realizada, todas as unidades devem possuir a mesma estrutura, podendo variar um pouco em relação à nomenclatura e a organização. Além disso, a unidade estudada foi a primeiro dentro da empresa a ser reconhecida oficialmente como sendo uma organização nível 2 de maturidade de desenvolvimento de software de acordo com o padrão SW-CMM, sendo também a primeira empresa do

Estado do Rio Grande do Sul a obter tal reconhecimento. A figura 5.9 apresenta o organograma da unidade na área de desenvolvimento de software,

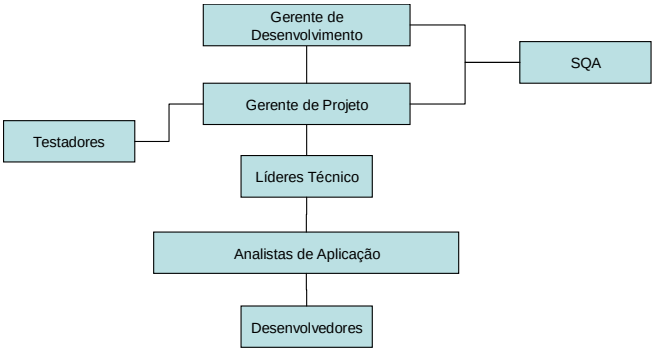

Figura 5.9 – O centro estudado

Todos os projetos desenvolvidos são monitorados pelo gerente de desenvolvimento. Além disso, eles possuem um gerente do projeto, responsável pelo gerenciamento das atividades e da equipe do projeto. Os membros da equipe de projeto podem representar os papéis de líder técnico, analista de aplicação, desenvolvedor, testador e analista de qualidade (SQA). O papel de analista de aplicação é alocado dependendo das características do projeto, enquanto que os outros papéis devem estar presentes em todos os projetos desenvolvidos. Os papéis podem ser representados por mais de uma pessoa de acordo com as características do projeto.

5.2.2 Caracterização dos Projetos Analisados

Como critérios de seleção, optou-se por projetos caracterizados pelo desenvolvimento distribuído, tendo como suporte a definição de DDS fornecida no capítulo 2. O **Projeto 1** tinha como objetivo desenvolver uma aplicação para gerenciar talentos dentro da empresa, e seria utilizado pela área de recursos humanos na matriz nos Estados Unidos. De acordo com a classificação proposta no capítulo 4, a equipe de projeto, clientes e usuários estavam assim relacionados:

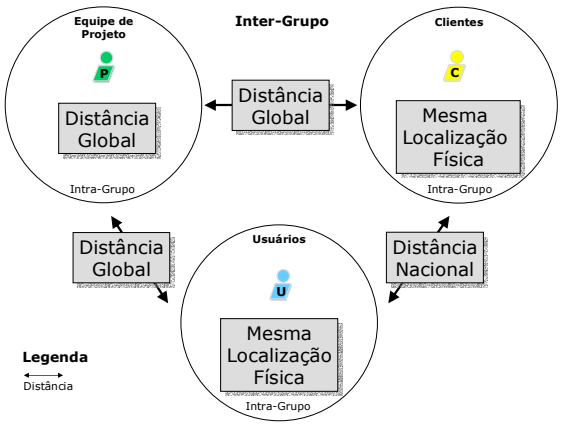

Figura 5.10 – Projeto 1.

A equipe de projeto estava localizada no Brasil e nos Estados Unidos, enquanto que os clientes (departamento de recursos humanos) estavam localizados nos Estados Unidos no mesmo espaço físico. Além disso, os usuários também estavam localizados nos Estados Unidos no mesmo espaço físico, mas distantes dos clientes.

O **Projeto 2** tinha como objetivo desenvolver uma aplicação para a área de manufatura da organização. De acordo com a classificação proposta no capítulo 4, a equipe de projeto, clientes e usuários estavam assim relacionados (Figura 5.11):

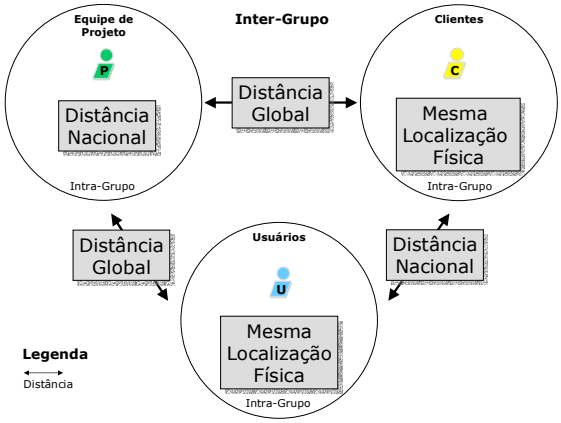

Figura 5.11 – Projeto 2

A equipe de projeto estava fisicamente distante, localizada no Brasil. Os clientes e os usuários estavam localizados nos Estados Unidos, no mesmo espaço físico, mas distantes fisicamente entre si.

5.2.3 Processo de Desenvolvimento de Software

A unidade de desenvolvimento de software analisada possui um processo de desenvolvimento de software bem definido, seguindo as normas e padrões do modelo de maturidade SW-CMM (nível dois). O processo é baseado em outros processos conhecidos na comunidade acadêmica e empresarial, tendo como base o MSF (*Microsoft Solution Framework*) para o desenvolvimento de software e a metodologia do PMI para o gerenciamento de projetos. Além disso, processos como o RUP e *Agile Modeling* também são utilizados, dependendo das características dos projetos.

Estes processos não são aplicados na sua totalidade, sendo adaptados à realidade da organização e à outras partes do processo, desenvolvidos pelos próprios funcionários. Com relação ao ciclo de vida, os projetos utilizam tanto ciclo de vida em cascata quanto em espiral (desenvolvimento iterativo). Além disso, ainda utilizam um ciclo de vida denominado de falso espiral, onde todas as atividades relacionadas ao levantamento de requisitos são realizadas em cascata, passando para um ciclo em espiral a partir das atividades de desenvolvimento. As regras para definir um ciclo de vida

específico estão centradas principalmente na estabilidade e clareza dos requisitos de cada projeto.

Segundo informado na entrevista, não existe ainda uma obrigatoriedade de os processos serem iguais em todas as unidades, apesar de ser fortemente recomendado. Isto se deve em parte pela diferença de maturidade dos processos e pelas unidades serem relativamente novas. Neste caso, em cada projeto se define como o processo será executado e gerenciado, procurando sempre manter um processo coerente com o que já está definido, incorporando-se algumas variações. O processo está dividido em quatro fases (as quatro fases definidas pelo MSF[9] na sua antiga versão) e segue rigorosamente as práticas definidas no modelo SW-CMM. Cada fase deriva diversas atividades, desempenhadas pelos integrantes das equipes de projeto.

5.2.4 Caracterização dos Respondentes e sua Participação

A pesquisa foi desenvolvida de acordo com a abordagem metodológica apresentada no capítulo 3. Foram realizadas entrevistas com 10 profissionais. Os participantes foram selecionados em função de seu papel na organização. Foram entrevistados gerentes de desenvolvimento, gerentes de projeto, líderes técnicos, desenvolvedores, responsáveis pela qualidade e integrantes do SEPG.

Todos os participantes entrevistados possuem pelo menos 5 anos de experiência na área de Informática, sendo o tempo médio de experiência de 12,4 anos. A média de idade dos entrevistados é de 32,1 anos, sendo a idade mínima de 23 anos e a idade máxima de 46 anos. De todos os entrevistados, 50% possuem um tempo de atuação na organização entre 2 e 5 anos, 37.5% possuem um tempo de atuação entre 1 e 2 anos e o restante possui um tempo de atuação de até entre 7 e 12 meses. As entrevistas tiveram uma duração média de 31,3 minutos (entre um mínimo de 16 minutos e um máximo de 55 minutos de duração) e contaram com total disponibilidade e atenção dos participantes. Foram fornecidas todas as informações solicitadas, sempre respeitando a política de privacidade e confidencialidade da organização.

De todos os entrevistados, sete pessoas faziam parte das equipes dos dois projetos selecionados e uma pessoa era integrante do SEPG. Com relação ao nível de formação, o grupo representa adequadamente o alto nível de qualificação dos funcionários da organização, sendo que 7 possuem no mínimo Curso Superior completo. Com relação à formação acadêmica, 7 vêm das áreas da Ciência da Computação e 1 da Engenharia Mecânica. A função dos respondentes se distribui conforme a figura 5.12.

[9] A nova versão do MSF foi publicada em junho de 2003 (http://www.microsoft.com/msf), mas ainda não está sendo utilizada pela organização.

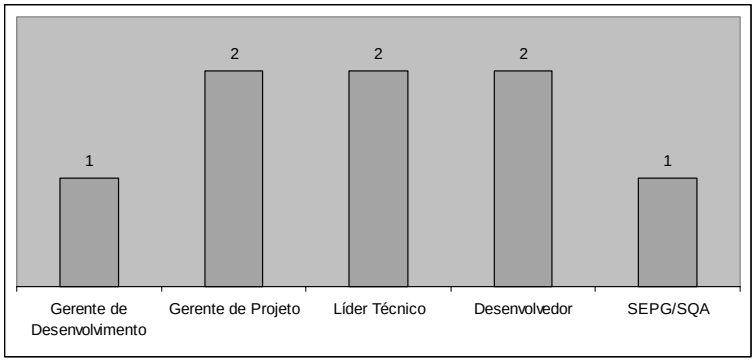

Figura 5.12 – Função dos respondentes

5.2.5 Elementos de Análise

A seguir apresentam-se os elementos analisados e as categorias obtidas.

5.2.5.1 Dificuldades do DDS

Foram citadas dificuldades relacionadas com mudanças freqüentes na definição dos requisitos, na falta de definição de protocolos de comunicação ágeis, falta de planejamento inicial do trabalho, dificuldades em se formar uma equipe com confiança entre os seus integrantes, devido a fatores como diferenças culturais e falta de compartilhamento de contexto.

Além disso, citaram-se também dificuldades devido à diferença de padrões de trabalho entre as equipes distribuídas, muitas vezes motivadas pelas diferenças culturais entre as equipes. Identificou-se ainda a falta de uma definição comum de processo de desenvolvimento de software e dificuldades com o idioma diferente. Neste sentido, após as discussões entre os pesquisadores envolvidos (orientador e pesquisador), as seguintes categorias foram definidas (Quadro 5.7):

Quadro 5.7 – Dificuldades encontradas

Dificuldades do DDS
Engenharia de Requisitos
Barreiras de Comunicação e Idioma
Diferenças culturais e contexto
Planejamento e definições de padrões de trabalho
Processo de Desenvolvimento de Software
Confiança

Alguns trechos das entrevistas transcritas permitem ilustrar estes resultados, como, por exemplo, a citação de um dos gerentes de projeto do centro:

> "Este projeto teve uma falha logo no início onde o pessoal da equipe distribuída forneceu ao cliente uma data final de entrega do produto antes de haver um detalhamento dos requisitos, ou seja, antes de ter uma estimativa fechada eles deram uma data final. Isto gerou um primeiro problema. Quando recebemos os primeiros requisitos e geramos a estimativa para o primeiro módulo, ficou claro que não seria possível cumprir a data final pela equipe no Brasil. Os módulos foram divididos entre as equipes localizadas no Brasil e nos Estados Unidos, e a equipe do

> Estados Unidos disse então para nós entregarmos o que fosse possível dentro do prazo fornecido, e o que não for possível dentro da nossa estimativa eles iriam fazer lá. Mas eles não estavam imaginando em seguir o mesmo padrão de desenvolvimento, mas sim um ”vamos fazer do jeito que dá para entregar no prazo prometido”. Então tu tens dois grupos distribuídos, mas com perspectivas totalmente diferentes um do outro. Para mim ai está o grande problema. Nós aqui no Brasil estamos trabalhando dentro de um processo que já foi reconhecido nos padrões do modelo SW-CMM nível 2 enquanto que a equipe americana entrou no projeto para salvar a data final, com uma filosofia de empresa que ainda está no nível 1, ou seja, os chamados heróis. Para mim, isto é a fonte de todos os problemas que tivemos no projeto, com dois grupos trabalhando com visões diferentes. Se o pessoal lá trabalhasse com estimativa, processo, documentação, testes, etc., nós não teríamos tantos problemas como tivemos até agora “”.

5.2.5.2 Soluções Encontradas para as dificuldades do DDS

As soluções se concentraram principalmente no planejamento do trabalho, na concentração de maiores esforços na fase de levantamento de requisitos, na necessidade de se estabelecer um padrão de trabalho e uma confiança entre as equipes distribuídas. Também se citou a necessidade de treinamentos específicos para facilitar a interação nos projetos. O quadro 5.8 apresenta as categorias finais identificadas:

Quadro 5.8 – Soluções encontradas

Soluções Encontradas
Planejamento
Treinamento
Definição de padrões de trabalho e comunicação
Aquisição de confiança
Engenharia de requisitos (definição do escopo do projeto)

Alguns trechos das entrevistas transcritas permitem ilustrar estes resultados, sendo que, a seguir, apresenta-se a posição de um líder técnico com relação à necessidade da definição de um processo único e à velocidade de resolução de problemas foi a seguinte:

> “Tem uma série de pequenos problemas que vimos que temos que levantar previamente, principalmente nos aspectos que irão afetar todo o desenvolvimento. Percebemos que é necessário mapear praticamente todos no início para evitar polêmica (padrões de código, metodologia, ciclo de vida que será utilizado, como irá funcionar a gerência de configuração, quem será o dono do código, como o desenvolvimento será feito, se as equipes distribuídas estarão trabalhando nos mesmos módulos ou em módulos diferentes...). Pelo fato de as equipes estarem distantes, existia uma grande diferença na velocidade com que nós resolvíamos os problemas. Sempre levávamos um tempo a mais. Isto nos remetia para a necessidade de se estabelecer um processo de comunicação bem ágil para que isto funcionasse bem. No nosso caso, tentamos estabelecer um processo de comunicação via e-mail e *conference call* para qualquer tipo de problema”.

Por outro lado, um dos gerentes de desenvolvimento entrevistados discorre sobre o desafio de se desenvolver software de forma distribuída:

> “Um grande trabalho que foi feito e resolveu boa parte dos problemas foi colocar um basta nas alterações de requisitos durante o projeto. Foi definida uma data para a entrega do projeto e nada mais foi alterado. Outro fato que ajudou foi a realização de diversas reuniões e todas foram documentadas. Não adiantava eu receber um e-mail e falar por telefone. Era necessário documentar, deveria ser seguido um fluxo normal. A conseqüência de todos os problemas foi o atraso de um mês no projeto. Já o impacto das soluções foi bastante positivo. Se nós não tivéssemos tomado algumas decisões, como a de evitar a ocorrência de tantas

> requisições de mudança (*changes request*) e fazer com que a equipe nos Estados Unidos entendesse o processo que estava sendo utilizado, talvez o projeto teria atrasado ainda mais ou nem teria terminado. Recebemos diversas propostas da equipe do projeto nos Estados Unidos, para trabalhar a noite inteira para acabar o projeto em tempo, mas nós sabíamos que o problema não era esse, tanto que o responsável pela equipe americana, quando visitou a equipe do Brasil, comentou que o problema era com a equipe dele e ele iria voltar e resolver ".

5.2.5.3 Fatores Críticos de Sucesso do DDS

Diversos fatores críticos de sucesso foram citados, entre os quais a necessidade de se gerenciar as expectativas, definindo claramente papéis e responsabilidades dos integrantes das equipes. Além disso, um fator muito citado foi a integração face-a-face entre as equipes, na medida do possível. Também se identificou a comunicação aberta e o *feedback*, a definição de requisitos, a definição dos processos e do padrão de trabalho, o nivelamento do conhecimento, a honestidade (confiança), uma boa infra-estrutura de comunicação e ferramentas de suporte ao desenvolvimento. Um aspecto relevante diz respeito à necessidade de se haver uma clara definição do *engagement* (engajamento das equipes no projeto) e também de um planejamento inicial, antes de qualquer interação relacionada ao projeto. O quadro 5.9 apresenta as categorias finais identificadas:

Quadro 5.9 – Fatores Críticos de Sucesso

Fatores Críticos de Sucesso
Gerenciamento de expectativas
Integração das equipes
Comunicação aberta e *feedback*
Processo de desenvolvimento de Software
Treinamento (nivelamento do conhecimento)
Planejamento e *Engagement*
Infra-estrutura

Alguns trechos das entrevistas transcritas permitem ilustrar estes resultados, sendo que, a seguir, apresenta-se um trecho da entrevista com um integrante do SEPG:

> "É necessário o estabelecimento de um processo de *engagement* bem definido, onde as duas partes distribuídas entrem em acordo. Também é preciso focar bastante a gestão de requisitos, nivelar o conhecimento entre as equipes, criando oportunidades para as pessoas se aproximarem e se conhecer pessoalmente. Além disso, são necessários uma disciplina rigorosa no seguimento dos processos e o estabelecimento de uma relação de confiança entre as equipes".

Como complemento, um desenvolvedor ainda acrescenta:

> "Eu acho que inicialmente deve haver uma comunicação aberta entre as equipes para que a confiança seja a máxima possível. É claro que quando a cultura das equipes é diferente, a aquisição de confiança não é tão fácil, mas se nós não chegarmos em um coeficiente razoável de trabalho, isto vai acabar comprometendo o projeto inteiro. Mesmo que os processos sejam diferentes entre as equipes distribuídas, onde quer que elas estejam, eu acho que deveria ser identificado um processo interno ao projeto (ferramentas padrão, fluxo das atividades, etc.), que as equipes distribuídas pudessem seguir para que não houvesse divergências e ambigüidades nas atividades desenvolvidas. E hoje o que eu vejo é que isto está muito amarrado ao centro (como o Brasil faz, como os Estados Unidos faz). Além disso, às vezes não recebemos tanto *feedback* quanto gostaríamos, de saber como as equipes estão percebendo o trabalho umas das outras".

5.2.5.4 Comparação do Desenvolvimento Distribuído com o Centralizado

A maioria dos respondentes acredita que o desenvolvimento distribuído é mais complexo que o desenvolvimento centralizado devido principalmente a falta de contato pessoal e a obrigatoriedade de haver um relacionamento distribuído, aumentando a necessidade de comunicação. As colocações foram centradas na gerência de riscos, onde alguns afirmaram que gerenciar riscos em projetos distribuídos é uma tarefa essencial e constante.

Além disso, outros entrevistados acrescentaram que projetos distribuídos necessitam de um maior controle, e conseqüentemente de um maior investimento na gerência do projeto. Por fim, alguns entrevistados citaram que o DDS pode ser encarado como um fator motivador, visto que envolve viagens e interação com pessoas em outros países. Além disso, é uma experiência que não se vê freqüentemente nos projetos desenvolvidos em ambientes centralizados. O quadro 5.10 apresenta as categorias finais identificadas:

Quadro 5.10 – Comparação do DDS com o DCS

DDS x DCS
Gerência de Riscos
Gerência de Projeto
Motivação
Comunicação

Para ilustrar as opiniões coletadas, um dos gerentes de projeto entrevistados disse:

> "O desenvolvimento centralizado geralmente é bom, pois tu conheces toda a equipe, não havendo dificuldades de comunicação face-a-face. Além disso, a equipe está mais na 'tua mão' e os riscos são mais fáceis de serem gerenciados. Em um desenvolvimento distribuído existem riscos adicionais para gerenciar, além das dificuldades de comunicação e idioma (quando for em países diferentes), mas a grande diferença está na identificação e no gerenciamento dos riscos".

