Uma proposta para processo de requisitos em ambientes de desenvolvimento distribuído de software

Leandro T. Lopes[*], Azriel Majdenbaum[**] e Jorge Luiz N. Audy[+]

[*] Mestrando, Programa de Pós-Graduação em Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Brasil.
lteixeira@inf.pucrs.br

**azriel_majdenbaum@dell.com
[+] Orientador, Professor Titular da Faculdade de Informática, Programa de Pós-Graduação em Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul
audy@inf.pucrs.br

Resumo. A crescente tendência de distribuição do processo de desenvolvimento tem afetado diversas áreas da engenharia de software. A engenharia de requisitos, por ser um processo que exige grande volume de comunicação e compreensão, sofre influência direta de fatores como linguagem, contexto e cultura. Existe necessidade de um novo processo de requisitos que enderece essas dificuldades, de forma a reduzir o impacto causado pela distribuição das equipes de desenvolvimento. Nesse sentido, este estudo propõe, de forma preliminar, um processo de requisitos para ambientes de desenvolvimento distribuído de software.

1 Introdução

Nos últimos anos, o software se tornou um componente vital de negócios. O sucesso de uma organização cada vez mais depende da utilização do software como um diferencial competitivo. Ao mesmo tempo, a economia tem convertido os mercados nacionais em mercados globais, criando novas formas de competição e colaboração [1].

Entretanto, o mercado global de software vem passando por diversas crises. Por um lado, um grande número de falhas em projetos. De outro, uma demanda crescente, atingida pela escassez de recursos capacitados. Nesse contexto, muitas organizações perceberam o desenvolvimento distribuído de software como uma alternativa, experimentando o desenvolvimento em locais remotos. A necessidade de padronização e coordenação dos esforços no desenvolvimento distribuído de software levou as empresas a buscar modelos de qualidade como o SW-CMM (*Capability Maturity Model*).

Contudo, problemas com a engenharia de requisitos tem sido reconhecida como a maior razão para falhas em projetos onde o produto final não atende as expectativas do cliente [2][3]. Estes problemas tendem a se aprofundar em ambientes de

desenvolvimento distribuído, onde questões como distância física, diferença de fusos horários e cultura influenciam diretamente as atividades de engenharia de software.

Segundo [4] o desenvolvimento distribuído de software necessita de um novo processo de engenharia de requisitos que enderece as novas imposições do ambiente.

Este trabalho visa dar um primeiro passo nesse sentido, apresentando uma proposta preliminar de processo de requisitos para ambientes de desenvolvimento distribuído de software.

A seção 2 apresenta a base teórica de engenharia de requisitos e desenvolvimento distribuído. A seção 3 apresenta o método de pesquisa utilizado. Na seção 4 é apresentado o ambiente de desenvolvimento objeto deste estudo. Em seqüência, é apresentado o processo proposto na seção 5. Na seção 6 é realizada uma análise crítica do processo em relação à base teórica. Em conclusão, são apresentadas as considerações finais.

2 Base Teórica

O presente estudo utiliza como base teórica diversos conceitos referentes à engenharia de requisitos, como fases e artefatos envolvidos e inserção no modelo SW-CMM. Além disso, foram utilizados diversos fundamentos de desenvolvimento distribuído de software. Esta base teórica está descrita a seguir, nesta seção.

2.1 Engenharia de Requisitos

Uma completa compreensão dos requisitos de software é fundamental para um desenvolvimento de software bem sucedido. Não importa quão bem projetado ou quão bem codificado seja, um programa mal analisado e especificado não atenderá as expectativas do usuário e trará problemas para o desenvolvedor.

Embora ainda não exista uma solução à prova de falhas para essa questão, um processo sólido de engenharia de requisitos é a melhor solução existente para assegurar que especificamos um software de acordo com as necessidades do cliente, e que satisfará suas expectativas [5].

Entretanto, a maior dificuldade na construção de um software é decidir precisamente o que construir. Nenhuma outra parte do trabalho conceitual é mais difícil quanto estabelecer detalhadamente os requisitos técnicos. Nenhuma outra parte é mais difícil de ser corrigida tardiamente [6].

Nesse sentido, a engenharia de requisitos é a ciência e disciplina preocupada com a análise e documentação dos requisitos, incluindo análise das necessidades e análise e especificação dos requisitos, fornecendo mecanismos apropriados para facilitar as diversas atividades relacionadas [3].

2.1.1 Fases da Engenharia de Requisitos

Um processo é um conjunto de atividades que transformam entradas em saídas. É difícil definir um plano seqüencial de atividades que descrevam adequadamente o processo de engenharia de requisitos [4]. Para fins deste estudo, são consideradas as

fases da engenharia de requisitos segundo [5] e [7], conforme descritas abaixo. Estas fases não ocorrem necessariamente de forma linear, comumente realizando iterações, ou sendo executadas em paralelo.

Elicitação dos Requisitos: nesta fase são identificadas as expectativas e necessidades dos *stakeholders* com relação ao software a ser desenvolvido;

Análise e Negociação dos Requisitos: depois de obtidos os requisitos iniciais, estes são utilizados como base para análise dos requisitos. A análise distribui os requisitos em categorias, explora as relações entre eles, e classifica a importância de cada um dos requisitos de acordo com as necessidades dos *stakeholders*;

Especificação dos Requisitos: nesta etapa é produzida a especificação do software. Esta especificação pode ser um documento, um modelo gráfico, um modelo matemático formal, um conjunto de cenários de uso, um protótipo, ou uma combinação destes;

Modelagem do sistema: a construção de um modelo de sistema visa fornecer um modelo significativo do que será construído, explorando características como entradas e saídas;

Validação dos requisitos: esta etapa examina a especificação do software, de forma a assegurar que todos os requisitos foram definidos sem ambigüidades, inconsistências ou omissões, e que todos os erros foram detectados e corrigidos;

Gerência dos requisitos: a gerência dos requisitos é o conjunto de atividades que ajudam a equipe a identificar, controlar e rastrear os requisitos e suas modificações em todos os momentos durante o processo.

2.1.2 Artefatos relacionados com a engenharia de requisitos

De acordo com o processo de software utilizado por uma organização, o conjunto de artefatos envolvidos com a engenharia de requisitos pode variar. O RUP [8] (*Rational Unified Process*), por exemplo, utiliza como artefatos em seu *workflow* de gerenciamento de requisitos o documento visão (*Vision*), o modelo de casos de uso (*Use Case Model*) e a especificação suplementar (*Supplementary Specification*). Estes dois últimos documentos formam a especificação dos requisitos do software (*Software Requirements Specification* – SRS). De forma similar, outros processos de software tendem a seguir o padrão da IEEE [9], e utilizar um documento de especificação de requisitos (SRS).

O SRS é o principal artefato produzido pela engenharia de requisitos, descrevendo de forma clara e precisa cada um dos requisitos essenciais do software e suas interfaces externas. Cada requisito é definido, no SRS, de forma que sua implementação possa ser verificada de forma objetiva [3].

Um SRS, em geral, serve como base para acordos formais entre clientes e fornecedores, com relação ao que o software deve realizar [9]. Com base no SRS são realizadas estimativas de esforço e custos, bem como posteriores alterações.

Considerada a importância deste documento, é fundamental que os requisitos sejam expressos de forma clara, não ambígua, correta, completa, priorizada, consistente, verificável, modificável e rastreável.

2.1.3 Engenharia de requisitos no SW-CMM

O SW-CMM (*Capability Maturity Model*) foi desenvolvido pela comunidade de software, com liderança do *Software Engineering Institute* (SEI), sediado na universidade de Carnegie-Mellon, Estados Unidos. A primeira versão foi apresentada em 1992, descrevendo os princípios e práticas que apóiam a maturidade do processo de software. O SW-CMM visa auxiliar as organizações de software a ampliarem a maturidade de seus processos de forma evolucionária, de processos caóticos a processos maduros, disciplinados.