Por outro lado, uma líder técnica entrevistada comentou:

> "Eu acho que dependendo da organização, desenvolver projetos de forma centralizada pode ser tão complicado quanto desenvolver de forma distribuída. De qualquer forma, o desenvolvimento distribuído necessita de um maior controle, confiança e união entre as equipes, e o centralizado muitas vezes é mais ágil".

Nas mesmas linhas, um dos gerentes de desenvolvimento teceu o seguinte comentário:

> "O desenvolvimento centralizado é muito mais fácil, tem uma probabilidade de sucesso muito maior. Em compensação, tem um lado motivacional quando o desenvolvimento é distribuído, principalmente quando é em outros países. As pessoas se motivam de tal jeito que é difícil de ver isto em um desenvolvimento centralizado, e isto para mim é um fator positivo. É algo novo, com possibilidades de se aprender outros idiomas e eventualmente conhecer outras partes do mundo".

5.2.5.5 Comentários Adicionais

A última parte da entrevista foi caracterizada por um espaço onde os entrevistados foram convidados a acrescentar qualquer tipo de comentário que se considerasse necessário, de forma a complementar a entrevista. Verificou-se neste caso que a maioria dos participantes efetivamente utilizou este espaço, concentrando os comentários em sugestões para melhorar os projetos desenvolvidos de forma distribuída com idéias que não foram citadas quando perguntados sobre as soluções encontradas para contornar as dificuldades.

Foram citadas dificuldades adicionais com relação às diferenças culturais entre as equipes, especificamente na forma como uma pessoa leva determinados problemas para o seu superior (escalar problemas). Também se acrescentou como dificuldades a freqüente falta de preparo dos líderes no desenvolvimento de projetos. Como soluções adicionais, citou-se novamente a necessidade de treinamentos, focando principalmente em treinamentos relacionados às diferentes culturas, bem como em treinamentos de preparação de líderes.

Entre os comentários, um dos gerentes de projeto entrevistados disse o seguinte:

> "Nós temos algumas questões culturais que eu acho que são muito importantes e que podemos realizar algum trabalho nisso. A primeira é uma questão bem cultural entre latinos e anglo-saxões em relação a escalar problemas. Nós brasileiros não possuímos esta cultura, pois imaginamos que no momento que tu escalas um problema tu estás prejudicando alguém, fazendo uma reclamação formal sobre uma determinada pessoa. Por outro lado, o pessoal nos Estados Unidos não tem essa visão. Quando alguém escala algum problema para a chefia, eles encaram como uma redistribuição de horas de trabalho e de prioridades. Eu acho que poderíamos trabalhar mais nestas questões, pois hoje nós escalamos problemas para a nossa chefia apenas quando ficamos brabos com alguém, no sentido de reclamar, e não no sentido de resolver o problema".

Em um outro comentário, um desenvolvedor acrescentou:

> "Em um projeto de desenvolvimento distribuído, as pessoas que estarão nas posições-chave, principalmente gerentes de projeto e líderes técnicos, devem ter uma condição favorável para exercer suas funções, isto é, estar bem preparadas em termos de liderança, ter uma forma clara para se expressar e ter uma boa base para diminuir ao máximo os ruídos que possam surgir".

5.3 CONSOLIDAÇÃO DOS RESULTADOS DOS ESTUDOS DE CASO

Os dois estudos de caso realizados evidenciaram diversos aspectos do DDS em duas organizações de grande porte, atuando em projetos distribuídos em nível internacional. Neste sentido, esta seção consolida os resultados dos dois estudos desenvolvidos. As categorias identificadas separadamente foram confrontadas com a teoria estudada e consolidaram-se os resultados obtidos na forma de lições aprendidas (seção 5.4). Claramente visualiza-se que as dificuldades, soluções e FCS envolvem três

dimensões: uma técnica, outra não-técnica e outra híbrida (envolvendo fatores técnicos e não-técnicos). Esta constatação direcionou a pesquisa a uma diferenciação, que se refletirá no modelo de DDS proposto. A seguir apresentam-se os resultados do esforço de consolidação para os elementos de análise estudados.

5.3.1 Dificuldades do DDS

De acordo com as categorias identificadas na análise preliminar, pode-se dizer que as dificuldades do DDS, nas organizações estudadas, estão centradas na falta de uma padronização das atividades entre as equipes distribuídas, na dificuldade de compartilhar informações e na falta de um processo de desenvolvimento bem definido (refletindo nas atividades de engenharia de requisitos). Além disso, também se verificaram dificuldades em relação a barreiras de idioma e comunicação, diferenças culturais, contexto e aquisição de confiança entre as equipes distribuídas (Quadro 5.11).

Quadro 5.11 – Dificuldades encontradas

Dificuldades do DDS	Dimensão	Fonte
Engenharia de Requisitos	Técnica	EC1, EC2
Processo de Desenvolvimento de Software	Técnica	EC1, EC2
Gerência de Configuração	Técnica	EC1
Gestão do Conhecimento	Técnica	EC1
Barreiras de Comunicação e Idioma	Não-técnica	EC1, EC2
Diferenças Culturais e Contexto	Não-técnica	EC2
Confiança	Não-técnica	EC1, EC2

A engenharia de requisitos foi identificada como sendo uma dificuldade constante, desde a necessidade de se identificar os requisitos dos projetos, passando pela análise, especificação, validação e gerência. Grande parte dos entrevistados mencionou que pelo fato de o projeto ser distribuído, os requisitos devem ser passados com o maior nível de detalhes possíveis, sem margens para falsas interpretações.

O processo de desenvolvimento de software é tido como uma das grandes dificuldades, pois nem sempre as equipes distribuídas estão ligadas por um mesmo processo. Os dados extraídos do estudo de caso permitiram constatar que nem sempre o processo é o mesmo, e muitos problemas podem surgir, evidenciando também outros aspectos, entre eles comunicação, confiança e gerência do projeto.

A gerência de configuração envolve o controle e o versionamento dos artefatos dos projetos. Os resultados obtidos através do estudo de caso permitiram identificar que no DDS a gerência de configuração é um ponto crítico, sendo a origem de muitos problemas relacionados ao desenvolvimento propriamente dito, pois muitas vezes os artefatos não possuem a mesma versão, nem o mesmo conteúdo nos diferentes locais onde o projeto está sendo desenvolvido.

Com relação à gestão do conhecimento, sabe-se que um dos principais recursos de uma organização hoje em dia é o seu capital intelectual. E o grande problema em relação ao capital intelectual é que ele geralmente não está em uma ferramenta, nem em um banco de dados, mas sim nas pessoas. Neste sentido, o estudo de caso evidenciou a falta de uma prática consistente de gestão do conhecimento em ambas as organizações, havendo uma grande dificuldade no sentido de compartilhar informações em ambientes distribuídos devido ao pouco investimento nesta atividade.

Com relação a barreiras de comunicação e idioma, sabe-se que o desenvolvimento de software, particularmente nos estágios iniciais, requer muita comunicação. Em ambientes de DDS, percebeu-se que as dificuldades de comunicação ocorrem freqüentemente, e com elas têm-se as dificuldades decorrentes da diferença de idiomas entre os integrantes das equipes distribuídas. A existência de diferenças culturais dá origem a diversas dificuldades em ambientes de desenvolvimento distribuído, principalmente quando a interação ocorre com equipes de diferentes países. As entrevistas realizadas identificaram dificuldades com relação à diferença de cultura tanto entre os membros das equipes quanto entre os locais fisicamente distantes.

Por último, a confiança em DDS é vital para o bom desenvolvimento dos projetos. No estudo desenvolvido identificaram-se algumas dificuldades relacionadas à criação de uma atmosfera de confiança e respeito entre equipes distribuídas. Este é tido como um princípio fundamental para o desenvolvimento de projetos de DDS.

5.3.2 Soluções encontradas para as dificuldades do DDS

Apesar de existirem diversas possibilidades de soluções aplicáveis visando minimizar os problemas do DDS, acredita-se que as soluções se concentram principalmente na necessidade de definir padrões de trabalho, investimento em planejamento e constante gerência de riscos, integração e aquisição de confiança entre as equipes, treinamento constante, um processo de desenvolvimento único e uma efetiva engenharia de requisitos. Neste sentido, conclui-se que, no que tange as soluções para as dificuldades do DDS, estas se concentram nas seguintes categorias (Quadro 5.12):

Quadro 5.12 – Soluções encontradas

Soluções Encontradas	Dimensão	Fonte
Planejamento e *Engagement*	Híbrida	EC1, EC2
Treinamento	Híbrida	EC1, EC2
Padronização	Técnica	EC1, EC2
Gerência de Riscos	Técnica	EC1
Processo de Desenvolvimento de Software	Técnica	EC1, EC2
Engenharia de Requisitos	Técnica	EC1, EC2
Aquisição de Confiança e Integração	Não-técnica	EC1, EC2

O planejamento inicial foi um dos aspectos identificados como necessário para selecionar corretamente projetos de DDS, avaliando se o projeto em questão se enquadrava em determinadas características para o DDS. Além disso, percebeu-se que antes de qualquer tipo de interação entre equipes distribuídas relacionadas à um projeto, se fazia necessário alinhar os objetivos das equipes, avaliar e optar pela melhor forma de interação (*engagement*).

O treinamento foi uma das soluções mais adotadas para as dificuldades encontradas. Ambas organizações investiram em uma alta carga de treinamento para as equipes das unidades estudadas, focando em temas como liderança, comunicação, cultura, gestão de projetos distribuídos e treinamentos técnicos exigidos pelo projeto. A padronização foi uma solução adotada quando os processos de desenvolvimento de software existentes não eram idênticos. Desta forma, as equipes interagiam para encontrar a melhor forma de desenvolver o projeto, padronizando todas as atividades que eram possíveis, visando principalmente minimizar problemas no decorrer do desenvolvimento.

Com relação à gerência de riscos, esta foi uma atividade identificada como estando em evolução em ambas as organizações. Percebeu-se que, pelo fato de a maioria dos riscos identificados nos projetos não envolverem questões específicas do DDS, fez-se necessário implantar uma gerência de riscos mais eficiente, tendo como premissa que as ações de mitigação executadas ao longo do projeto poderiam minimizar diversas dificuldades.

Pelo fato de ter-se identificado como dificuldade a inexistência de um mesmo processo de desenvolvimento de software entre as equipes distribuídas, ambas as organizações têm adotado como solução um alto investimento na padronização das atividades de desenvolvimento, motivadas principalmente pelo fato de serem reconhecidas como organizações SW-CMM nível 2 de maturidade de processo de software.

Para solucionar as dificuldades envolvendo a engenharia de requisitos, as organizações têm investido consideravelmente no levantamento de requisitos através de reuniões face-a-face. Cabe ressaltar que isto depende das características do projeto (projetos complexos, ou simples, grandes ou pequenos, tipo de tecnologia, entre outros). Além disso, tem-se investido na documentação e aprovações formais dos artefatos de projeto. Por último, mesmo com um grande investimento em treinamento, planejamento, gerência de riscos, entre outros, algumas atividades de integração das equipes têm sido executadas com sucesso. Estas visam principalmente à aquisição de confiança e o conhecimento dos participantes dos projetos. As atividades são quase todas

desenvolvidas quando as equipes (ou parte delas) podem se encontrar face-a-face. Mesmo assim, algumas atividades remotas geralmente ocorrem.

5.3.3 Fatores Críticos de Sucesso (FCS) para DDS

Quando se fala em fatores críticos de sucesso faz-se necessário ressaltar que a identificação destes fatores esta diretamente ligada com a forma de trabalho de uma determinada organização. Para uma mesma atividade podem existir diferentes fatores em cada organização, cada um ligado à estratégia adotada. Visando a consolidação dos resultados, pode se agrupar os FCSs em categorias como gerenciamento das expectativas das equipes (definição clara de papéis e responsabilidades, documentos existentes, entre outros), integração das equipes, comunicação eficaz e *feedback* constante, definição e formalização de um processo único, incluindo padrões de trabalho, investimento em treinamento, planejamento e clara definição do *engagement*, infra-estrutura para suportar a interação distribuída e o acompanhamento do projeto por todos os membros das equipes, na medida do possível. Neste sentido, conclui-se que, no que tange os fatores críticos de sucesso, as principais categorias são (Quadro 5.13):

Quadro 5.13 – Fatores Críticos de Sucesso

Fatores Críticos de Sucesso	Dimensão	Fonte
Processo de Desenvolvimento de Software	Técnica	EC1, EC2
Treinamento	Híbrida	EC1, EC2
Planejamento e *Engagement*	Híbrida	EC1, EC2
Infra-estrutura	Técnica	EC2
Gerenciamento de Expectativas	Não-técnica	EC2
Integração das Equipes	Não-técnica	EC1, EC2
Comunicação e *Feedback*	Não-técnica	EC1, EC2

A existência de um processo de desenvolvimento de software comum foi tida como um dos mais importantes fatores de sucesso dos projetos. Da mesma forma, o investimento em uma alta carga de treinamento refletiu principalmente no relacionamento entre as equipes distribuídas. O planejamento contribuiu no sentido de avaliar e selecionar corretamente projetos com características para serem desenvolvidos de forma distribuída. Por último, atividades de integração desenvolveram o lado não-técnico das equipes, promovendo a confiança e minimizando as diferenças culturais, repercutindo diretamente na comunicação e *feedback* necessários para o desenvolvimento dos projetos.

5.3.4 Comparação do Desenvolvimento Distribuído com o Centralizado

Percebeu-se durante o desenvolvimento do estudo de caso que a comparação do DDS em relação ao DCS depende muito das pessoas e dos papéis que elas representam em um determinado projeto. Enquanto alguns entrevistados apontaram

para a inexistência de diferença, sendo totalmente transparente, outros apontaram para a existência de uma maior carga relacionada à gestão dos projetos, uma maior freqüência e conseqüente dificuldade de comunicação, além de maiores riscos no processo como um todo. A maioria das respostas citou que apesar de o DCS ser mais ágil na resolução de determinados problemas, o DDS possui um incentivo a mais para as equipes, além de aumentar a motivação, por ser algo novo e atrair uma maior responsabilidade. Neste sentido, conclui-se que as principais diferenças entre o DDS e o DCS podem ser resumidas nas seguintes categorias (Quadro 5.14):

Quadro 5.14 – Comparação do DDS com o DCS

DDS X DCS	Dimensão	Fonte
Gerência de Projetos	Técnica	EC1, EC2
Gerência de Riscos	Técnica	EC1, EC2
Incentivo e motivação	Não-técnica	EC1, EC2
Comunicação	Não-técnica	EC1

A tabela anterior ilustra as categorias onde foram identificadas as principais diferenças entre DDS e DCS. Pode-se concluir que, como consolidação dos dados extraídos do estudo, o maior esforço na gerência de projetos repercute em uma maior necessidade de comunicação. Este esforço também existe na gerência de riscos, pois os desafios em ambientes de DDS são maiores. Além disso, a motivação e o incentivo foram citados em ambas as organizações como aspectos que atuam de forma diferente em cada tipo de desenvolvimento (distribuído e centralizado).

5.4 LIÇÕES PARA O ESTUDO

Os dois estudos de caso realizados nas organizações 1 e 2 ilustraram diversos aspectos presentes nos ambientes de DDS (seções 5.1 e 5.2 deste capítulo). A seguir destacam-se alguns destes aspectos e compara-se com a teoria da área (Capítulo 2). Estes resultados finais estão baseados no confronto entre a teoria e as descobertas empíricas, estão apresentados na forma de lições aprendidas, que será uma das bases de sustentação do modelo de referência proposto neste estudo.

Lição 1: A gerência de projetos e conseqüente gerência de riscos requerem um esforço e passos adicionais

Segundo [PMI 00] a gerência de projetos é a aplicação de conhecimentos, habilidades e técnicas para projetar atividades que visam atingir as necessidades e expectativas das partes envolvidas em um projeto. Uma má gerência de projeto pode significar a perda do projeto e dos recursos envolvido [ROY 98]. Com relação aos riscos, [McC 96] define que gerenciar riscos é uma das atividades do gerenciamento do projeto, e envolve identificar, tratar e eliminar fontes de risco antes que eles se tornem uma

ameaça concreta para o término de um projeto de software. Riscos podem ser tratados em diferentes níveis e não é uma tarefa simples. Atualmente um dos maiores desafios é a avaliação dos riscos e prevenção de possíveis danos.

No estudo realizado, as atividades envolvendo gerência de riscos e gerência de projetos foram tidas como de grande importância para projetos distribuídos e a grande maioria dos gerentes de desenvolvimento e gerentes de projeto entrevistados citaram que em projetos distribuídos estas atividades tomam proporções maiores que em projetos tradicionais (centralizados), requerendo um maior esforço e alguns passos adicionais aos modelos tradicionais existentes.

Lição 2: A existência de um processo de desenvolvimento de software único e bem definido responde por grande parte dos resultados obtidos em um projeto de desenvolvimento distribuído

Segundo [PRE 01], um processo bem definido é reconhecido como sendo processo que possui uma documentação que detalha o que é feito (produto), quando (passos), por quem (atores), os artefatos utilizados (insumos) e os artefatos produzidos (resultados). Além disso, o ponto de partida para a arquitetura de um processo é a escolha de um modelo de ciclo de vida (cascata, espiral, prototipação, entre outros).

Baseado nesta definição pode-se dizer que desenvolver software sem possuir um processo é um grande risco a ser gerenciado. Mais do que isso, a falta de um processo bem definido pode tornar ambíguas as atividades de um projeto, a ponto de a equipe de projeto não saber o que deve fazer em determinadas etapas.

O estudo realizado nas duas organizações permitiu identificar que nos projetos onde não havia uma clara definição do processo de desenvolvimento diversas dificuldades surgiram, muitas relacionadas com o próprio processo (requisitos, gerência de configuração, teste, etc.) e outras relacionadas com dificuldades derivadas, tais como comunicação, sincronização, confiança entre as equipes. Percebeu-se que um processo único e bem definido de acordo com o ambiente em que o projeto está sendo desenvolvido pode ser a solução para muitas dificuldades do desenvolvimento distribuído.

Lição 3: A gestão do conhecimento incentiva o compartilhamento de informações e estimula a aprendizagem por experiência

Gestão de conhecimento é uma disciplina relativamente nova cujo objetivo é absorver o capital intelectual de uma organização para um determinado fim [LIN 02]. Mas o grande problema relacionado ao capital intelectual é que ele está intrínseco ao ser-humano. O conceito de gerar informação para outros indivíduos transformarem em conhecimento próprio não é novo, mas ficou conhecido como gestão de conhecimento apenas a partir dos anos 80, tendo o seu crescimento acentuado a partir dos anos 90

[LIN 02]. Especificamente nas empresas de desenvolvimento de software este conceito tem sido aplicado no intuito de investir no aprendizado por experiência, ou seja, um indivíduo pode aprender a partir das experiências vivenciadas por outro, desde que estas experiências sejam relatadas.

O grande desafio é saber qual a utilidade das informações compartilhadas, como elas serão divididas entre os indivíduos e quais problemas estarão sendo atacados. A gestão do conhecimento auxilia na tomada de decisões, pode aumentar a qualidade e diminuir custos e tempo dos projetos (reuso), além de formar uma base de informações consistente para serem utilizadas no futuro.