Quanto à sua estrutura, o CMM está baseado em uma série de práticas, organizadas em cinco níveis crescentes de maturidade. Cada nível define o grau de maturidade dos processos de uma organização e é composto por várias áreas-chave de processo. Cada área-chave permite alcançar um conjunto de metas ou objetivos. A satisfação desses objetivos é que permite dizer se a organização atingiu um determinado nível de maturidade. Os níveis de maturidade do SW-CMM são apresentados na Fig. 1:

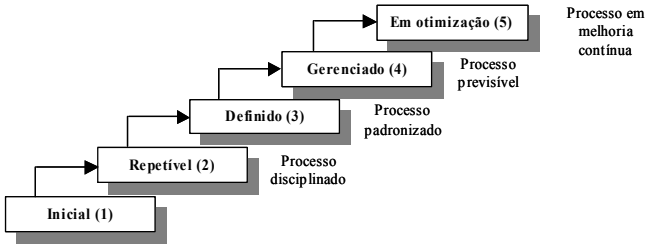

Fig. 1. Níveis de maturidade do SW-CMM [10]

A gerência de requisitos é a primeira área-chave que deve ser endereçada para uma organização atingir o nível 2 do SW-CMM. O objetivo da KPA de gerência de requisitos é, em síntese, estabelecer um entendimento comum entre o cliente e a equipe de software em relação aos requisitos. Este entendimento comum serve como base para acordos entre o cliente e a equipe de desenvolvimento, definindo e controlando as atividades a serem realizadas.

Nos diversos níveis do SW-CMM, recomendações especificam que todas as atividades, planos, agendas e produtos de software sejam desenvolvidos e modificados conforme necessário para serem consistentes com os requisitos alocados para o software. Para apoiar esse processo, os requisitos devem ser documentados e revisados por gerentes de software e grupos afetados, incluindo representantes dos clientes e usuários.

No modelo SW-CMM os requisitos se apresentam como entidades centrais do processo de desenvolvimento [2]. Com isso, o processo de requisitos aparece em todos os níveis do modelo e em diversas áreas-chave.

O processo de análise é necessário para assegurar que os requisitos façam sentido, e estejam definidos de forma clara, completa, não ambígua, consistente entre si, e testável.

Além disso, no modelo, a rastreabilidade dos requisitos é altamente valorizada. Os requisitos, projeto, código, e casos de teste são rastreados para origem de onde derivaram e para os produtos das atividades subseqüentes de engenharia. Dessa

forma, o CMM fornece um conjunto de atividades que devem ser realizadas para melhorar a qualidade de software e a produtividade. O processo de requisitos faz parte integral deste modelo, exercendo um papel central na atividade de desenvolvimento.

2.2 Desenvolvimento Distribuído de Software (DDS)

A crescente globalização do ambiente de negócios tem afetado diretamente o mercado de desenvolvimento de software [1]. Em busca de vantagens competitivas como baixos custos, produtividade ou qualidade na área de desenvolvimento de sistemas, diversas organizações optaram por distribuir o processo de desenvolvimento de software dentro de seu país, ou em outros países, como a Índia e o Brasil. Estas regiões oferecem, muitas vezes, incentivos fiscais ou possuem grande concentração de massa crítica em determinadas áreas.

Embora o desenvolvimento de software tenha evoluído consideravelmente nas últimas décadas, ainda são enfrentadas diversas dificuldades. Muitos projetos de software têm sido entregues atrasados ou com custos superiores ao esperado. Freqüentemente, o software não realiza as funcionalidades desejadas por seus usuários. Ao optar por instanciar um ambiente de desenvolvimento distribuído, além de todas as dificuldades inerentes ao processo de desenvolvimento co-localizado, uma organização começa a enfrentar diversos desafios de adaptação, diferenças culturais, planejamento do trabalho, treinamento, entre outros.

De forma a embasar a contextualização deste estudo, são apresentados na seqüência formas de caracterizar os níveis de DDS.

2.2.1 Níveis de DDS

Como base para caracterização dos níveis de DDS, é utilizada neste artigo a classificação de [11], onde são considerados dois fatores: atores envolvidos e distância física. Estes fatores estão descritos brevemente na seqüência.