Além de investir na absorção do capital intelectual de uma organização, a gestão do conhecimento pode estar relacionada com a coleta de informações sobre os projetos. Especificamente em projetos distribuídos, com os projetos sendo desenvolvidos de uma forma cada vez mais dispersa, a gestão do conhecimento é uma forma de estimular o compartilhamento de informações e conseqüentemente estimular a aprendizagem por experiência entre as equipes fisicamente distantes [DES 02]. Os projetos possuem diversos tipos de informações que, se compartilhadas, podem trazer vantagens como as citadas anteriormente.

As entrevistas realizadas indicaram que um grande diferencial do desenvolvimento distribuído de software está ligado ao fato de que um investimento na gestão do conhecimento (seja através de ferramentas ou atividades de incentivo ao compartilhamento de informação) estimula o aprendizado por experiência e minimizam certas dificuldades. Isto se deve ao fato de que muitas pessoas vivenciam certas situações e não compartilham, pois não são estimuladas para tal.

Lição 4: A engenharia de requisitos é o maior desafio do ponto de vista do processo de desenvolvimento de software

A engenharia de requisitos constitui em um conjunto de atividades muito importante e crítico no processo de desenvolvimento de software. Segundo [OBE 00], um requisito é a condição ou capacidade que um sistema que está sendo desenvolvido deve satisfazer. Além disso, uma pesquisa mostrou que 70% dos requisitos eram difíceis de serem documentados enquanto que 54% não estavam bem claros e organizados [OBE 00]. Estimativas também mostram que em torno de 40% dos requisitos geram retrabalho durante o ciclo de vida de um projeto [ZOW 02]. Na prática, algumas vezes problemas ocorrem devido a falta de alguns requisitos importantes, enquanto que em outras ocasiões os problemas são resultantes da especificação insuficiente dos requisitos, de forma a não contemplar todas as funcionalidades requeridas em um projeto [PRI 02a]. Em ambientes de DDS, a engenharia de requisitos surge como um grande desafio, na exata medida em que se deve buscar a melhor forma de identificar, analisar, negociar

e documentar os requisitos [DAM 03b], [DAM 02a], utilizando-se de ferramentas de apoio e técnicas que permitem uma maior proximidade das equipes distribuídas, juntamente com clientes e usuários, sendo os dois últimos responsáveis por identificar os requisitos do projeto em um nível mais alto.

Praticamente todas as entrevistas realizadas com os gerentes de projetos e líderes técnicos apontaram para dificuldades nas atividades que envolvem a engenharia de requisitos. Em um dos projetos a instabilidade dos requisitos foi o maior problema, principalmente devido à distância entre as equipes, o que dificultava o entendimento e a convergência de idéias. Em todos os projetos identificaram-se os requisitos como um grande desafio, envolvendo atividades desde a realização de reuniões até a formalização (documentação) dos requisitos que eram definidos, a rastreabilidade e controle dos mesmos.

Lição 5: A fase de planejamento é importante para a organização e a gerência de projetos de desenvolvimento distribuído de software

Definir as estratégias de uma empresa na área de sistemas de informação, a partir de um processo formal de planejamento (PSI) é um grande desafio para as organizações [AUD 01]. A ausência de uma etapa formal de planejamento pode ser apontada como um dos principais problemas no processo de desenvolvimento de software, diluindo decisões críticas ao processo como um todo em etapas subseqüentes onde perde-se a dimensão sistêmica da aplicação ou do problema em análise [AUD 01], [REP 98]. Segundo [MAR 91], a ausência de maior rigor na etapa de planejamento acarreta um grande número de problemas nas etapas subseqüentes.

Como lição do estudo desenvolvido nas duas organizações, identifica-se o planejamento como uma etapa fundamental para decidir e conseqüentemente planejar o desenvolvimento de um projeto de forma distribuída. Apesar de existir suporte ao planejamento na literatura da área de ES, a literatura relacionada ao DDS se concentra principalmente no desenvolvimento de software e na gerência de projeto. Por este motivo, de forma a complementar a literatura da área, acredita-se que a visão do processo de planejamento como sendo uma etapa preliminar a um conjunto de projetos de desenvolvimento de software é um fator decisivo também em projetos distribuídos.

Lição 6: O investimento em gestão de equipes distribuídas minimiza as dificuldades da dimensão não-técnica

Freqüentemente, os gerentes de projeto têm tido a necessidade de organizar e gerenciar projetos de desenvolvimento de software envolvendo equipes formadas por indivíduos provenientes de culturas diferentes, distribuídos geograficamente. Segundo [KIE 03], as barreiras técnicas para esta realidade estão sendo resolvidas rapidamente.

Por outro lado, os aspectos humanos têm sido estudados como uma menor ênfase. Sendo assim, [KIE 03] acredita que quando um projeto distribuído falha, geralmente é resultado de uma combinação de problemas envolvendo aspectos sociais, culturais, lingüísticos e políticos.

Outros aspectos ainda podem ser acrescentados a esta lista (comunicação, contexto, relacionamento interpessoal), mas o que deve ser ressaltado é o que foi comprovado no estudo de caso desenvolvido. A falta de investimento na gestão de toda a equipe distribuída, focando aspectos como a comunicação, diferenças culturais, confiança, idioma e contexto podem repercutir em problemas inesperados em todo o desenvolvimento do projeto.

Uma das organizações estudadas investiu em treinamentos para a equipe localizada na unidade de desenvolvimento de software. Como resultado, as interações com parte da equipe distribuída (incluindo equipe de projeto, clientes e usuários) fluíram mais facilmente. Problemas identificados antes dos treinamentos passaram a ocorrer com menos freqüência, indicando que a gestão das equipes distribuídas não pode ser menosprezada.

Lição 7: Ferramentas de apoio atuam como facilitador na interação distribuída

Embora os estudos de caso desenvolvidos não apresentem fortemente a importância das ferramentas como apoio para as atividades em projetos distribuídos, a teoria [LAN 02], [LAN 03], [SAR 02], [HER 02], [ALT 98] evidencia que parte dos problemas relatados podem ser endereçados pelo uso de ferramentas especificamente desenvolvidas para suportar ambientes de DDS e suas atividades (colaboração, cooperação, gestão de conhecimento, engenharia de requisitos, etc.). Cabe ressaltar que, segundo [HER 02], o uso de ferramentas de apoio depende muito das características das equipes pois, visto que as pessoas são diferentes, estas diferenças podem se refletir no uso de uma determinada ferramenta.

Lição 8: Desenvolver software de forma distribuída é um processo que amadurece com o tempo

Segundo [HOU 03], maturidade significa o estado ou condição de pleno desenvolvimento, estado ou qualidade de maduro. Quando o modelo CMM [PAU 93] foi proposto, ele foi estruturado e desenvolvido de uma forma que pudesse ser representado em níveis de maturidade, ou seja, que de alguma forma houvesse a representação da maturidade dos processos de uma organização através de níveis, identificando assim o quão maduros e desenvolvidos estariam estes processos. Em cada nível existiria um conjunto de práticas e padrões a serem obedecidos [PAU 93]. No caso dos processos de

software, estes estavam organizados no modelo de maturidade conhecido por SW-CMM, atualmente evoluído para o modelo conhecido como CMMI [CMM 03]. Após a criação do *framework* CMM, diversos outros modelos de maturidade surgiram em diferentes áreas e para diferentes atividades [NAW 01], [MAT 01], [FRA 97].

A partir dos dados extraídos do estudo de caso desenvolvido, percebeu-se claramente que existia uma diferença de maturidade entre as organizações com relação ao desenvolvimento distribuído de software. Por estar a mais tempo atuando em projetos distribuídos, a *Organização 1* estava em um estágio mais avançado, com outros tipos de problemas sendo identificados. Por outro lado, além de identificar uma maior quantidade de problemas, o tempo de atuação em projetos distribuídos era consideravelmente maior na *Organização 2* em relação à *Organização 1*. Neste sentido, com base nos dados coletados e das entrevistas realizadas, chegou-se a conclusão que o desenvolvimento distribuído de software é um processo que amadurece com o passar do tempo.

Apesar de existir suporte à modelos de maturidade na literatura da área de ES, a literatura relacionada ao DDS não aborda muito o conceito de maturidade especificamente em projetos distribuídos. Os trabalhos existentes apresentam apenas conceitos de maturidade de equipes distribuídas [CAR 99]. Além disso, um trabalho realizado por [CAR 02] procurou identificar a maturidade das organizações para trabalhar com desenvolvimento *offshore*, sendo um trabalho mais focado ao planejamento e estratégia adotada e não ao desenvolvimento dos projetos em si. Por este motivo, a identificação da maturidade de projetos distribuídos complementa a literatura da área.

Resumindo as lições apresentadas, o quadro 5.15 sintetiza os pontos observados, relacionados com as lições aprendidas durante este estudo. Estas lições constituem-se em um dos elementos de sustentação do modelo proposto.

Quadro 5.15 – Pontos observados no estudo

Lição	Ponto Observado
1	Gerência de Projeto e Gerência de Risco
2	Processo de Desenvolvimento de Software
3	Gestão do Conhecimento
4	Engenharia de Requisitos
5	Planejamento
6	Gestão de equipes distribuídas
7	Ferramentas
8	Maturidade

6 MODELO DE REFERÊNCIA PROPOSTO

"Somos o que repetidamente fazemos. A excelência não é, portanto um feito, mas um hábito". Aristóteles

Neste capítulo apresenta-se o modelo de referência proposto. Na seção 5.1 descreve-se o modelo e suas características e na seção 5.2 apresenta-se uma proposta de estágios de capacidade baseada no modelo descrito.

As dimensões propostas por [EVA 00] e as forças centrífugas e centrípetas propostas por [CAR 99] se concentram nos principais fatores presentes nas equipes em projetos distribuídos e projetos de desenvolvimento distribuído de software respectivamente. Além disso, [KAR 98] identifica um conjunto de atividades necessárias ao longo do ciclo de vida de um projeto distribuído. Adicionalmente, outros estudos ilustrados no quadro 2.2 aprofundam a análise destes fatores e acrescentam outros presentes não apenas nas equipes, mas em projetos de DDS como um todo. Tendo em vista estes estudos e o estudo de caso desenvolvido e descrito no capítulo 5, esta pesquisa propõe um modelo de referência para DDS. Este modelo tem por objetivo identificar as características dos projetos em ambientes de DDS e servir de apoio ao desenvolvimento de software realizado por equipes de trabalho heterogêneas e geograficamente dispersas.

O modelo é composto de variáveis críticas, identificando o relacionamento entre elas. Além disso, oferece uma base para a condução de projetos de DDS, visando (1) facilitar a identificação de fraquezas e planejar melhorias visando minimizar possíveis problemas, (2) garantir que projetos de DDS sejam viáveis com grupos de diferentes níveis de capacidade, e (3) aprimorar a capacidade da organização como um todo. A seguir descreve-se o modelo proposto, denominado MuNDDoS – **M**at**u**ridade **N**o **D**esenvolvimento **D**istribuíd**o** de **S**oftware.

6.1 MUNDDOS – MATURIDADE NO DDS

O modelo de referência (MuNNDoS) foi elaborado para atuar como facilitador nos projetos de desenvolvimento distribuído de software. Além disso, a forma como o modelo foi concebido permite a identificação de fraquezas e oportunidades de melhorias nos projetos. Para isso, o modelo sugere a existência de duas dimensões: organizacional e de projetos.

Ampliando a visão relativa ao processo de desenvolvimento de software, buscando adotar uma visão estratégica com relação ao processo, pode-se identificar a etapa de planejamento como sendo a primeira a ocorrer. Esta etapa envolve a definição das estratégias que deverão conduzir o processo de desenvolvimento como um todo, ao longo do tempo (esfera organizacional). Pode-se identificar a etapa de planejamento como sendo preliminar a um conjunto de ciclos de projetos de desenvolvimento de software (esfera de projetos) derivados do processo de planejamento. A figura 6.1 apresenta o modelo proposto.

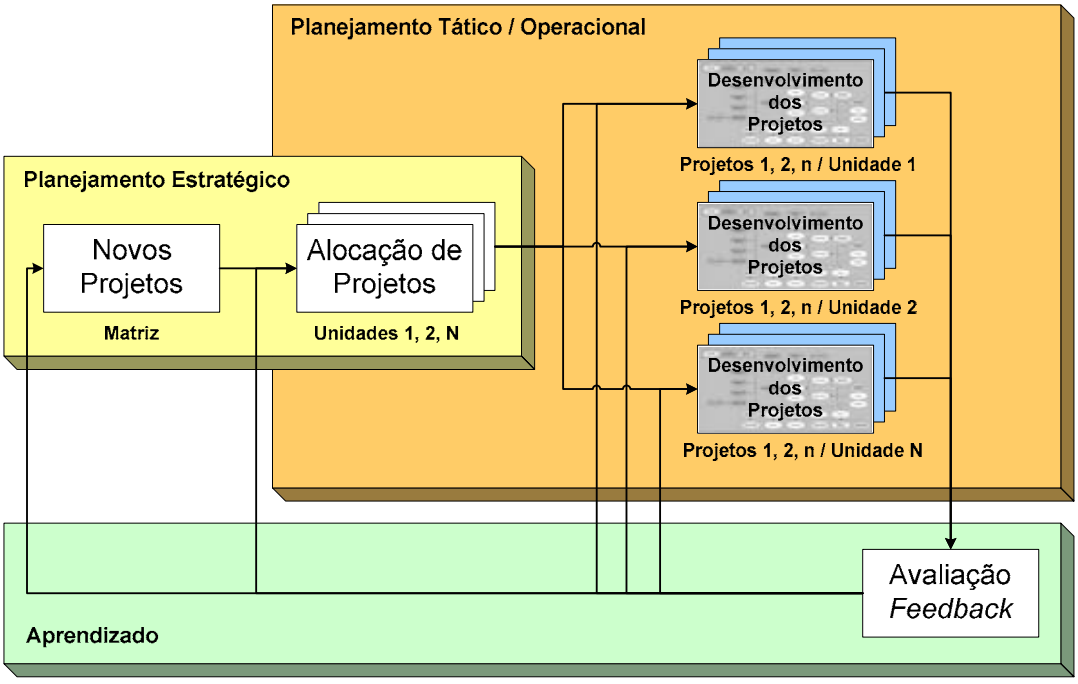

Figura 6.1 – Modelo de referência proposto

Identificam-se dois ciclos de planejamento necessários para a gestão de projetos de DDS. O primeiro envolve o planejamento estratégico. Este é conduzido pela matriz e diz respeito à identificação e priorização de novos projetos a serem desenvolvidos, sejam eles projetos internos (de departamentos internos à organização) ou externos (projetos requisitados por clientes externos). Além disso, cabe aos participantes deste nível de planejamento buscar o alinhamento estratégico entre os objetivos de cada unidade distribuída e a matriz.

O segundo ciclo envolve o planejamento tático-operacional, no âmbito da unidade de desenvolvimento de software distribuída. A transição entre os dois ciclos de planejamento ocorre exatamente na alocação dos projetos, envolvendo atividades de planejamento e definição dos projetos que serão desenvolvidos em cada unidade distribuída, de acordo com políticas de alocação previamente definidas. O planejamento tático é de responsabilidade final (aprovação) dos responsáveis por cada unidade de desenvolvimento, enquanto que o planejamento operacional envolve a gestão do projeto (esfera de projetos), sob responsabilidade do gerente de projeto.

Na esfera de projetos, consideram-se três grandes dimensões de fatores: técnica, não-técnica e híbrida. A dimensão técnica é composta de fatores relacionados a conhecimentos técnicos considerados como base para a construção de software. Já a dimensão não-técnica é composta de fatores relacionados a conhecimentos de áreas relacionadas, necessárias ao desenvolvimento dos projetos, envolvendo fatores sociais, culturais, lingüísticos, comportamentais, entre outros. A dimensão híbrida é composta de fatores que necessitam tanto de conhecimentos técnicos quanto não-técnicos. Todos os fatores, sozinhos ou inter-relacionados, formam uma base para identificar e minimizar possíveis fraquezas e dificuldades encontradas nos projetos.

O último ciclo proposto no modelo de referência é o de aprendizado, relativo à avaliação das atividades realizadas e estratégias adotadas. O modelo sugere a existência de um processo para suportar a coleta de dados, envolvendo a avaliação dos trabalhos realizados, lições aprendidas, etc. Desta forma, o modelo realimenta os ciclos de planejamento. A seguir apresentam-se, em detalhes, as características e atividades de cada processo definido no modelo proposto, tendo por base o *workflow* de representação de processos do RUP [KRU 00] e mapas conceituais [INS 03].

6.1.1 Novos Projetos

Este processo visa identificar e captar novos projetos a serem desenvolvidos pela organização. Para isto, envolve inicialmente uma atividade de prospecção, cujo objetivo é identificar clientes e seus respectivos projetos. Uma vez identificados os projetos, as atividades seguintes dizem respeito à captação e análise estratégica dos mesmos. Para isso, deve-se realizar uma filtragem nos clientes e projetos previamente identificados, direcionando os esforços no sentido analisá-los contra os objetivos da organização. Se for possível, devem-se consultar dados e experiências anteriores.

A análise realizada deve verificar o enquadramento dos projetos captados à estratégia existente, decidindo se existe vantagem no desenvolvimento dos projetos pela organização. O processo de Novos Projetos é necessário independente da relação

existente entre a organização e seus clientes (clientes internos ou clientes externos). A figura 6.2 apresenta o *workflow* correspondente.

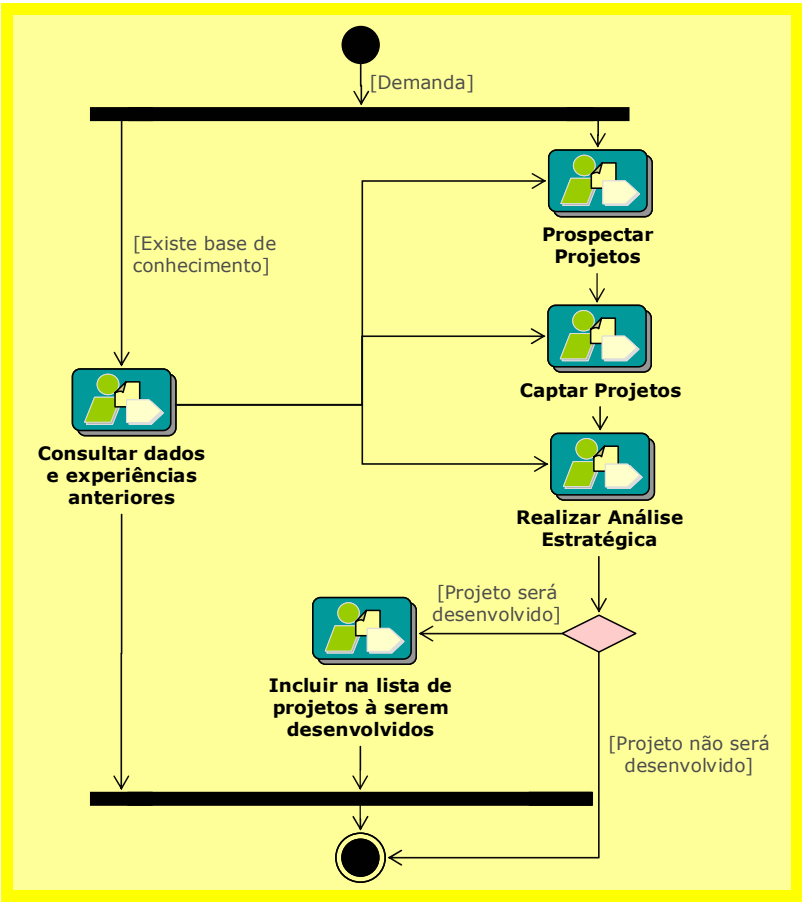

Figura 6.2 – Processo de Novos Projetos

O quadro 6.1 apresenta a fonte de onde foi identificada cada atividade.

Quadro 6.1 – Processo de Novos Projetos

Atividade	Fonte
Prospecção	Teoria
Captação	Teoria
Análise Estratégica	Teoria
Lista de Projetos	EC1, EC2
Experiências Anteriores	EC1

A saída do processo de Novos Projetos é uma lista de projetos a serem desenvolvidos. A figura 6.3 identifica a entrada e a saída deste processo.

Figura 6.3 – Entrada e saída de Novos Projetos

6.1.2 Alocação de Projetos

A Alocação de Projetos (Figura 6.4) é um processo que envolve a seleção de projetos que serão desenvolvidos em cada unidade distribuída, de acordo com uma política de alocação definida pela organização. A alocação tem por base a lista de projetos gerada no processo de Novos Projetos. De modo a auxiliar este processo, sugere-se a seguir um processo de alocação baseado em 3 passos (P): (P1) análise preliminar, (P2) análise do risco e do benefício da distribuição e (P3) seleção da unidade (envolvendo a análise do risco e do benefício de cada unidade). Salienta-se que este processo atua como um suporte ao processo decisório e não toma as decisões propriamente ditas. O principal objetivo é tornar o processo decisório mais objetivo.

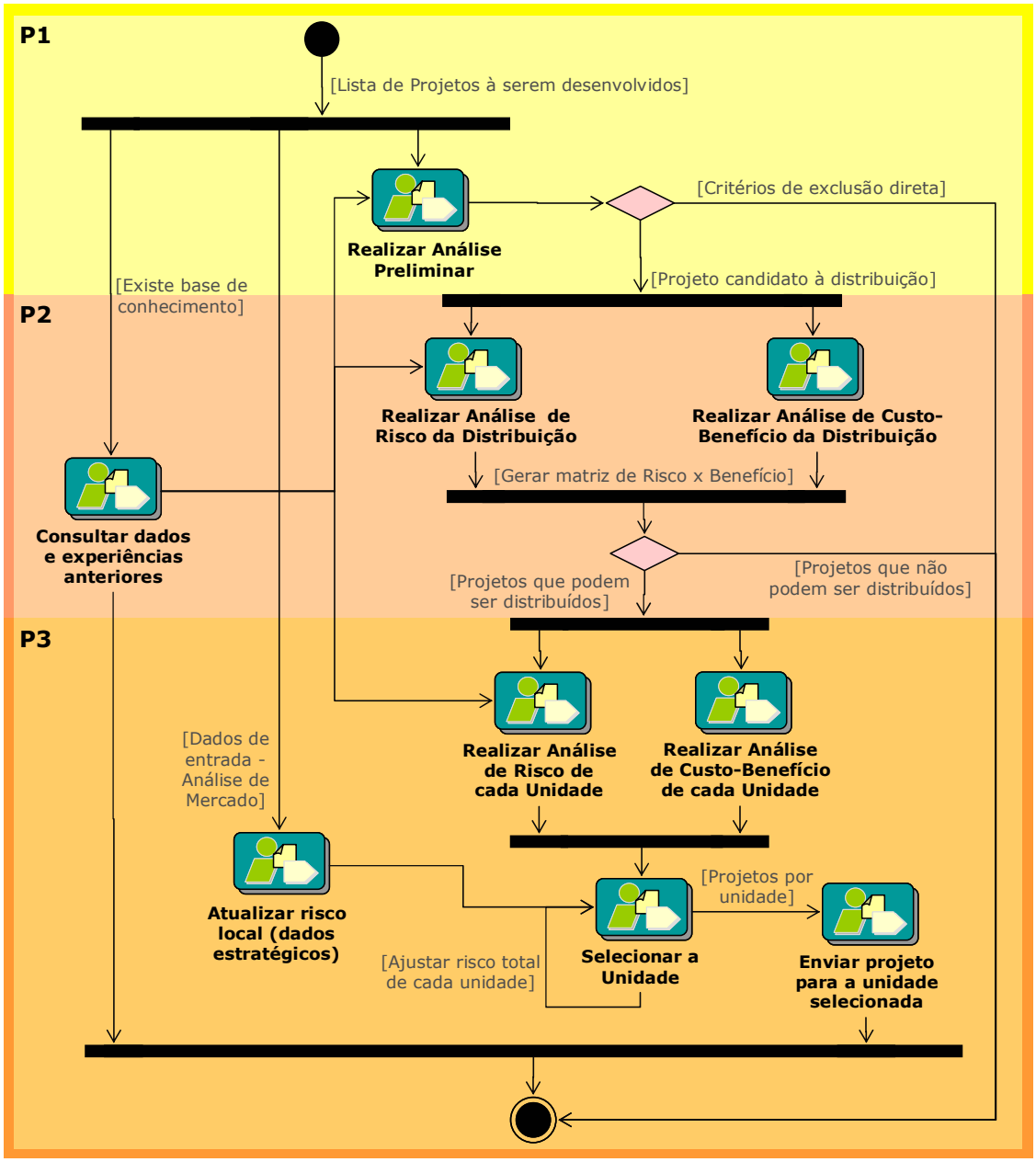

Figura 6.4 – Alocação de Projetos

O quadro 6.2 apresenta a fonte de onde se identificou cada passo.

Quadro 6.2 – Processo de Alocação de Projetos

Passo	Fonte
Análise Preliminar	EC2
Análise de Risco x Benefício da distribuição	Teoria, EC2
Seleção da unidade	Teoria, EC1, EC2

A seguir detalha-se cada passo do processo de alocação:

6.1.2.1 Passo 1 – Análise Preliminar

A lista de projetos alimentada no processo de **Novos Projetos** deve ser percorrida de modo a identificar todos os projetos que se enquadram na categoria de desenvolvimento distribuído de software. Para isto, o primeiro passo proposto é o de realizar uma análise preliminar em cada projeto, verificando se existe algum critério de exclusão direta, ou seja, algum critério que rejeita, em primeira análise, o desenvolvimento do respectivo projeto de forma distribuída. Estes critérios podem variar de organização para organização, mas alguns pontos que devem ser considerados são:

- **Restrições de exportação**: este critério diz respeito à existência de restrições relacionadas à exportação nas leis de determinados locais. Isto é importante no sentido de identificar se nos locais onde estão as unidades distribuídas possam existir normas que dificultem o DDS.
- **Privacidade dos dados**: deve-se considerar se o projeto é crítico do ponto de vista de privacidade de dados, isto é, se necessita enviar dados críticos (sob o ponto de vista da matriz) para as unidades distribuídas;
- **Propriedade intelectual**: deve-se considerar se o projeto contém um grande valor de propriedade intelectual e quais as regras existentes no local onde se localiza a unidade distribuída em relação a isto;
- **Disponibilidade de ambiente físico**: deve-se avaliar se o projeto necessita de, por exemplo, a replicação de algum tipo de ambiente físico que é inviável de ser disponibilizado nas unidades dispersas;
- **Restrições de segurança**: deve-se avaliar a existência de algum comprometimento em relação ao envio do projeto e os dados relacionados para a unidade fisicamente dispersa;
- **Tipo de *engagement***: deve-se avaliar o tipo de *engagement* relacionado ao projeto (manutenção, novo projeto, alterações e/ou melhorias em projetos existentes, suporte, entre outros) e se as unidades distribuídas estão em condições de receber um determinado tipo de projeto.

Ao final deste passo, deve ser identificada uma lista de projetos que são candidatos a serem desenvolvidos de forma distribuída e uma outra lista de projetos que não seguirão no processo de alocação por não se enquadrar no DDS. A figura 6.5 apresenta a entrada e a saída do passo 1.

Figura 6.5 – Entrada e saída do passo 1

A lista de projetos que não serão distribuídos indica os projetos sugeridos para a organização desenvolver de forma tradicional, ou seja, com recursos co-localizados.

6.1.2.2 Passo 2 – Análise do Risco e do Custo-Benefício da distribuição

Uma vez identificados os projetos que são candidatos a serem desenvolvidos de forma distribuída, se faz necessária uma análise do risco e do benefício da distribuição de cada um. O objetivo deste passo é identificar os projetos que se enquadram nos objetivos da distribuição, não sendo considerado neste momento a unidade que o desenvolverá.

No que tange a análise de risco, sugere-se quantificar um fator de risco ponderando os seguintes critérios:

- **Nível de documentação existente**: avaliar a documentação disponível e o grau de interação necessária com os atores envolvidos para elaborar a documentação correspondente ao projeto (ênfase na documentação que descreve o escopo do projeto e no documento de requisitos);
- **Clareza e estabilidade dos requisitos**: verificar o quão estáveis e claros estão os requisitos do projeto, sob o ponto de vista do negócio, e a interação necessária com os atores envolvidos;
- **Riscos de tecnologia**: verificar as tecnologias disponíveis nas unidades e a existência de profissionais capacitados para lidar com as mesmas;
- **Experiência dos atores em projetos distribuídos**: avaliar a experiência dos atores envolvidos (clientes e usuários, por exemplo), no desenvolvimento de projetos de DDS;
- **Capacidade de controle**: avaliar a capacidade gerencial dos responsáveis pelas atividades de gerência do projeto por parte dos atores envolvidos, focando na experiência em gerenciar projetos distribuídos;

- **Complexidade e duração do *engagement***: verificar o quão complexo será o *engagement* e quando ele iniciará. Isto implica necessariamente na existência de recursos humanos disponíveis para integrar a equipe do projeto e no tipo de projeto que será desenvolvido;
- **Tamanho do projeto**: verificar o tamanho do projeto (normalmente a quantidade de membros na equipe) e as implicações de ser desenvolvido de forma distribuída;

Por sua vez, a análise de benefício está centrada no custo de desenvolver o projeto de forma distribuída. Neste caso, devem ser considerados critérios tais como o percentual de esforço que será necessário nas unidades fisicamente dispersas e a necessidade de haver recursos humanos junto do cliente e/ou usuário. Com isto, deve-se calcular o benefício da distribuição, comparando o custo de se desenvolver de forma centralizada e de forma distribuída. Um critério a ser considerado é o custo do percentual de *overhead* gerencial que a distribuição do projeto poderia causar.

Ao final deste passo, deve ser identificada uma lista de projetos que estão aptos (possuem benefício e risco considerados razoáveis) a serem desenvolvidos de forma distribuída. A figura 6.6 apresenta a entrada e a saída do passo 2.

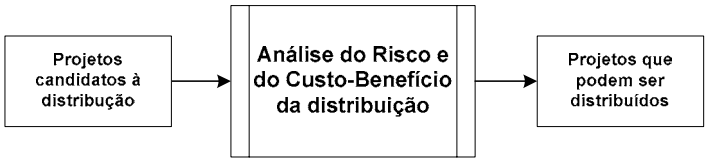

Figura 6.6 – Entrada e saída do passo 2

6.1.2.3 Passo 3 – Selecionar Unidade

O terceiro e último passo do processo de alocação está centrado na escolha da melhor unidade para receber o projeto. Neste sentido, deve-se selecionar a melhor forma de distribuí-lo. Para isto, deve ser feita uma a análise de risco e benefício idêntica à realizada no passo anterior, com a exceção de ser uma análise mais específica e envolver outros critérios e características das unidades. Deve-se lembrar que até o passo 2 não estava sendo levado em conta características específicas das unidades distribuídas existentes. O objetivo era apenas saber, sob o ponto de vista da matriz da organização, se o projeto apresentava características de ser desenvolvido de forma distribuída.

No que tange a análise de risco, sugere-se quantificar um fator de risco para cada unidade existente, considerando os seguintes critérios:

- **Capacidade e experiência da unidade em projetos similares**: verificar se a unidade analisada já participou do desenvolvimento de

algum projeto similar ao que está sendo analisado e a experiência da equipe;

- **Existência de algum centro de competência na tecnologia requerida no projeto**: verificar se a unidade analisada possui algum centro de competência (ex: todos os projetos em *Java*[10] deverão ser desenvolvidos em Pernambuco) para o desenvolvimento do projeto;
- **Disponibilidade de recursos humanos**: verificar se existem recursos humanos disponíveis para o desenvolvimento do projeto na unidade analisada, ou se é necessário contratar ou treinar novos colaboradores;
- **Tempo necessário para treinar novos colaboradores**: no caso de contratação ou treinamento de novos colaboradores, deve-se quantificar o tempo necessário para as respectivas atividades, informando em quanto tempo a equipe estará pronta;
- **Espaço físico disponível**: verificar se existe espaço disponível na unidade analisada, caso seja necessária a contratação de novos colaboradores;
- **Fator de *turn-over***: identificar o fator de *turn-over* existente na unidade analisada, avaliando o risco de colaboradores saírem no meio do projeto;
- **Barreiras de idioma**: verificar a existência de dificuldades com relação ao idioma a ser utilizado no desenvolvimento do projeto;
- **Barreias de fuso-horário**: verificar a existência de dificuldades com relação à diferença de fuso-horário entre os locais que estarão envolvidos no desenvolvimento do projeto, principalmente no que diz respeito à realização de reuniões e comunicação síncrona.

Em relação à análise do benefício, esta envolve um cálculo mais específico do real custo de desenvolver o projeto em cada unidade e a identificação da unidade que apresenta os menores valores. Neste caso, deve ser considerado também o custo do percentual de *overhead* gerencial na unidade de desenvolvimento distribuída.

Assim, têm-se dados para decidir se, de acordo com a análise realizada, existem vantagens de se enviar um determinado projeto para uma unidade distribuída. De forma a sofisticar ainda mais o processo, sugere-se no passo 3 a utilização de uma matriz que relaciona o risco e o benefício, de modo a entender graficamente os benefícios de cada unidade. Adicionalmente, como contribuição do estudo, também se sugere o cálculo de um fator de risco geral de cada unidade, chamado de risco de concentração.

---

[10] *Java* – Linguagem de programação orientada a objetos, desenvolvida pela *Sun Microsystems Inc*. (http://wwws.sun.com/software/learnabout/java/).

Este fator é representado pela média ponderada do número de projetos existentes em cada unidade e o um fator de risco local (risco de cada cidade, estado ou país). A sub-seção a seguir explica em detalhes o risco de concentração.

6.1.2.3.1 Risco de Concentração de Projetos

No passo 3, além das atividades descritas anteriormente, propõe-se a introdução do conceito de risco de concentração. Em linhas gerais, é uma análise da quantidade de projetos já existentes em uma determinada unidade distribuída e a ponderação com o risco do local onde a unidade se encontra. O risco de concentração pode ser utilizado para auxiliar na identificação da sobrecarga de projetos em uma determinada unidade e na identificação de riscos fora do escopo do projeto (características específicas do local onde está uma determinada unidade, que pode refletir diretamente no desenvolvimento dos projetos). Para isto, no *workflow* apresentado existem duas atividades relativas ao risco de concentração: a atualização de dados referentes à análise de mercado (risco local) e o ajuste do risco total de cada unidade cada vez que um novo projeto é selecionado.

A título de exemplo, considera-se uma organização com duas unidades de desenvolvimento (UD1 e UD2) fisicamente dispersas da matriz e localizadas cada uma em um determinado país. Neste caso, o risco de concentração deve considerar o risco de cada país onde as unidades estão localizadas e o número de projetos em desenvolvimento que cada unidade possui. O risco de cada país considera critérios tais como política, economia, taxas, segurança, podendo ainda conter outros critérios de acordo com a política de cada organização. Sendo assim, o risco de concentração é a ponderação entre o número de projetos sendo desenvolvidos em cada unidade e o risco de cada país. Os exemplos 1 e 2 (Tabelas 6.1 e 6.2) a seguir sugerem uma forma de calcular este risco, considerando as duas unidades UD1 e UD2:

Tabela 6.1 – Risco de concentração (exemplo 1)

		UD1	**UD2**
(C)	Coeficiente de risco do país (ou região)	4.0	2.4
(N)	Número de projetos em desenvolvimento (N)	**10**	**5**
(P)	Ponderação dos projetos com o risco de cada país (C x N)	40	12
(T)	Total de projetos, considerando a ponderação com o risco	52	

O coeficiente de risco do país (C) deve receber um valor referente à análise de mercado de cada país. Sendo assim, para chegar ao valor do risco de concentração é necessário apenas verificar o percentual de projetos que cada unidade possui, considerando o coeficiente do risco do país, assim:

- Risco de concentração (**UD1**) = P (**UD1**) / (T) = 77%
- Risco de concentração (**UD2**) = P (**UD2**) / (T) = 23%

Percebe-se que, considerando o risco do país, a UD1 está concentrando 77% dos projetos enquanto que a UD2 concentra apenas 23%. Se não fosse considerado o risco do país, a UD1 teria 67% dos projetos e a UD2 33%. O objetivo deve ser sempre buscar um valor balanceado, de modo que em locais com um risco alto exista um número menor de projetos e vice-versa. Baseando-se no mesmo exemplo explicado anteriormente, apenas invertendo o número de projetos em desenvolvimento em cada país, tem-se o seguinte resultado:

Tabela 6.2 – Risco de concentração (exemplo 2)

		UD1	**UD2**
(C)	Coeficiente de risco do país (ou região)	4.0	2.4
(N)	Número de projetos em desenvolvimento (N)	**5**	**10**
(P)	Ponderação dos projetos com o risco de cada país (C x N)	20	24
(T)	Total de projetos, considerando a ponderação com o risco	44	

- Risco de concentração (**UD1**) = P (**UD1**) / (T) = 45%
- Risco de concentração (**UD2**) = P (**UD2**) / (T) = 55%

Neste caso, a UD1, que possui o maior risco local, recebeu um menor número de projetos, ficando com um risco de concentração de 45%. Por sua vez, a UD2 ficou com 55% de risco. Se fosse considerada apenas a quantidade de projetos, a UD1 teria apenas 33% de risco contra 67% da UD2. Sendo assim, no processo de alocação de projetos, ao selecionar um projeto para uma determinada unidade, o risco de concentração deve sempre ser atualizado para refletir a realidade da empresa.

Os exemplos apresentados consideraram apenas o número de projetos existentes em cada unidade. De modo a obter um valor mais preciso para o cálculo do risco de concentração, o número total de projetos em desenvolvimento em cada unidade poderia ser substituído pelo total de risco existente em cada uma. Com relação aos valores referentes ao risco local, os dados de diversos países podem ser encontrados em pesquisas de entidades oficiais, tais como o WMRC[11] (*World Market Research Council*).

Ao final do passo 3, levando-se em consideração todos os critérios apresentados e o risco de concentração de cada unidade, deve ser identificada a unidade que melhor se enquadra nos objetivos de cada projeto. Isto fica caracterizado com a atividade de seleção da unidade e o envio do projeto para a mesma. Além disso, uma atividade importante e prevista para estar sempre presente no processo de alocação é a consulta a dados e experiências anteriores (base de conhecimento), caso a organização possuir. A figura 6.7 apresenta a entrada e a saída do passo 3.

[11] WMRC – Organização privada responsável por fornecer informações sobre o mercado em diversos países no mundo. (http://www.wmrc.com/).