Os atores envolvidos no processo de desenvolvimento de software podem ser divididos em equipe de projeto, clientes e usuários. A equipe de projeto (P) representa todos os envolvidos no desenvolvimento de um determinado projeto, podendo também ser formada por um conjunto de subequipes. 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) é a pessoa física ou jurídica que solicitou e contratou o desenvolvimento de um determinado projeto. O usuário (U) representa os responsáveis por utilizar o produto gerado. Às vezes, clientes e usuários podem ser as mesmas pessoas, representando os dois papéis simultaneamente.

Com relação à distância física, podemos classificar em: mesma localização física, distância nacional, distância continental e distância global. Mesma localização física é a situação onde a empresa possui toda a sua equipe instalada em um mesmo local (sala, prédio), considerando todos os atores envolvidos nesta situação. Reuniões ocorrem sem dificuldades e a equipe pode interagir estando fisicamente presente. Distância nacional caracteriza-se por ter a equipe localizada dentro de um mesmo país. Nesta situação, a equipe pode se reunir em curtos intervalos de tempo. Distância

continental é a situação onde a equipe esta localizada em países diferentes dentro de um mesmo continente. Distância global caracteriza-se por ter a equipe localizada em países e continentes distintos.

Com base nestes fatores, [11] definiu dois critérios para classificação da dispersão: distância física inter-atores e distância física intra-atores.

Distância física inter-atores é o critério que, considerando os três atores existentes (equipe de projeto, clientes e usuários), define a distância física existente para eles. Caso existam diversos tipos de distribuições, é considerada a maior distância física existente entre atores.

Distância física intra-atores é o critério que define a distância física existente dentro de cada equipe de atores (por exemplo, usuários). Caso existam diversos tipos de distribuições, é considerada a maior distância física dentro da equipe.

3 Método de pesquisa

Este artigo é um resultado parcial de um projeto de pesquisa na área de DDS que busca focar em engenharia de requisitos. O estudo fundamenta-se em uma forte base teórica em engenharia de software nas áreas de DDS, SW-CMM e engenharia de requisitos.

A parte empírica da pesquisa está sendo realizada em duas grandes empresas da área de tecnologia da informação que atuam em ambientes de desenvolvimento distribuído de software em nível global.

4 Ambiente de DDS

Ambientes de desenvolvimento distribuído podem se apresentar de diversas formas. Considerando as distâncias inter e intra-atores, podemos instanciar muitas distribuições distintas dos elementos envolvidos.

O principal foco deste estudo é o desenvolvimento *offshore outsourcing*. O *offshore outsourcing* é a prática de contratar uma organização externa, localizada em outro país, para o desenvolvimento do software. Esta opção de desenvolvimento tem se tornado bastante popular nos últimos anos.

No desenvolvimento *offshore* o ambiente de DDS se caracteriza pela equipe de projeto estar distante continental ou globalmente de seus clientes e usuários. Por exemplo, em uma empresa global, os clientes podem estar na sede da empresa, nos Estados Unidos. Os usuários podem estar distribuídos nos Estados Unidos e Canadá, e a equipe de desenvolvimento pode estar distribuída entre Estados Unidos, Índia, Brasil e China (Fig. 2).

Fig. 2. Exemplo de ambiente *offshore*.

Esta mesma situação pode ser representada utilizando o formato de [11], conforme Fig. 3. Os círculos representam cada um dos atores envolvidos no processo. A distância interna aos círculos representa a distância física intra-atores. A distância apresentada entre os círculos representa a distância física inter-atores.