Figura 6.7 – Entrada e saída do passo 3

Apesar de ser uma prática já adotada em diversas empresas globalmente [CAR 99], [KAR 98], [HER 01a], não se aprofundou neste estudo a possibilidade de se dividir o desenvolvimento de um projeto para mais de uma unidade, caracterizando o desenvolvimento conhecido por *follow-the-sun* (seção 2.4). Isto se deve principalmente a não adoção desta prática nas organizações estudadas, fazendo com que não existam dados suficientes para a realização de um estudo mais específico.

6.1.3 Desenvolvimento dos Projetos

Seguindo o modelo proposto, o desenvolvimento dos projetos distribuídos envolve a identificação de diversos fatores que podem dificultar ou até mesmo comprometer a execução dos projetos com sucesso. Este é um ponto crítico do modelo, pois se sabe que mesmo após diversas análises de risco e benefício, os projetos estão sujeitos a imprevistos e a problemas. Cabe salientar que esta etapa do modelo não está focada na definição de um processo em si, no caso o processo de desenvolvimento de software, mas sim na identificação dos fatores envolvidos em projetos de DDS (ver seções 1.1 e 1.2 – objetivos do estudo e motivação). De modo a minimizar eventuais dificuldades, identificaram-se cinco categorias de fatores que estão presentes em projetos de DDS (Figura 6.8). Cada uma destas categorias possui fatores relacionados e todas podem indicar um **fator de dispersão** de um projeto distribuído. Para cada um dos fatores identificados, pode-se trabalhar na mensuração da dispersão existente como um todo e analisar os fatores que representam o maior grau de dispersão.

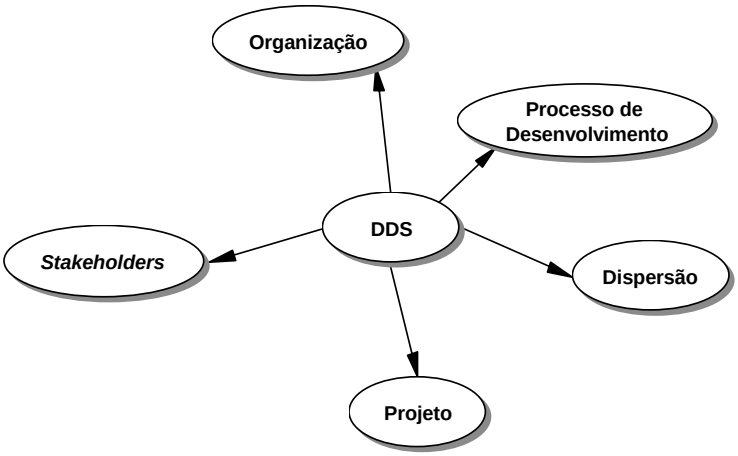

Figura 6.8 – Categorias de fatores identificadas

A categoria **Projeto** representa todos os fatores relacionados com as características e atividades em um projeto. A categoria **Processo de Desenvolvimento** representa os fatores relacionados com o processo existente para o correto desenvolvimento dos projetos. A categoria **Dispersão** representa os fatores relacionados com a distância existente. A categoria **Organização** representa os fatores relacionados com a organização que refletem no desenvolvimento dos projetos. A categoria ***Stakeholders*** representa todos os fatores relacionados com as pessoas interessadas e envolvidas nos projetos.

O quadro 6.3 apresenta a fonte de onde foi identificada cada categoria:

Quadro 6.3 – Categorias de fatores identificadas

Categoria	Fonte
Processo de Desenvolvimento	EC1, EC2, Teoria
Projeto	EC2, Teoria
Stakeholders	EC1, EC2, Teoria
Organização	EC1
Dispersão	Teoria

Para cada uma das categorias, foram identificados os fatores associados. A seguir apresentam-se os fatores classificados em cada categoria e a fonte de onde cada fator foi identificado.

6.1.3.1 Categoria Processo de Desenvolvimento

Nesta categoria identificarem-se os seguintes fatores (Figura 6.9):

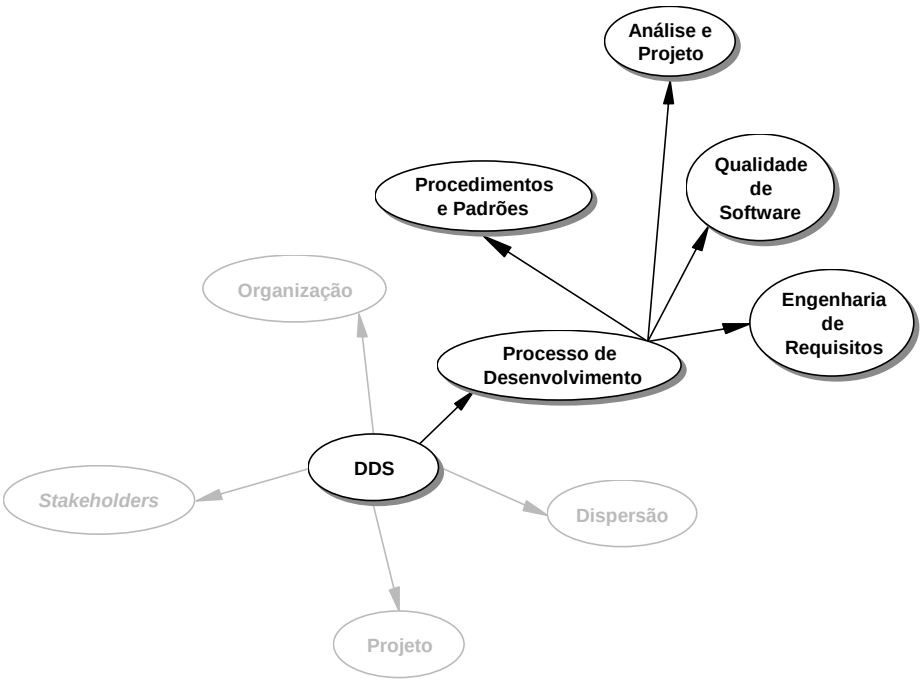

Figura 6.9 – Fatores relacionados ao processo de desenvolvimento

Os fatores identificados na categoria Processo de Desenvolvimento envolvem apenas aqueles que foram considerados os principais dentro de projetos distribuídos. Sendo assim, foram identificados: a análise e o projeto (modelagem), com foco na arquitetura e modularização; a engenharia de requisitos, considerada uma das principais atividades e uma área chave para o sucesso de projetos de DDS; a qualidade de software, identificada como uma atividade essencial para manter os padrões das equipes sempre de acordo com os da organização; e os procedimentos e padrões existentes, pois apesar de existir um processo bem definido, cada *stakeholder* pode ter a sua própria forma de participar do projeto, evitando respeitar procedimentos e padrões previamente definidos.

O quadro 6.4 apresenta a fonte de onde foi identificado cada fator:

Quadro 6.4 – Fatores relacionados ao processo de desenvolvimento

Fatores	Fonte
Análise e Projeto	Teoria
Engenharia de Requisitos	EC1, EC2, Teoria
Qualidade de Software	Teoria
Procedimentos e Padrões	EC1, EC2, Teoria

6.1.3.2 Categoria Dispersão

Nesta categoria identificarem-se os seguintes fatores (Figura 6.10):

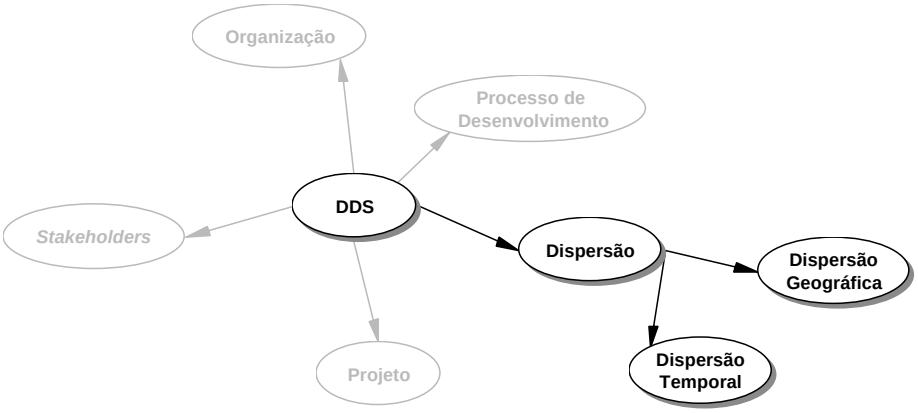

Figura 6.10 – Fatores relacionados à dispersão

Na categoria Dispersão identificaram-se dois fatores que podem exercer um grande impacto na forma como os projetos distribuídos são concebidos. São eles: dispersão temporal, caracterizada pelas diferenças de fuso-horário existentes e suas conseqüências para o desenvolvimento dos projetos; e a dispersão geográfica, caracterizada pela distância física entre os stakeholders, fator este que foi aprofundado no capítulo 4 desta dissertação.

O quadro 6.5 apresenta a fonte de onde foi identificado cada fator:

Quadro 6.5 – Fatores relacionados à dispersão

Fatores	Fonte
Dispersão Temporal	EC2, Teoria
Dispersão Geográfica	EC1, EC2, Teoria

6.1.3.3 Categoria Organização

Nesta categoria identificaram-se os seguintes fatores (Figura 6.11):

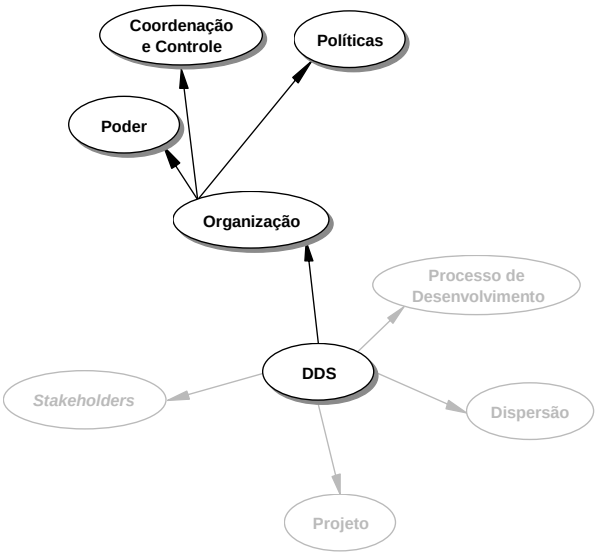

Figura 6.11 – Fatores relacionados à organização

Os fatores da categoria Organização envolvem características relacionadas com a cultura organizacional. Neste sentido, foram considerados o poder, as políticas existentes e a coordenação e controle. O poder pode interferir em projetos de DDS dependendo dos impactos que ele possui para a organização (estrutura organizacional fraca, forte ou balanceada). Com relação às políticas, elas são responsáveis por definir a forma como os valores da organização são mantidos. Por último, a coordenação e controle dizem respeito à maneira de controlar as equipes, os artefatos e os projetos como um todo.

O quadro 6.6 apresenta a fonte onde foi encontrado cada fator identificado nesta categoria.

Quadro 6.6 – Fatores relacionados à organização

Fatores	Fonte
Coordenação e controle	EC1, Teoria
Poder	Teoria
Políticas	EC1, EC2

6.1.3.4 Categoria Stakeholders

Nesta categoria identificarem-se os seguintes fatores (Figura 6.12):

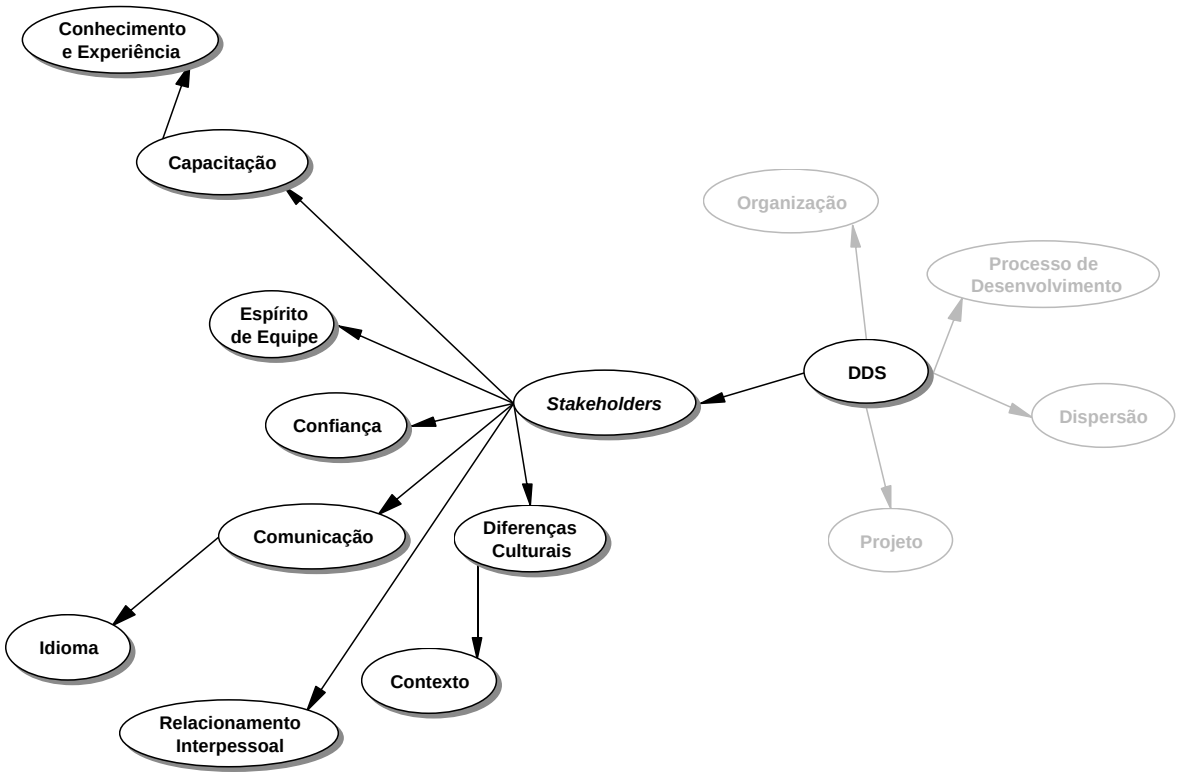

Figura 6.12 – Fatores relacionados aos *stakeholders*

Os fatores da categoria *Stakeholders* envolvem aspectos relacionados com as pessoas envolvidas nos projetos de DDS. Desta forma, questões tais como o espírito de equipe, confiança e relacionamento interpessoal têm uma grande importância no sucesso dos projetos. Em paralelo as diferenças culturais, o contexto, a comunicação e o idioma são fatores podem gerar diferenças significativas entre os *stakeholders*, dificultando o trabalho. Por último, a capacitação, o conhecimento e a experiência podem ser os balizadores de algumas das dificuldades existentes.

O quadro 6.7 apresenta a fonte de onde foi identificado cada fator:

Quadro 6.7 – Fatores relacionados às pessoas

Fatores	Fonte
Capacitação	EC1, EC2
Conhecimento e experiência	EC1, EC2
Espírito de Equipe	EC1
Diferenças culturais	EC2, Teoria
Contexto	EC2, Teoria
Confiança	EC1, EC2, Teoria
Comunicação	EC1, EC2, Teoria
Idioma	EC1, EC2, Teoria
Relacionamento Interpessoal	EC1, EC2

6.1.3.5 Categoria Projetos

Nesta categoria identificarem-se os seguintes fatores (Figura 6.13):

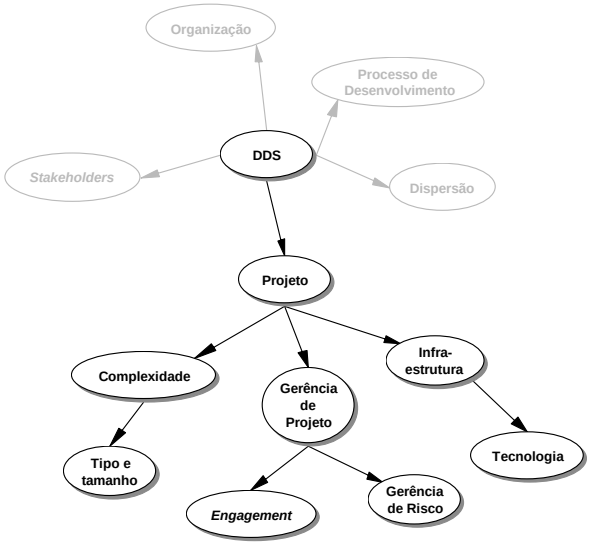

Figura 6.13 – Fatores relacionados ao projeto.

A última categoria identificada, Projetos, diz respeito a aspectos importantes do projeto como um todo. Envolve características tais como a complexidade, o tipo e o tamanho do projeto. Além disso, a infra-estrutura e tecnologias disponíveis podem ser o diferencial para projetos de DDS, principalmente no que diz respeito às tecnologias de colaboração (*groupware*, por exemplo). O último fator identificado nesta categoria está relacionado com a existência de modelos para uma efetiva gerência dos projetos de DDS. Como parte da gerência de projeto, dois fatores emergem como de grande importância: a gerência de risco e o *engagement*. A gerência de risco foi considerada nos estudos de caso realizados uma grande influência no sucesso ou fracasso dos projetos. Por sua vez, o *engagement* é considerado como o marco do início dos projetos de DDS, pois é o momento onde as equipes distribuídas são integradas, padronizações e formas de trabalho únicas são incorporadas e é onde o projeto começa.

O quadro 6.8 apresenta a fonte de onde foi identificado cada fator:

Quadro 6.8 – Fatores relacionados ao projeto

Fatores	Fonte
Complexidade	Teoria
Tipo e tamanho	EC1, EC2, Teoria
Gerência de Projeto	EC1, EC2, Teoria
Gerência de Risco	EC1, EC2, Teoria
Engagement	EC2
Infra-estrutura	EC2, Teoria
Tecnologia	Teoria

Com a identificação das categorias e os fatores envolvidos em cada uma, foi possível consolidar um diagrama identificando todos os fatores envolvidos em projetos de DDS. A figura 6.14 apresenta um diagrama de influência com esta consolidação, separando os fatores em técnicos (esferas cinzas), não-técnicos (esferas pretas) e híbridos (esferas listradas), caracterizando assim as três respectivas dimensões. As esferas brancas representam as categorias principais identificadas.

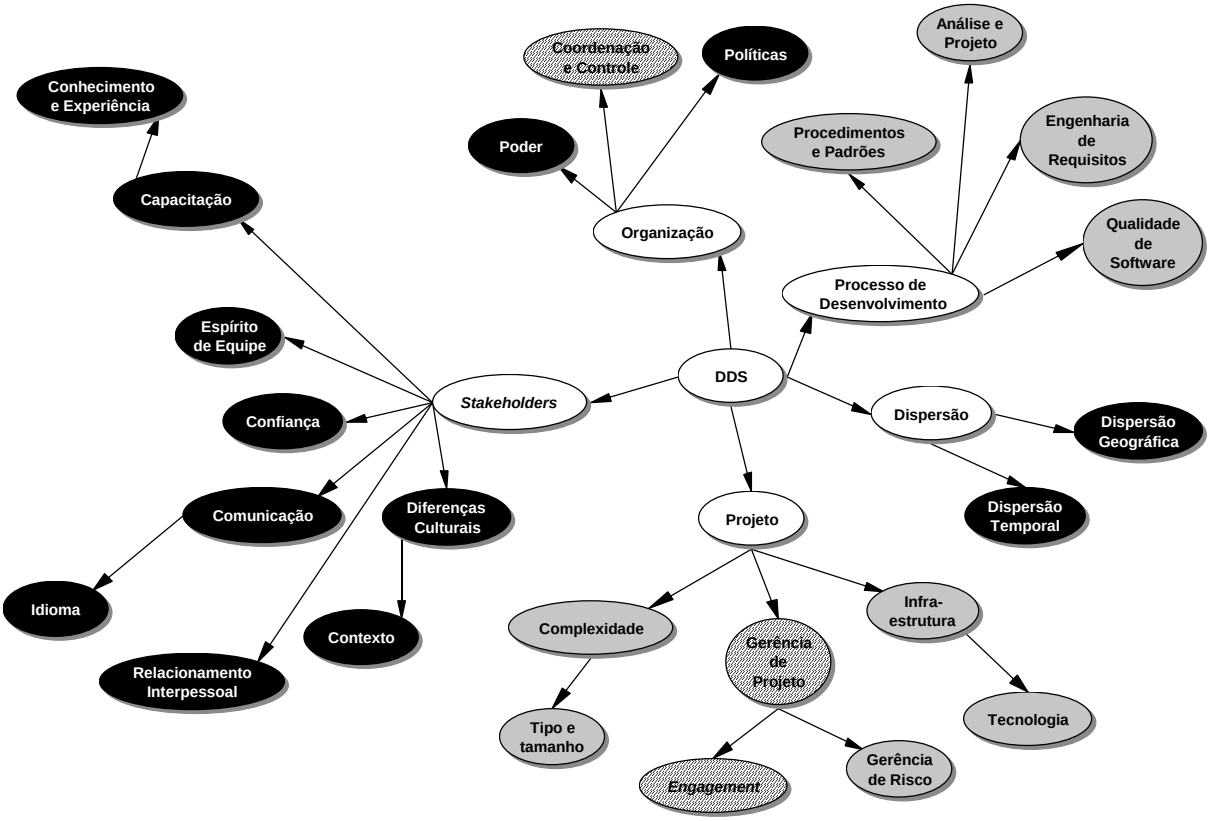

Figura 6.14 – Consolidação dos fatores identificados

Conforme mencionado anteriormente, a esfera de projetos, caracterizada pelo desenvolvimento dos mesmos, não estava focada na definição de um processo. Neste sentido, a utilização de mapas conceituais se deve ao fato de este estudo ter como principal objetivo a identificação dos fatores e a relação entre eles. Assim, mapas conceituais podem prover uma representação gráfica simples dos principais elementos identificados e fica claro quais os conceitos envolvidos e a categoria diretamente afetada [INS 03]. Pode-se ainda avaliar a necessidade de ajustes no desenvolvimento dos projetos, em decorrência da existência ou da inexistência de algum fator.

6.1.4 Avaliação e Feedback

O ultimo ciclo do modelo proposto diz respeito ao aprendizado, sendo caracterizado por um processo de **Avaliação e *Feedback***. Este processo é importante à medida que se busca sempre uma melhoria no processo de desenvolvimento e no

produto final. Por estar em ambientes fisicamente dispersos, é muito importante avaliar e analisar criticamente as estratégias e o processo adotado, contando com a participação de todos os envolvidos. Especificamente no processo de desenvolvimento de software, verificou-se que alguns processos tais como o RUP e o MSF possuem atividades de avaliação e aprendizado com relação ao trabalho realizado. De qualquer forma, quando se encontram mecanismos de avaliação eles se referem basicamente aos testes de software e verificação de conformidade com padrões de qualidade específicos da área de engenharia de software (CMM, normas ISO, entre outros). Sendo assim, não existe uma fase específica onde se possa fazer uma avaliação completa relacionando com os ciclos do modelo proposto. Por este motivo, o processo de **Avaliação e *Feedback*** (Figura 6.15) visa avaliar as estratégias adotadas e os resultados obtidos em cada projeto desenvolvido, tendo por base a relação entre o processo de negócio que será suportado pelo sistema, o próprio sistema e a tecnologia utilizada.

Este processo deve ser executado ao final dos projetos, fornecendo uma retroalimentação para os processos dos ciclos iniciais. Ou seja, a conclusão do processo de desenvolvimento de software dá-se pelo processo de avaliação, estabelecendo o mecanismo de *feedback* para os ciclos de planejamento definidos anteriormente (primeiro e segundo ciclo).

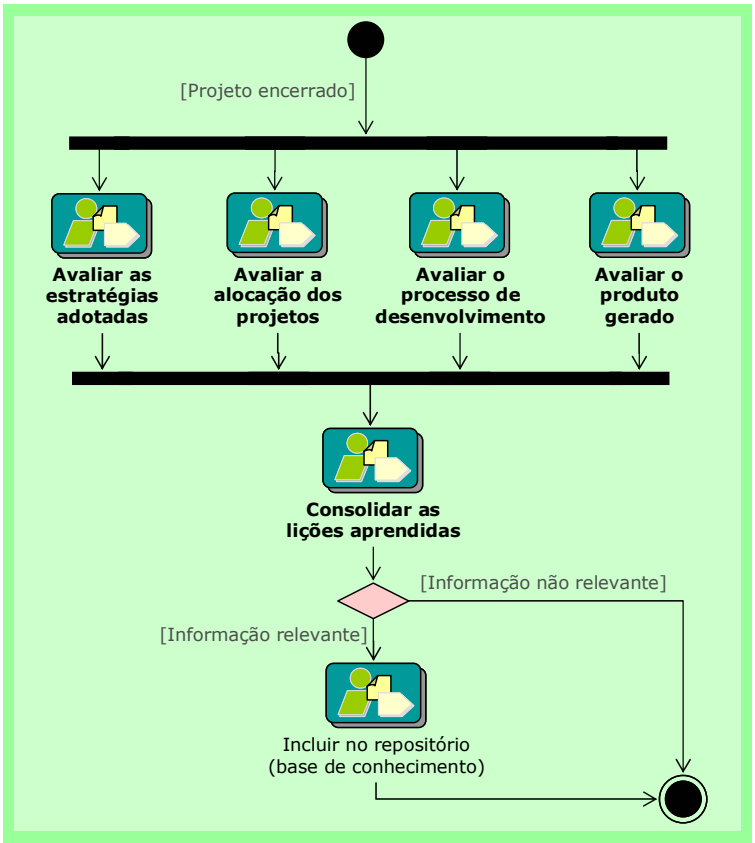

Figura 6.15 – Processo de Avaliação e *Feedback*

O quadro 6.9 apresenta a fonte de onde foi identificada cada atividade:

Quadro 6.9 – Processo de Avaliação e *Feedback*

Fatores	Fonte
Avaliar estratégias	Teoria
Avaliar alocação	Teoria
Avaliar processo	EC1, EC2
Avaliar produto	EC1, EC2
Lições aprendidas	Teoria
Base de conhecimento	EC2, Teoria

Sendo assim, o processo de **Avaliação e *Feedback*** tem por objetivo avaliar as estratégias adotadas, a alocação dos projetos, o processo de desenvolvimento e o produto gerado, considerando os riscos envolvidos e as lições aprendidas desde a prospecção até a entrega do projeto. Deve-se identificar o grau de satisfação dos *stakeholders*, determinando pontos fortes e fracos do processo como um todo.

A quantificação do resultado da avaliação é utilizada como indicativo para determinar pontos fortes e fracos apresentados durante o ciclo de vida do projeto. Os resultados dos dados compilados devem ser utilizados para a formação de uma base de conhecimento que potencialmente atuará como um vetor de aprendizagem contínua, direcionando para melhores resultados nos próximos projetos.

6.1.5 Estágios de Capacidade

Ao recuperar os dados identificados nos estudos de caso desta pesquisa (Organizações 1 e 2), percebe-se claramente que a capacidade da organização em DDS se desenvolve ao longo do tempo, com o ganho constante de experiência. Das duas organizações estudadas, a primeira tem atuado em projetos de DDS há quatro anos, enquanto que a segunda tem atuado há apenas um ano. Nas entrevistas ficaram claras algumas diferenças no nível de maturidade de atuação em ambientes de DDS, tais como as características do processo de desenvolvimento, a carga de treinamento existente, entre outros.

Neste sentido, analisando o modelo proposto sob o ponto de vista de capacidade em projetos distribuídos, identificam-se quatro estágios de maturidade, de acordo com a experiência em projetos de DDS existente nas organizações. Os estágios propostos baseiam-se nos modelos de maturidade para processos de software [NAW 01], [PAU 93] e para equipes dispersas [CAR 99] existentes na literatura. Os estágios de capacidade propostos baseiam-se no modelo de referência (seção 6.1), e está ilustrado na figura 6.16.

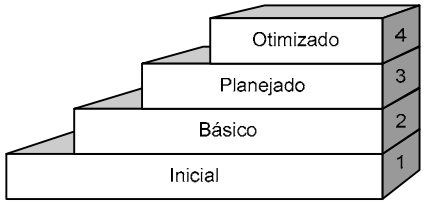

Figura 6.16 – Estágios de capacidade em projetos de DDS

A seguir descreve-se cada estágio e suas principais características:

Estágio 1: Inicial

Neste estágio os projetos distribuídos são desenvolvidos de uma forma desorganizada. Os projetos são considerados únicos e não existe uma maior preocupação em buscar informações de projetos similares ou das equipes envolvidas. Existe apenas o processo de captação de novos projetos a serem desenvolvidos (**Novos Projetos**). A decisão de desenvolver o projeto de forma distribuída é por conveniência. Não existe um planejamento e processo formais, nem uma avaliação das vantagens do desenvolvimento dos projetos em ambientes de DDS. As ações são caracterizadas como sendo reativas.

Estágio 2: Básico

Neste estágio os projetos ainda são considerados únicos e apresentam um nível básico de organização. Os fatores envolvidos na dimensão de projetos (**Desenvolvimento dos Projetos**) passam a ser analisados para minimizar dificuldades. Existe uma tendência para a prevenção de problemas, concentrando-se apenas nos atores envolvidos, sem consulta à experiências anteriores. A decisão de desenvolver o projeto de forma distribuída continua sendo por conveniência. O planejamento e processo formais são realizados somente em nível de projeto. Não existe um processo estabelecido de avaliação e *feedback* (Figura 6.17).

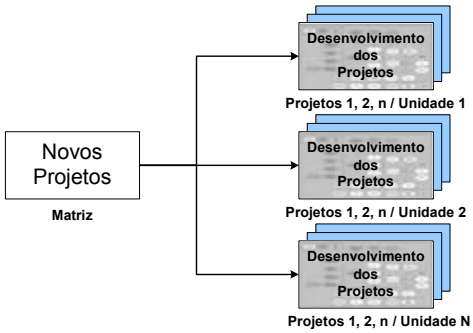

Figura 6.17 – Estágio Básico

Estágio 3: Planejado

Neste estágio inserem-se os ciclos de planejamento estratégico (esfera organizacional) e tático-operacoinal (esfera de projetos). Um projeto não é mais considerado único. Além dos fatores envolvidos na dimensão de projetos, existe um

processo formal para analisar e decidir se existem vantagens de se desenvolver o projeto de forma distribuída (**Alocação de Projetos**). Isto faz com que a abordagem preventiva não seja apenas uma tendência, mas uma realidade. A consulta à experiências anteriores ocorre apenas internamente à cada processo. Neste sentido, não existe um processo estabelecido de avaliação e *feedback*. (Figura 6.18).

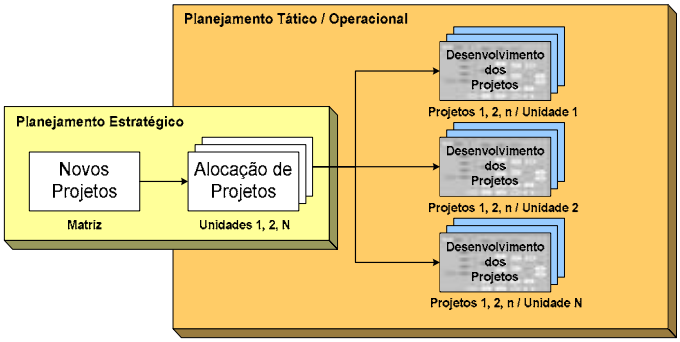

Figura 6.18 – Estágio Planejado

Estágio 4: Otimizado

Neste estágio, insere-se o processo de **Avaliação e *Feedback***. Além do processo de Alocação de Projetos e dos fatores envolvidos na dimensão de projetos, todos os projetos já desenvolvidos geram uma base de conhecimento que será utilizada como subsídio para o desenvolvimento de novos projetos, retroalimentando os ciclos de planejamento estratégico e tático-operacional (Figura 6.19).

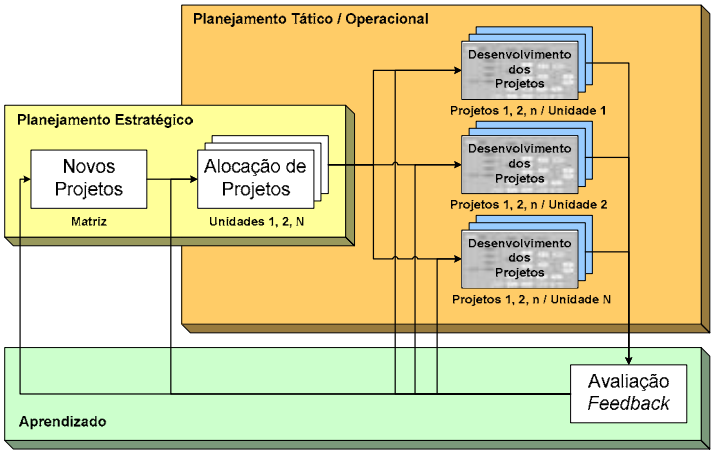

Figura 6.19 – Estágio Otimizado

Como a capacidade diz respeito à maturidade da organização em projetos distribuídos, e tendo por base a forma como o estudo foi concebido, o reconhecimento de uma determinada organização em um estágio em específico deve ser realizado através dos processos descritos nas seções 6.1.1, 6.1.2, 6.1.3 e 6.1.4. Sendo assim, para uma organização ser reconhecida em um estágio de capacidade ela deve satisfazer todas as atividades envolvidas nos processos do respectivo estágio.

7 CONSIDERAÇÕES FINAIS

"Comece pelo começo, e continue até que atinja o fim: então, ao alcançar o fim, pare". Lewis Carroll, 1865.

O software é cada vez mais indispensável para a sociedade moderna [CAR 01a], onde a globalização é uma característica fundamental. Atualmente diversas empresas estão distribuindo seus processos de desenvolvimento de software ao redor do mundo, visando ganhos de produtividade, redução de custos e melhorias na qualidade. Neste contexto, o ambiente de DDS surge como um grande desafio na área de ES.

O DDS tem levado os pesquisadores na área de ES a defrontar-se com a necessidade de novos conhecimentos e uma abordagem mais transdisciplinar. O modelo de referência para DDS proposto é uma tentativa de contribuir na busca de respostas a uma nova classe de problemas que o ambiente de trabalho distribuído apresenta. Os resultados encontrados apresentam indícios de que a área de ES necessita de mais pesquisas voltadas especificamente para esta nova de classe de problemas que o ambiente de DDS está trazendo. Do ponto de vista científico, a legitimação do modelo de referência proposto é decorrente do processo de pesquisa como um todo, representado pelo desenho de pesquisa desdobrado nas diversas fases do estudo, conforme apresentado no capítulo 3.

7.1 CONTRIBUIÇÕES DA PESQUISA

A principal contribuição desta pesquisa é a proposta de um modelo de referência para projetos de desenvolvimento distribuído de software (apresentado no capítulo 6), que contempla as dimensões técnica e não-técnica. O modelo soma-se aos existentes na literatura como mais uma contribuição para responder aos desafios da área de DDS.

Ao longo do processo de formulação do modelo de referência proposto, descreveram-se as características da área de DDS (capítulo 2), identificaram-se os fatores críticos de sucesso e as práticas utilizadas por organizações que atuam nesta área (capítulo 5). Adicionalmente, este estudo contribui com a proposta de um modelo de classificação do nível de dispersão de projetos distribuídos [PRI 02b], [PRI 03a], [PRI 03c] (Capítulo 4), e com a proposta inicial de um modelo de estágios de capacidade de projetos distribuídos baseado no modelo de referência proposto (capítulo 6).

7.2 LIMITAÇÕES DA PESQUISA

Uma das principais limitações da pesquisa refere-se ao número de empresas estudadas na parte empírica do estudo, restringindo a generalização dos resultados obtidos. Deve-se, entretanto, destacar que especificamente os resultados do estudo de caso, principalmente os da categorização dos elementos, foram sustentados na base teórica estudada, o que permite um bom grau de segurança nas conclusões obtidas. Isto também é típico do tipo de pesquisa desenvolvida, exploratória e de base qualitativa, permitindo o uso de inferências nas conclusões obtidas.

Nesta pesquisa não se aprofundou a análise das razões que levam uma organização a adotar estratégias de distribuição de projetos e do seu processo de desenvolvimento de software, nem o processo de desenvolvimento de software em si. Em relação ao modelo de referência proposto, este não contempla atividades visando a subdivisão de projetos e alocação dos subprojetos (ou de um mesmo projeto) em diversas unidades distribuídas. Por último, o modelo de estágios de capacidade de projetos distribuídos apresentado trata-se de uma proposta inicial.

7.3 PESQUISAS FUTURAS

Identifica-se grande potencial de crescimento nesta linha de pesquisa, onde os pontos fortes envolvem uma parceria estável entre a academia e a indústria, criando condições de experimentação e aprendizagem únicas, decorrentes de uma sinergia positiva entre os parceiros.

Como pesquisas futuras, sugere-se:

- Continuidade do projeto de acordo com o desenho de pesquisa apresentado no capítulo 3, desenvolvendo uma ferramenta de gestão de conhecimento para o modelo proposto (etapa 2 do desenho de pesquisa) e a validação do modelo por meio de estudos empíricos (etapa 3);

- Refinamento do processo de alocação de projetos, incorporando dois procedimentos importantes, a saber:
    - De acordo com características específicas dos projetos, sugerir a melhor maneira de quebrá-lo em partes menores para distribuir entre as unidades;
    - De acordo com características e capacidade de cada unidade, verificar a possibilidade de desenvolver um projeto em *n* unidades distribuídas, utilizando *follow-the-sun* ou não.
- Aprofundar o estudo dos níveis de capacidade (maturidade) do modelo proposto, detalhando a descrição de cada estágio (aspectos táticos e operacionais).

7.4 REFLEXÃO FINAL

O desenvolvimento de software sempre se apresentou de forma complexa. Existem uma série de problemas e desafios inerentes ao processo [PRI 02a]. Assim como o processo de desenvolvimento de software tem se tornado cada mais complexo, a distribuição das equipes no tempo e no espaço tem tornado os projetos distribuídos cada vez mais comuns. Não é necessário falar a um gerente experiente das vantagens de se gerenciar projetos centralizados ao invés de distribuídos. Projetos centralizados permitem muitas vezes uma gerência através de observação, tendo algumas desvantagens tais como a comunicação informal. Por outro lado, a distribuição possui suas vantagens. Entre elas, pode-se citar a existência de grupos propensos à inovação e um maior formalismo em todas as tarefas e ações.