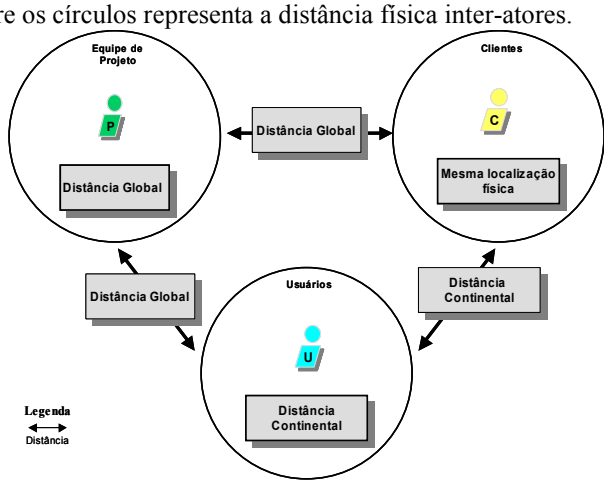

Fig. 3. Ambiente *offshore* representado segundo [11]

4.1 Requisitos em ambientes de DDS

Em relação à engenharia de requisitos os papéis comumente envolvidos no desenvolvimento de software *offshore*, na equipe de projeto, são de analista de negócios e analista de aplicações. Estes papéis podem se confundir no contexto das organizações. Para fins deste estudo serão considerados da seguinte forma:

Analista de negócios é o responsável pela condução da elicitação, análise, negociação, especificação e validação dos requisitos junto aos *stakeholders*. Mantém a rastreabilidade dos requisitos em relação à origem.

Analista de aplicação é o responsável pela modelagem do software com base na especificação e manutenção da rastreabilidade dos requisitos durante as etapas de modelagem, codificação e teste.

No desenvolvimento co-localizado de software estes papéis são, muitas vezes, realizados pela mesma pessoa ou grupo. No desenvolvimento *offshore* há uma tendência a serem realizados por pessoas ou grupos distintos por estarem, comumente, separados fisicamente.

Assim, podemos identificar duas equipes com tarefas distintas no processo de engenharia de requisitos em ambientes *offshore*. Para fins deste estudo, a equipe ligada ao analista de negócios é chamada de equipe de especificação e a equipe ligada ao analista de aplicação é chamada de equipe de desenvolvimento.

Sobrepondo estas considerações ao exemplo de desenvolvimento *offshore* dado anteriormente, obtemos a Fig. 4. Esta distribuição representa a equipe de especificação situada nos Estados Unidos e as equipes de desenvolvimento localizadas no Brasil, Índia e China.

É importante reconhecer que além da distribuição entre as equipes de especificação e projeto, pode existir distribuição entre a equipe de especificação e os *stakeholders*, afetando as etapas da engenharia de requisitos que envolvem estes grupos, como destacado por [12] e [13].

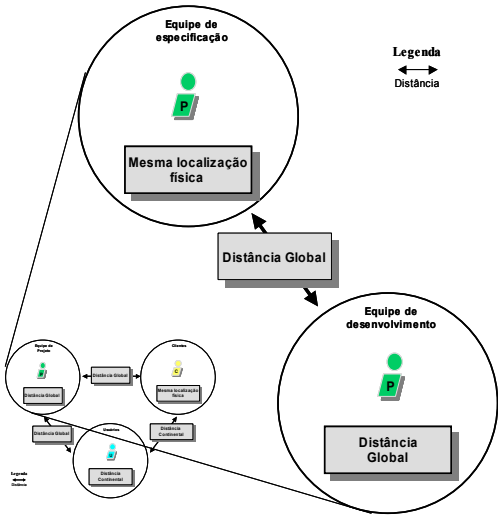

Fig. 4. Distribuição das equipes de engenharia de requisitos

4.2 Dificuldades na engenharia de requisitos em DDS

O DDS promove um aprofundamento das dificuldades inerentes ao processo de desenvolvimento de software, ou ainda, o surgimento de novas dificuldades, como diferenças culturais e linguagem.

Como a distância envolvida pode compreender países distantes, comumente, a linguagem e a cultura são diferentes. Com isso, os problemas causados por ambigüidade e falta de clareza nos requisitos são intensificados. A compreensão dos requisitos ao serem lidos em uma língua diferente da nativa é mais limitada, levando a interpretações incorretas. Diferenças culturais como atitude em relação à hierarquia, riscos e valores culturais podem ampliar a possibilidade de conflitos.

Sem o conhecimento presencial do ambiente onde o software será inserido, a compreensão das razões e expectativas do software por parte da equipe de desenvolvimento é reduzida [12].

A comunicação também apresenta novos desafios. Com a perda de contato face-a-face entre a equipe de desenvolvimento e os usuários, existe uma maior dificuldade de esclarecimento em caso de dúvidas. Além disso, com a utilização de meios de baixo contexto, como correio eletrônico, o contato informal entre os membros dos diversos grupos é limitado, reduzindo a confiança entre eles.

Em casos de grupos separados por diversos fusos horários, em geral, ocorre uma maior demora na tomada de decisões. Uma simples troca de mensagens por correio eletrônico para esclarecimento de um requisito pode levar dias se os horários de trabalho não coincidirem.

Reuniões por vídeo ou teleconferência podem não ser tão efetivas. Grupos com reduzida confiança tendem a evitar comprometimentos. Algumas culturas valorizam mais questões como pontualidade e agenda que outras, ampliando as possibilidades de conflitos.

A tabela 1 apresenta uma síntese das principais dificuldades identificadas na engenharia de requisitos em ambientes de DDS. Muitos destes desafios ocorrem na engenharia de requisitos em ambientes co-localizados, entretanto a distância tende a exacerba-los, ampliando o impacto causado.

Tabela 1. Principais dificuldades encontradas na engenharia de requisitos em DDS

Identificação dos *stakeholders*
Ambigüidade e falta de clareza
Compreensão do ambiente onde os requisitos se aplicam
Comunicação
Confiança
Demora
Baixa efetividade nas reuniões por vídeo ou teleconferência
Conflitos
Diferenças culturais

Considerando as diversas dificuldades apresentadas pelo ambiente em estudo, na seção seguinte é descrita uma proposta de processo, de forma a abrandar o impacto causado por algumas destas dificuldades.

5 Processo Proposto

O processo proposto visa reduzir as dificuldades causadas pela distribuição no processo de requisitos, sob o ambiente explanado na seção anterior, fazendo com que a equipe de desenvolvimento realize uma tarefa de entendimento e adaptação do SRS. Este processo é composto de cinco etapas, conforme a Fig. 5.

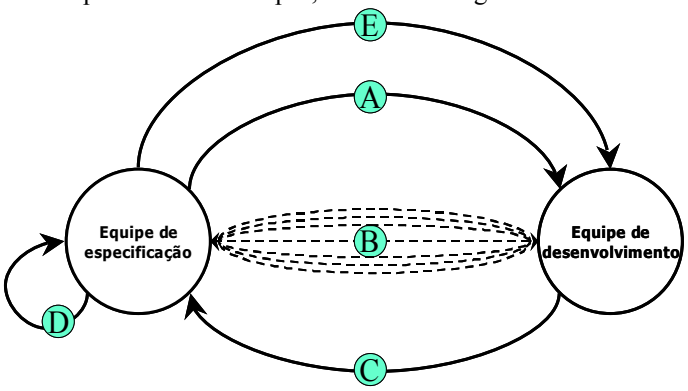

Fig. 5. Processo Proposto

As etapas do processo são as seguintes:

A. Envio do SRS para a equipe de desenvolvimento.

Enquanto realiza as etapas de elicitação, análise, negociação e especificação junto aos *stakeholders*, a equipe de especificação cria a versão inicial do SRS. Quando esta versão é finalizada, o SRS é enviado para equipe de desenvolvimento.

Em um processo co-localizado de engenharia de requisitos poderia ser dado início, neste momento, à modelagem e codificação do software. Entretanto, considerando o contexto em estudo, as dificuldades causadas pela distribuição do desenvolvimento tendem a reduzir o entendimento do SRS pela equipe de desenvolvimento. Diferenças culturais e de idioma podem provocar erros de interpretação, produzindo um software que não atende as necessidades dos clientes. Com isso, são necessárias atividades de adaptação.

B. Análise do SRS pela equipe de desenvolvimento.

Dependendo da etapa de engajamento da equipe de desenvolvimento, esta poderá ser a primeira vez que esta equipe tem contato com o documento de requisitos do software.