O DDS, ao acrescentar fatores como dispersão geográfica, dispersão temporal e diferenças culturais, acentuou alguns dos desafios existentes e acrescentou novos desafios ao processo de desenvolvimento. Entre estes desafios pode-se citar questões estratégicas, questões culturais, gestão do conhecimento e gerência de riscos como de grande importância. Por isso, o trabalho em ambientes de DDS é mais problemático do que em ambientes centralizados. O valor da interação social não deve ser subestimado. A construção de confiança entre as equipes distribuídas deve ser facilitada [CAR 99], [KIE 03], [MAR 01]. Além disso, os riscos técnicos e tecnológicos estarão sempre presentes, e estudos relacionados com os fatores técnicos têm sido amplamente divulgados [ALT 98], [DAM 02a], [HER 99], [KAR 98]. Por isso, o trabalho na prevenção de dificuldades e problemas decorrentes tanto dos fatores técnicos como dos não-técnicos deve ser sempre valorizado.

REFERÊNCIAS BIBLIOGRÁFICAS

"Para adquirir conhecimento, é preciso estudar; para adquirir sabedoria, é preciso observar". Anônimo

[ALB 00] ALBERTIN, Alberto Luiz. **Comércio Eletrônico: modelos, aspectos e contribuições de sua aplicação**. São Paulo: editora Atlas, 2000. 296 p.

[ALT 98] ALTMANN, Josef; WEINREICH, Rainer. An Environment for Cooperative Software Development Realization and Implications. In: HICSS, 1998, Havaí. **Proceedings...** EUA, p. 27-37, 1998.

[AUD 01] Audy, Jorge Luis N. **Modelo de Planejamento Estratégico de Sistemas de Informação: Contribuições do processo decisório e da aprendizagem organizacional**. 195 f. 2001. Tese (Doutorado), PPGA – UFRGS, Porto Alegre, Brasil, 2001.

[BAT 01] BATTIN, Robert; CROCKER, Don; KREIDLER, Joe; SUBRAMANIAN, K. Leveraging resources in Global Software Development. **IEEE Software**, California, v. 16, n. 2, p. 70-77, Mar./Abr. 2001.

[BEN 87] BENBASAT, Izak; GOLDSTEIN, D; MEAD, M. The case reserch strategy in studies of information system. **MIS Quarterly**, EUA, v. 11, n.3, p. 369-386, Set. 1987.

[BER 97] BERNSTEIN, Peter L. **Desafio aos Deuses – A Fascinante História do Risco**. Brasil: Campus, 1997. 389 p.

[BER 00] BERANEK, Peggy. The Impacts of Relational and Trust Development Training on Virtual Teams: An Exploratory Investigation. In: HICSS, 2000, Havaí. **Proceedings**... EUA, p. 1-10, 2000.

[BIA 03] BIANCHI, Alessandro; CAIVANO, Danilo; LANUBILE, Filippo; VISAGGIO, Giuseppe. Defect Detection in a Distributed Software Maintenance Projects. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 48-52, 2003.

[BIU 02] BIUK-AGHAI, Robert. Customizable Software Engineering Environments for Flexible Distributed Software Teams. In: APSEC, 1998, Taipei. **Proceedings...** Taiwan, p. 228-235, 1998.

[BOA 97] BOAR, Bernard. **Strategic thinking for information technology**. EUA, Nova York: John Wiley and Sons, 1997. 288 p.

[CAR 99] CARMEL, Erran. **Global Software Teams – Collaborating Across Borders and Time Zones**. EUA: Prentice Hall, 1999. 269 p.

[CAR 01a] CARMEL, Erran; AGARWAL, Ritu. Tatical Approaches for Alleviating Distance in Global Software Development. **IEEE Software**, California, v. 16, n. 2, p. 22-29, Mar. /Abr. 2001.

[CAR 01b] CARVALHO, Ariadne M. B. R., CHIOSSI, Thelma. C. S. **Introdução à Engenharia de Software**. São Paulo: Editora da Unicamp, 2001, 148 p.

[CAR 02] CARMEL, Erran; AGARWAL, Ritu. The Maturation of Offshore Outsourcing of Information Technology Work. **MIS Quarterly Executive**, EUA, v. 1, n.2, p. 65-77, Jun. 2002.

[CMM 03] CMMI – Capability Maturity Model Integration. Disponível em http://www.sei.cmu.edu/cmmi/. Acesso em 13 set. 2003.

[COC 02] COCKBURN. Alistair. **Agile Software Development – The Agile Software Development Series**. EUA: Addison-Wesley, 2002. 256 p.

[COR 01] CORTES, Mario. L., et al. **Modelos de Qualidade de Software**., São Paulo: Editora da Unicamp, 2001 148 p.

[DAM 02a] DAMIAN, Daniela. The study of requirements engineering in global software development: as challenging as important. In: International Workshop on Global Software Development at ICSE, 2002, Florida. **Proceedings...** EUA, p. 8-11, 2002.

[DAM 02b] DAMIAN, Daniela; ZOWGHI, Didar. An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In: HICSS, 2002, Havaí. **Proceedings...** EUA, p. 1-10, 2002.

[DAM 02c] DAMIAN, Daniela (editor). International Workshop on Global Software Development at ICSE, 2002, Flórida. **Proceedings...** EUA: 2002, 58 p.

[DAM 03a] DAMIAN, Daniela (editor). International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA: 2003, 81 p.

[DAM 03b] DAMIAN, Daniela; CHISAN, James; ALLEN, Polly; CORRIE, Brian. Awareness meets requirements management: awareness needs in global software development. In: International Workshop on Global Software Development at ICSE, 2003, Florida. **Proceedings...** EUA, p. 6-10, 2003.

[DEL 03] DELMONTE, Anthony J.; McCARTHY, Richard V. Offshore Software Development: Is the Benefit Worth the Risk? In: AMCIS, 2003, Florida. **Proceedings...** EUA, p. 1607-1613, 2003.

[DES 02] DESOUZA, Kevin; EVARISTO, Roberto. Global Knowledge Management Strategies. **European Management Journal**, Inglaterra, v. 21, n. 1, p. 62-67. Fev. 2003.

[DOU 97] DOUBLAIT, Stephane; Standard Reuse Practices: Many Myths vs. a Reality. **StandardView**, EUA, v. 5, n. 2, p. 84-91, Jun. 1997.

[ESP 02] ESPINOSA, J. Alberto; CUMMINGS, Jonathon N.; WILSON, Jeanne M. Research on Teams with Multiple Boundaries. In**:** HICSS, 2002, Havaí. **Proceedings**… EUA, p. 253-262, 2002.

[ESP 03] ESPINOSA, J. Alberto; Carmel, Erran. Modeling coordination costs due to time separation in Global Software Teams. In**:** International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 64-69, 2003.

[EVA 99] EVARISTO, Roberto; FENEMA, Paul. A Typology of Project Management: Emergence and Evolution of New Forms. **International Journal of Project Management**, Inglaterra, v. 17, n. 5, p. 275-281, 1999.

[EVA 00] EVARISTO, Roberto; SCUDDER, Richard. Geographically distributed project teams: a dimensional analysis. In: HICSS, 2000, Havaí. **Proceedings...** EUA, p. 1-15, 2000.

[EVA 03] EVARISTO, Roberto. The Management of Distributed Projects Across Cultures. **Journal of Global Information Management (JGIM)**, Nova Zelândia, v. 11, n. 4, p. 58-70, 2003.

[FAV 01] FAVELA, Jesús; PEÑA-MORA, Feniosky. An experience in collaborative software engineering education. **IEEE Software**, California, v. 18, n. 2, p. 47-53, Mar. /Abr. 2001.

[FRA 97] FRASER, Martin; VAISHNAVI, Vijay. A Formal Specifications Maturity Model. **Communications of the ACM**, EUA, v. 40, n. 12, p. 95-103, 1997.

[GIL 95] GIL, A. **Métodos e Técnicas de pesquisa social**. São Paulo: Atlas, 1995, 207 p.

[GLA 98] GLASS, Robert L. **Software Runaways – Lessons Learned from Massive Software Project Failures**. EUA: Prentice Hall PTR, 1998. 224p.

[GRI 99] GRINTER, Rebecca; HERBSLEB, Jamers; PERRY, DEWAYNE. The Geography of Coordination: Dealing with Distance in R&D Work. In: Group'99 Conference, 1999, NY. **Proceedings...** ACM, EUA, p. 306-315, 1999.

[HER 99] HERBSLEB, James. D; GRINTER, Rebecca. Splitting the organization and integrating the code: Conway's Law revisited. In: ICSE, 1999, Carolina do Norte. **Proceedings...** EUA, p. 85-95, 1999.

[HER 00] HERBSLEB, James; MOCKUS, Audris; FINHOLT, Thomas; GRINTER, Rebecca. Distance, Dependencies and Delay in a Global Collaboration. In: CSCW'2000, Philadelphia. **Proceedings...** EUA, p. 319-328, 2000.

[HER 01a] HERBSLEB, James; MOITRA, Deependra. Global Software Development. **IEEE Software**, California, v. 16, n. 2, p. 16-20, Mar. /Abr. 2001.

[HER 01b] HERBSLEB, James; MOCKUS, Audris; FINHOLT, T.; GRINTER, Rebecca. An empirical study of global software development: distance and speed. In: ICSE, 2001, Toronto. **Proceedings...** Canada, p. 81-90, 2001.

[HER 02] HERBSLEB, James; BOYER, David; HANDEL, Mark; FINHOLT, Thomas. Introducing Instant Messaging and Chat in the Workplace. In: CHI, 2002, Minneapolis. **Proceedings...** Minnesota, EUA, p. 171-178, 2002.

[HER 03] HERBSLEB, James; MOCKUS, Audris. An empirical study of speed and communication in globally distributed software development. **IEEE Transactions on Software Engineering**. EUA, v.29, n.6, p.481-494, 2003.

[HOP 97] HOPPEN, Norberto. Avaliação de artigos de pesquisa em sistemas de informação: proposta de um guia. In XXI Congresso da ANPAD, Rio de Janeiro. **Anais... Brasil**, 1997, 27 p.

[HOU 03] Houaiss. **Dicionário Houaiss**, Disponível em: http://www.houaiss.com.br. São Paulo. Acesso em 2 set. 2003. CD-ROM.

[IEE 93] IEEE Standards Collection: Software Engineering. **IEEE Standard 610.12 – 1990**, IEEE, 1993.

[IFP 02] International Function Point Users Group. **IT Measurement – Pratical Advice from the Experts**. EUA: Addison Wesley. 2002. 759 p.

[INS 03] Inspiration Software, Inc. Disponível em http://www.inspiration.com. Acesso em 19 set. 2003.

[JAR 98] JARVENPAA, Sirkka; KNOLL, Kathleen; LEIDNER, Dorothy. Is Anybody Out There? Antecedents of Trust in Global Virtual Teams. **Journal of Management Information Systems**, EUA, v. 14, n. 4, p. 29-64, 1998.

[KAR 98] KAROLAK, Dale Walter. **Global Software Development – Managing Virtual Teams and Environments**. Los Alamitos, EUA: IEEE Computer Society, 1998. 159 p.

[KAT 02] KATZY, Bernard; EVARISTO, Roberto, ZIGURS, Ilze. Knowledge Management in Virtual Projects: A Research Agenda. In: HICSS, 2000, EUA. **Proceedings...** EUA, v. 1, p. 1018-1026, 2002.

[KHA 03] KHAN, Naureen; CURRIE, Wendy L. Developing a Model for Offshore Outsourcing. In: AMCIS 2003, Florida. **Proceedings...** EUA, p. 996-1003, 2003.

[KIE 03] KIEL, Lori. Experiences in Distributed Development: A Case Study. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 44-47, 2003.

[KOB 03] KOBLYNSKI, Rafael; CREIGHTON, OLIVER; Dutoit, ALLEN; BRUEGGE, Bernd. Building awareness in global software engineering: Using issues as context. In**:** International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 29-33, 2003.

[KRA 99] KRAMER, Roderick. **Trust and distrust in organizations: Emerging Perspectives, Enduring Questions.** Stanford University. Annual Reviews. 1999. 30p. Disponível em: http://www.AnnualReviews.org, 1999.

[KRI 80] KRIPPENDORFF, Klaus. **Content analysis: an introduction to its methodology**. Sage, 1980.

[KRU 00] KRUCHTEN, Philippe. **The Rational Unified Process – An Introduction**. EUA: Addison-Wesley, 2000. 298 p.

[LAN 02] LANUBILE, Filippo; MALLARDO, Teresa. Preliminary Evaluation of Tool-based Support for Distributed Inspection. In: International Workshop on Global Software Development at ICSE, 2002, Florida. **Proceedings...** EUA, p. 32-35, 2002.

[LAN 03] LANUBILE, Filippo. A P2P Toolset for Distributed Requirements Elicitation. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 11-14, 2003.

[LAY 00] LAYZELL, Paul; BRERETON, Pearl; FRENCH, Andrew. Supporting Collaboration in Distributed Software Engineering Teams. In: Seventh Asia-Pacific Software Engineering Conference (APSEC.00), 2000. **Proceedings...** Singapura, p. 28-45, 2000.

[LEE 89] LEE, As. A scientific methodology for MIS case studies. **MIS Quarterly**, EUA, v. 13, n.1, p. 33-50, Mar.1989.

[LIN 02] LINDVALL, Mikael; RUS, Ioana. Knowledge Management in Software Engineering. **IEEE Software**, California, v.19, n.3, Mai/Jun. 2002, 13 p.

[MAI 99] MAIDANTCHIK, Carmen L. L. **Gerência de Processos de Software para Equipes Geograficamente Dispersas**. Tese de Doutorado, COPPE–UFRJ, Rio de Janeiro, Brasil, 1999, 214 p.

[MAR 91] MARTIN, James. **Engenharia da Informação**. Rio de Janeiro: Campus, 1991.

[MAR 01] MARQUARDT, Michael J; HORVATH, Lisa. **Global Teams: how top multinationals span boundaries and cultures with high-speed teamwork**. Davies-Black Publishing. Palo Alto, EUA. 2001. 246p.

[MAT 01] MATTSSON, Mira; FORSSANDER, Stefam; OLSSON, Ulf Corrective Maintenance Maturity Model (CM): Maintainer's Education and Training. In: ICSE, 2001, EUA. **Proceedings...** EUA, p. 610-619, 2001.

[McC 96] McCONNEL, Steve. **Rapid Development: Taming Wild Software Schedules**. EUA, Redmond: Microsoft Press, 1996. 660 p.

[NAW 01] NAWROCKI, Jerzy; WALTER, Bartosz; WOJCIECHOWSKI, Adam. Toward Maturity Model for Extreme Programming. In: EUROMICRO Conference, EUA. **Proceedings...** EUA, p. 1-7, 2001.

[NON 94] NONAKA, Ikujiro. A Dynamic Theory of Organizational Knowledge Creation. **Organization Science**, EUA, v. 5, n. 1, p. 14-37, Fev. 1994.

[NOT 00] NOTARI, Daniel L. **Uma enciclopédia para ambientes distribuídos de desenvolvimento de software**. Dissertação (Mestrado em Ciência da Computação) – PPGC, UFRGS, Porto Alegre, 2000.

[OBE 00] OBERG, Roger, PROBASCO, Leslee, and ERICSSON, Maria, Applying Requirements Management with Use Cases. **Rational Software White Paper**, EUA, Cupertino, CA, p. 3-5, 2000.

[OPP 02] OPPENHEIMER, Heather. Project Management Issues in Globally Distributed Development. In: International Workshop on Global Software Development at ICSE, 2002, Florida. **Proceedings...** EUA, p. 47-50, 2002.

[PAA 03] PAASIVARA, Maria. Communication needs, practices and supporting structures in global inter-organizational software development projects. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, p. 59-63, 2003.

[PAU 93] PAULK, Mark; CURTIS Bill; CHRISSIS, Mary; WEBER, Charles. **Capability Maturity Model for Software, Version 1.1.** Software Engineering Institute. Technical Report. 1993. 82p.

[PET 01] PETERS, J. F.; PEDRYCZ, W. **Engenharia de Software, Teoria e Prática**. Rio de Janeiro: Editora Campus, Brasil, 2001. 601 p.

[PID 98] PIDD, Michael. **Modelagem Empresarial**. Porto Alegre: Bookman, 1998.

[PMI 00] Project Management Institute. **A guide to the project management body of knowledge (PMBOK guide).** Project Management Institute. Pensylvania, EUA, 2000. 216 p.

[PRE 01] PRESSMAN, Roger S. **Software Engineering: a practitioner's approach**. EUA: McGraw Hill, 2001. 860 p.

[PRI 02a] PRIKLADNICKI, Rafael. **Problemas, Desafios e Abordagens do Processo de Desenvolvimento de Software**. 2002. Trabalho Individual I, FACIN – PPGCC, PUCRS, Porto Alegre, Jun. 2002. 69 p.

[PRI 02b] PRIKLADNICKI, Rafael. **Desenvolvimento Distribuído de Software e Processos de Desenvolvimento de Software**. 2002. Trabalho Individual II, FACIN – PPGCC, PUCRS, Porto Alegre, Ago. 2002. 66 p.

[PRI 02c] PRIKLADNICKI, Rafael; PERES, Fernando; AUDY, Jorge Luis N.; MÓRA, Michael C.; PERDIGOTO, Antonio. Requirements specification model in a software development process inside a physically distributed environment. In: ICEIS, 2002, Ciudad Real. **Proceedings...** Espanha, p. 830-834, 2002.

[PRI 02d] PRIKLADNICKI, Rafael; AUDY, Jorge Luis N. Towards a Model of Software Development Process for a Physically Distributed Environment. In: CACIC 2002, Buenos Aires. **Proceedings...** Argentina, p. 1-12, Abr. 2002.

[PRI 03a] PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Distributed Software Development: Toward an understanding of the relationship between project team, users and customers. In: ICEIS, 2003, Angers. **Proceedings...** França, p. 417-423, Abr. 2003.

[PRI 03b] PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Requirements Management in Global Software Development: Preliminary Findings from a Case Study in a SW-CMM context. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, Mai. p. 53-58, 2003.

[PRI 03c] PRIKLADNICKI, Rafael; AUDY, Jorge Luis N. **MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software**. 2003, Seminário de Andamento, FACIN – PPGCC, PUCRS, Porto Alegre Ago. 2003. 12 p.

[PYY 03] PYYSIÄINEN, Jarkko. Building Trust in Global Inter-Organization Software Development Projects: Problems and Practices. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...**

EUA, p. 70-74, 2003.

[RAP 01] REPPENING, Alexander, et al. Using components for rapid distributed software development**. IEEE Software**, California, v. 18, n. 2, p. 38-45, Mar. /Abr. 2001.

[REP 98] REPONEN, Tapio. The Role of Learning in Information System Planning and Implementation. In: GALLIERS, H. e BAETS, R. IT and Organizational Transformation, 1998, Chichester. **Proceedings...** England, 1998.

[ROY 98] ROYCE, Winston. **Software Project Management – a Unified Framework**. EUA: Addison-Wesley, 1998. 448 p.

[SAB 99] SABHERWAL, Rajiv. The role of trust in outsourced IS development projects. **Communications of ACM**. EUA, v. 42, n. 2, p. 80-86, 1999.

[SAR 02] SARMA, Anita; HOEK, André. Palantír: Increasing Awareness in Distributed Software Development. In: International Workshop on Global Software Development at ICSE, Florida. **Proceedings...** EUA, p. 51-55, 2002.