De forma geral, tenta-se engajar a equipe de desenvolvimento logo no início da idealização do software, permitindo um melhor entendimento dos motivos que geraram a demanda. O contato com documentos como Visão/Escopo permite uma melhor contextualização e preparação da equipe de desenvolvimento. Entretanto, o documento de requisitos passa por um processo evolutivo difícil de ser acompanhado

à distância. Novas demandas surgem e muitos contatos com os *stakeholders* são realizados, tornando complexo o acompanhamento pela equipe de desenvolvimento.

Após receber o SRS da equipe de especificação, em primeiro lugar é necessário que a equipe de desenvolvimento busque o entendimento da especificação e seu contexto. Durante esta etapa o SRS é adaptado para reduzir potenciais fontes de problemas. Ambigüidade e falta de clareza são ainda mais prováveis quando diferenças culturais e de linguagem são envolvidas. Em casos de grandes dificuldades, o SRS pode ser completamente reescrito.

O conhecimento obtido durante o processo de criação do SRS tende a ser mais profundo que o conhecimento contido no documento. A adaptação ou reescrita serve como atividade auxiliar para evitar que este conhecimento seja perdido. Dúvidas surgem e devem ser esclarecidas com a equipe de especificação, caracterizando grande volume de comunicação entre as equipes.

A comunicação entre as equipes, além de esclarecimentos, visa obter consenso quanto à especificação em construção, simplificando as etapas posteriores do processo. Algumas vezes, esclarecimentos solicitados à equipe de especificação podem demandar novos contatos com usuários e clientes, caso abordem um ponto de dúvida ainda não percebido. Com isso, a qualidade da especificação tende a aumentar.

A reescrita ou adaptação do SRS pode servir como forma de padronização dos documentos de entrada no processo de desenvolvimento. Comumente, uma equipe de desenvolvimento pode atender a diversas equipes de especificação, que tendem a utilizar estruturas e nível de detalhe diferentes, por exemplo. A equipe de desenvolvimento pode utilizar padrões de documento, glossário, formato de casos de uso e requisitos, por exemplo, evitando que o SRS possua formatos distintos em cada projeto. Esta necessidade cresce quando se considera a aplicação de métricas e estimativas.

C. Envio do SRS para aprovação da equipe de especificação.

Uma vez que o SRS foi adaptado ou reescrito, o novo documento deve ser aprovado. Ao finalizar a etapa de análise do SRS pela equipe de desenvolvimento, o documento deve ser enviado para verificação pela equipe de especificação.

D. Validação do SRS pela equipe de especificação.

A equipe de especificação deve verificar o SRS, assegurando que após as mudanças ele ainda represente as necessidades e características solicitadas pelos *stakeholders*.

Uma dificuldade aparente nesta etapa seria o grande esforço de análise necessária pela equipe de especificação para validar o documento e assegurar que ele reflete as características necessárias. Entretanto, as diversas interações realizadas durante a segunda etapa permitem que a equipe de especificação acompanhe o processo de adaptação do SRS, reduzindo o esforço necessário para validação.

E. Versão final do SRS é definida.

Após aprovação pela equipe de especificação, a versão final do SRS é definida como o SRS aprovado. Esta versão é então utilizada pela equipe de desenvolvimento como base para a modelagem, codificação e testes do software.

6 Análise Crítica

Conforme apresentado por [4] é necessário um novo processo de engenharia de requisitos que enderece as dificuldades apresentadas pelo desenvolvimento distribuído de software. O processo proposto visa reduzir a influência causada por questões como diferenças de cultura e linguagem no processo de requisitos, além de ampliar a compreensão dos requisitos por parte da equipe de desenvolvimento.

De acordo com [9] o SRS não deve ser ambíguo tanto para quem o criou, quanto para quem o utiliza. Nesse sentido, este modelo de processo, embora simples, visa reduzir ambigüidades e aumentar a compreensão dos requisitos pela equipe de desenvolvimento distribuída.