[SCH 99] SCHULMEYER, G. Gordon; McMANUS, James. I. **Handbook of *Software* Quality Assurance**. EUA: Prentice Hall PTR, 1999. 712 p.

[SCH 00] SCHWALBE, Kathy. **Information Technology Project Management**. Cambridge, England: Course Technology, 2000. 561 p.

[SOM 03] SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Addison Wesley, 2003. 592 p.

[SPR 99] SPRAGUE, Ralph H.; McNURLIN, Barbara. C. **Information Systems Management in Practice**. Canadá: Prentice Hall, 1999. 528 p.

[STE 00] STEIN, Wolfgang. There's no business like e-business. **Scheer Magazine**. Saarbrücken – Alemanha, v. 9, p. 1-6, Maio 2000.

[STR 89] STRAUB, Detmar. Validating instruments in MIS research. **MIS Quarterly**, EUA, v. 13, n. 2, p. 147-169, jun. 1989.

[SWE 01] IEEE Computer Society. **Guide to the software engineering body of knowledge (SWEBOK).** IEEE. California, EUA, 2001. 228 p.

[TUR 95] TURBAN, E.; WETHERBE, J.; McLEAN, E. **Information Technology for Management: Improving Quality and Productivity**. Nova York: John Wiley and Sons, 1995. 848 p.

[VOG 01] VOGEL, Douglas; DAVISON, Robert; SHROFF, Ronnir; QURESHI, Sajda. Sociocultural Learning in Globally Distributed Teams. In: Informing Science Conference, 2001, Cracóvia. **Proceedings...** Polônia, p. 527-536, 2001.

[YIN 01] YIN, Robert. **Estudo de Caso: planejamento e** métodos. São Paulo: Bookman, 2001, 205 p.

[ZOW 02] ZOWGHI, Didar. Does Global Software Development need a diferent requirements engineering process? In: International Workshop on Global Software Development at ICSE, Florida. **Proceedings...** EUA, p. 56-58, 2002.

APÊNDICE 1 – PROTOCOLO PARA ESTUDO DE CASO

Protocolo para Estudo de Caso: Desenvolvimento Distribuído de Software (DDS)

Objetivo

Identificar as variáveis relacionadas ao desenvolvimento distribuído de software, presentes nas organizações onde serão desenvolvidos os estudos de caso (*Organização 1* e *Organização 2*). Será abordado o processo de desenvolvimento de software nas suas dimensões técnica e não-técnica.

Característica-chave do método de estudo de caso

Este é um roteiro para uma entrevista semi-estruturada com questões abertas que se caracteriza como uma pesquisa do tipo transeccional. O objetivo é identificar dificuldades, soluções (práticas) e fatores críticos de sucesso (FCS) do DDS.

Organização desse Protocolo

O protocolo será organizado com o segue:

1. Procedimentos

A. Reuniões para levantamento das questões e estruturação do guia para a entrevista	
Participantes:	Rafael Prikladnicki
Data:	24 de Março de 2003
Local:	AGT PUCRS – Agência de Gestão Tecnológica da PUCRS

B. Reuniões para revisão do guia para a entrevista	
Participantes:	Jorge Audy (Especialista, Pesquisador Sênior)
Data:	29 de Março de 2003
Local:	AGT PUCRS – Agência de Gestão Tecnológica da PUCRS

C. Autorização das empresas participantes	
Participantes:	Contato 1 (*Organização 1*) Contato 2 (*Organização* 2)
Data:	7 de Abril de 2003
Local:	Local: Sede das organizações

D. Validação de Face e Conteúdo	
Participantes:	Ricardo Bastos. Doutor em Ciência da Computação – UFRGS 1999. (Especialista, Pesquisador Sênior)
Data:	15 de Abril de 2003
Local:	Local: PPGCC (Programa de Pós Graduação em Ciência da Computação – FACIN – PUCRS)

E. Pré-teste		
Participantes:	Sabrina Marczak (SQA, mestranda – convênio Dell/ PUCRS) Maurício Cristal (Gerente de Projeto Dell Inc)	
Data:	14 de Abril de 2003	Horário: 14:00 -16:00
Local:	Local: Dell Brazil GDC	

F. Aplicação das entrevistas – Questões Organizacionais	
Participantes:	***Organização 1***: Gerente de Desenvolvimento ***Organização 2*:** Gerente de Projeto

G. Aplicação das entrevistas – Questões Referentes aos Projetos	
Participantes: - integrantes de dois projetos selecionados em cada organização; - dois Gerentes de Desenvolvimento; - um integrante do SEPG.	***Organização 1***: - Projeto 1: - Gerente de Desenvolvimento: **ok** - Gerente de Projeto: **ok** - Líder Técnico: **ok** - Desenvolvedor: **ok** - Projeto 2: - Gerente de Desenvolvimento: **ok** - Gerente de Projeto: **ok** - Líder Técnico: **ok** - Desenvolvedor: **ok** - Integrante do SEPG: **ok** - Coordenador de SQA: **ok** ***Organização 2***: - Projeto 1: - Gerente de Desenvolvimento: **ok** - Gerente de Projeto: **não foi possível** - Líder Técnico: **não foi possível** - Desenvolvedor: **ok** - Projeto 2: - Gerente de Desenvolvimento: **ok** - Gerente de Projeto: **ok** - Líder Técnico: **ok** - Desenvolvedor: **ok** - Integrante do SEPG: **ok** - Coordenador de SQA: **ok**

H. Local e Data de aplicação das entrevistas	
Local:	***Organização 1***: Porto Alegre, RS ***Organização 2***: São Paulo, SP
Data:	***Organização 1***: 5 e 6 de Maio de 2003 (2ª e 3ª) ***Organização 2***: 28 e 29 de Abril de 2003 (2ª e 3ª)

2. Escolha das pessoas entrevistadas

Respondentes:

- Gerente de desenvolvimento
- Gerente de projeto
- Líder Técnico (*Technical Leader*)
- Desenvolvedor
- Integrante do *SEPG*
- Coordenador de SQA

3. Outros recursos utilizados

Sistema computacional

- Sistema *Sphinx* (tabulação de dados)

Recursos financeiros (Convênio Dell/PUCRS)

- Deslocamento em São Paulo (Estudo de Caso na *Organização 1*)
- Uma (1) diária de hotel em São Paulo

Recursos materiais

- Um gravador para gravar as entrevistas
- Uma sala de reunião com reserva para dois (2) dias
- Papel e Caneta

4. Modelo do estudo e Dimensões da Pesquisa

O esquema a seguir representa graficamente os principais aspectos enfocados no desenvolvimento deste trabalho.

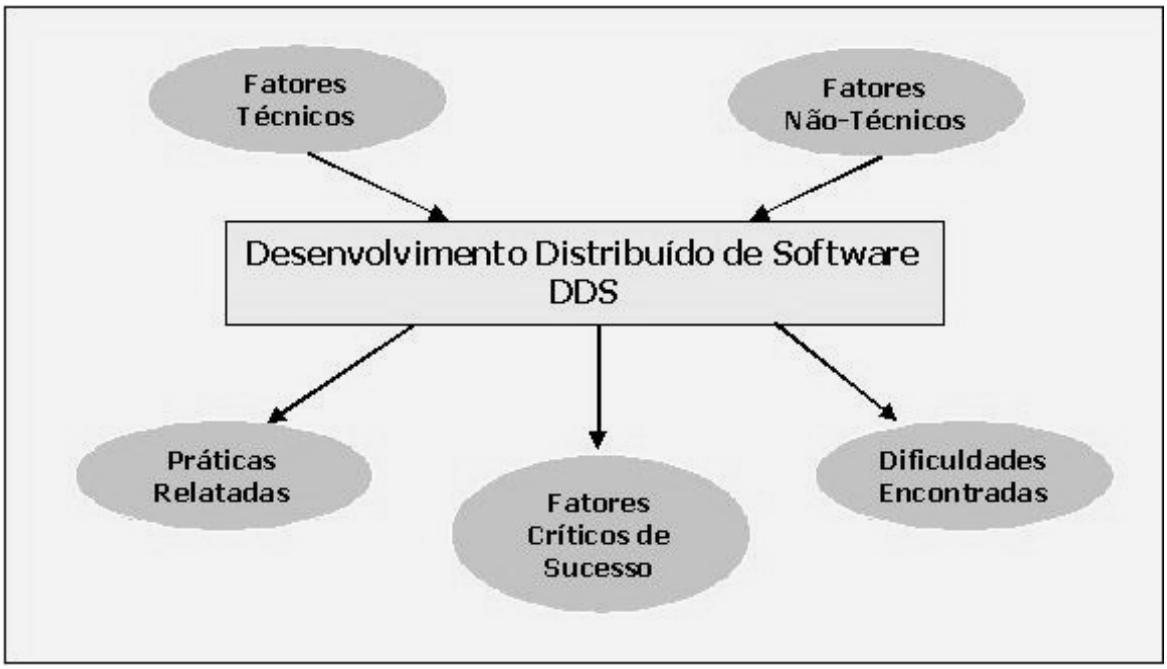

Figura 7.1 – Modelo do estudo de caso realizado.

5. Análise de dados

Foi realizada uma análise de dados baseada na técnica de análise de conteúdos, conforme proposto por [KRI 80], com o uso do software *Sphinx* para a análise de dados qualitativos. Esta entrevista insere-se em uma pesquisa de base qualitativa, exploratória, sendo o estudo de caso o principal método de pesquisa, aplicado conforme proposto por [YIN 01].

6. Dimensões e questões do guia para entrevista semi-estruturada

	QUESTÕES
DADOS DEMOGRÁFICOS	Nome: Idade:
	Escolaridade
	Curso (nível mais alto)
	Universidade
	Ano de Conclusão
	Tempo de experiência profissional na área de Informática
	Empresa: Tempo de empresa:
	Vínculo (empregatício)
	Função

	QUESTÕES
REF. ESTRATÉGICOS	**Questões Descritivas**
	1. Qual é o negócio da empresa?
	2. Qual é a missão da empresa?
	3. Quais são as políticas da empresa?
	4. Quais são os principais clientes da empresa?
	5. Por que a empresa adota uma estratégia de desenvolvimento de software *offshore*?

	QUESTÕES
PROC. DE NEGÓCIOS	**Questões Descritivas**
	6. Quais os principais processos de negócios existentes na empresa?Qual é a estrutura organizacional da empresa?

	QUESTÕES
ESTTRUT. ORGAN.	**Questões Descritivas**
	7. Qual é a estrutura organizacional da empresa?
	8. Qual é a estrutura da empresa na área de Desenvolvimento de Software?

QUESTÕES	
PROC. DES. DE SW	**Questões Descritivas**
	9. Existe algum processo formal de desenvolvimento de software utilizado pela empresa? - metodologia - ciclo de vida - atividades
	10. Quais são os tipos de projetos suportados pelo processo de desenvolvimento de software?
	11. Qual é a estrutura da equipe de um projeto de software (papéis, quantidade)?

QUESTÕES	
DIFICULDADES ENCONTRADAS	**Questões Descritivas**
	12. No seu projeto quais foram as principais dificuldades enfrentadas com relação ao desenvolvimento distribuído de software?

Dimensão Técnica	Dimensão Não-Técnica
- Processo, Ferramentas	- Diferenças Culturais
- Nível de Dispersão, Tipos de atores	- Protocolos de Comunicação
- Fuso horário, Sincronização	- Compartilhamento de Contexto
- Complexidade, Tipo de Projeto	- Confiança
- Processo de Desenvolvimento	- Etiquetas
- Procedimentos e Padrões	

	13. Na sua experiência em outros projetos similares na empresa, quais foram as principais dificuldades enfrentadas com relação ao desenvolvimento distribuído de software?
	14. Em quais atividades do processo de desenvolvimento de software cada dificuldade foi encontrada?

QUESTÕES	
SOLUÇÕES	**Questões Descritivas**
	15. Para cada dificuldade identificada, qual foi a solução adotada?
	16. Como esta solução foi identificada?
	17. Na sua análise, qual foi o impacto da solução adotada, tanto para a empresa quanto para o projeto?

QUESTÕES	
FCS	**Questões Descritivas**
	18. Tendo por base este projeto e a sua experiência em outros projetos similares na empresa, na sua opinião, quais são os fatores críticos de sucesso principais para o desenvolvimento de projetos de software em ambientes de desenvolvimento distribuído de software?

QUESTÕES	
OBSERVAÇÕES	**Questões Descritivas**
	19. Tendo por base a sua experiência em desenvolvimento de software, como você compara o desenvolvimento de software centralizado com o distribuído?
	20. Você tem algum comentário adicional que gostaria de acrescentar, considerando tudo o que foi dito nesta entrevista (Dificuldades no desenvolvimento distribuído de software, práticas adotadas para lidar com estas dificuldades e fatores críticos de sucesso)?

7. Roteiro das Entrevistas

Dimensão 1: Organizacional

Dados Demográficos

Identificação Pessoal

Nome: _____ **Idade:** _____ **Anos**

Escolaridade **(informe somente o maior grau)**

☐ **1º Grau** ☐ **2º Grau** ☐ **Superior Incompleto** ☐ **Superior Completo**

☐ **Especialização** ☐ **Mestrado / MBA Incompleto** ☐ **Mestrado / MBA Completo**

Identifique as informações solicitadas abaixo conforme a escolaridade assinalada acima

Curso: _____

Universidade: _____

Ano de conclusão: _____

Tempo de experiência profissional na área de Informática**:** _____ **anos**

Empresa: ☐ ***Organização 1*** ☐ ***Organização 2***

Tempo de empresa

☐ **Até 6 meses** ☐ **De 7 a 12 meses** ☐ **De 1 a 2 anos** ☐ **De 2 a 5 anos**

Vínculo

☐ **Contratado** ☐ **Terceirizado** ☐ **Estagiário**

Função**:** _____

(Conforme a terminologia utilizada na empresa)

Referenciais Estratégicos

1. Qual é o negócio da empresa?
2. Qual é a missão da empresa?
3. Quais são as políticas da empresa?
4. Quais são os principais clientes da empresa?
5. Por que a empresa adota uma estratégia de desenvolvimento de software *offshore*?

Processo de Negócio

6. Quais os principais processos de negócios existentes na empresa?

Estrutura Organizacional

7. Qual é a estrutura organizacional da empresa?
8. Qual é a estrutura da empresa na área de Desenvolvimento de Software?

Processo de Desenvolvimento de Software (PDS)

9. Existe algum processo formal de desenvolvimento de software utilizado pela empresa? (metodologia, ciclo de vida, atividades)
10. Quais são os tipos de projetos suportados pelo processo de desenvolvimento de software?
11. Qual é a estrutura da equipe de um projeto de software (papéis, quantidade)?

Dimensão 2: Projetos

Dados Demográficos

Identificação Pessoal

Nome: _____ **Idade:** _____ **anos**

Escolaridade **(informe somente o maior grau)**

☐ **1º Grau** ☐ **2º Grau** ☐ **Superior Incompleto** ☐ **Superior Completo**

☐ **Especialização** ☐ **Mestrado / MBA Incompleto** ☐ **Mestrado / MBA Completo**

Identifique as informações solicitadas abaixo conforme a escolaridade assinalada acima

Curso: _____

Universidade: _____

Ano de conclusão: _____

Tempo de experiência profissional na área de Informática**:** _____ **anos**

Empresa: ☐ ***Organização 1*** ☐ ***Organização 2***

Tempo de empresa

☐ **Até 6 meses** ☐ **De 7 a 12 meses** ☐ **De 1 a 2 anos** ☐ **De 2 a 5 anos**

Vínculo

☐ **Contratado** ☐ **Terceirizado** ☐ **Estagiário**

Função**:** _____

(Conforme a terminologia utilizada na empresa)

Dificuldades Encontradas

12. No seu projeto, quais foram as principais dificuldades enfrentadas com relação ao DDS?
13. Na sua experiência em outros projetos similares na empresa, quais foram as principais dificuldades enfrentadas com relação ao desenvolvimento distribuído de software?
14. Em quais atividades do PDS cada dificuldade foi encontrada?

Soluções Encontradas

15. Para cada dificuldade identificada, qual foi a solução adotada?
16. Como esta solução foi identificada?
17. Na sua análise, qual foi o impacto da solução adotada?

Fatores Críticos de Sucesso

18. Tendo por base este projeto e a sua experiência em outros projetos similares na empresa, na sua opinião, quais são os fatores críticos de sucesso principais para o desenvolvimento de projetos de software em ambientes de desenvolvimento distribuído de software?

Observações

19. Tendo por base a sua experiência em desenvolvimento de software, como você compara o desenvolvimento de software centralizado com o distribuído?
20. Você tem algum comentário adicional que gostaria de acrescentar, considerando tudo o que foi dito nesta entrevista (Dificuldades no desenvolvimento distribuído de software, práticas adotadas para lidar com estas dificuldades e fatores críticos de sucesso)?

APÊNDICE 2 – ARTIGOS PUBLICADOS

PRIKLADNICKI, Rafael; PERES, Fernando; AUDY, Jorge Luis N.; MÓRA, Michael C.; PERDIGOTO, Antonio. Requirements specification model in a software development process inside a physically distributed environment. In: ICEIS, Ciudad Real. **Proceedings...** Espanha, Abr. 2002. 5 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N. Towards a Model of Software Development Process for a Physically Distributed Environment. In: CACIC, 2002, Buenos Aires. **Proceedings...** Argentina, Abr. 2002. 12 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Distributed Software Development: Toward an understanding of the relationship between project team, users and customers. In: ICEIS, Angers. **Proceedings...** França, Abr. 2003. 7 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Requirements Management in Global Software Development: Preliminary Findings from a Case Study in a SW-CMM context. In: International Workshop on Global Software Development at ICSE, 2003, Oregon. **Proceedings...** EUA, Mai. 2003. 6 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N. Um Modelo de Referência para Desenvolvimento Distribuído de Software. In: WTES, 2003, Manaus. **Proceedings...** Brasil, Out. 2003. 12 p. (palestra convidada)

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Requirements Engineering in Global Software Development: Preliminary Findings from a Case Study in a SW-CMM context. In: SIMPROS, 2003, Recife. **Proceedings...** Brasil, Nov. 2003. 6 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. An Empirical Study on Global Software Development: Offshore Insourcing of IT Projects. In: International Workshop on Global Software Development at ICSE, 2004, Edimburgo, Escócia. **Proceedings...** UK, Mai. 2004. 6 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. A Reference Model for Global Software Development. In: PRO-VE at IFIP World Computer Congress, 2004, Toulosse. **Proceedings...** França, Ago. 2004. 8 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N.; EVARISTO, Roberto. Global Software Development in Practice: Lessons Learned. **Journal of Software Process Improvement and Practice, Special Issue on GSD**, 2004. (a ser publicado).

PRIKLADNICKI, Rafael; AUDY, Jorge Luis N. MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software. In: SBES 2004, Brasília. **Proceedings**, Brasil, Out. 2004. 16 p.

MuNDDoS na Internet:

http://www.inf.pucrs.br/~rafael/munddos

http://planeta.terra.com.br/informatica/rafapri/munddos

Este trabalho foi financiado pelo convênio Dell/PUCRS, através da Lei de Informática Brasileira (Lei no. 8.248/91).

Este trabalho foi digitado conforme o
Modelo de Dissertação da Biblioteca Central Irmão José Otão da PUCRS, segundo a NBR 14724, atualizado em 06 de agosto de 2003.