O modelo de qualidade de software SW-CMM recomenda, em uma de suas áreas-chave, que os requisitos sejam acordados por todos os grupos envolvidos. Para que este acordo seja obtido é indispensável a completa compreensão dos requisitos pelas partes envolvidas. Em ambientes de DDS isso envolve, muitas vezes, sobrepor barreiras de linguagem e cultura. O processo proposto auxilia na obtenção de um consenso entre os envolvidos, de forma consciente.

O conhecimento obtido na criação do documento de requisitos é, em geral, de maior extensão que o descrito neste documento. A elicitação, análise, negociação e especificação oferecem um grande volume de informações e experiência para os envolvidos. Com este conhecimento não pode ser descrito totalmente através do SRS, a equipe de desenvolvimento distribuído deixa de contar com informações importantes para contextualização do software a ser desenvolvido. Este processo, ao sugerir a adaptação ou reescrita do SRS na segunda etapa visa, além de clarificar a documentação existente, a obtenção do conhecimento inerente ao processo de especificação.

Quando a equipe de desenvolvimento atende a diversas equipes de especificação, é natural que os documentos de requisitos advindos das várias equipes tenham formatos e padronizações diferentes. Com isso, questões como o nível de detalhe dos requisitos, formato do documento, glossário e formato de casos de uso e requisitos, por exemplo, afetam diretamente o processo de desenvolvimento e as métricas envolvidas. A adaptação a padrões e formatos sugerida na segunda fase é importante para adequar o SRS ao processo de software utilizado pela equipe de desenvolvimento. Dessa forma, tenta-se padronizar a documentação de entrada, reduzindo o efeito causado pela diversidade de padrões das equipes de especificação.

A troca de informações entre as equipes permite que ocorra um aprendizado contínuo. Ao adaptar ou reescrever o SRS e enviar para equipe de especificação, o formato de padronização é aprendido e evoluído, fazendo com que novas especificações sejam feitas mais próximas da necessidade da equipe de

desenvolvimento. Além disso, com um melhor entendimento do SRS a confiança entre as equipes, fator fundamental em DDS, tende a crescer.

7 Considerações Finais

Este trabalho apresenta uma evolução em direção a um processo de requisitos para ambientes de desenvolvimento distribuído de software. A equipe de pesquisa está iniciando estudos empíricos envolvendo duas grandes empresas multinacionais da área de tecnologia da informação que atuam em ambientes de desenvolvimento distribuído de software, em nível global. Já foi desenvolvido um teste piloto neste sentido, que propiciou à equipe de pesquisa avançar no sentido de propor o modelo de processo explicado neste artigo.

A engenharia de software vem realizando excelentes progressos nos últimos anos. Modelos de processo como o RUP e modelos de maturidade como o SW-CMM têm sido adotados largamente e com sucesso no meio empresarial. Entretanto, novos desafios estão surgindo, exigindo diferentes abordagens para processos existentes. O desenvolvimento distribuído de software é um destes desafios.

A engenharia de requisitos sempre foi reconhecida como uma área crítica no desenvolvimento de software. O seu estudo em ambientes de desenvolvimento distribuído de software oferece grandes oportunidades de pesquisa por ser uma área nova, com crescimento acelerado e poucos trabalhos desenvolvidos no Brasil. Novas técnicas, processos e ferramentas são claramente necessários, aumentando a relevância dos estudos. Este estudo contribui no sentido de propor um processo de requisitos para ambientes de desenvolvimento distribuído de software.

8 Referências Bibliográficas

[1] Herbsleb, James; Moitra, Deependra. Global Software Development. IEEE Software. 2001. 5p.

[2] Leffingwell, Dean; Widrig, Don. Managing Software Requirements – A Unified Approach. Addison-Wesley. 2000. 492p

[3] Thayer, Richard; Dorfman, Merlin. System and Software Requirements Engineering, Second Edition. IEEE Computer Society Press. 2000. 528p

[4] Zowghi, Didar. Does Global Software Development need a diferent requirements engineering process? Proceedings of International Workshop on Global Software Development 2002. 2002. 3p.

[5] Pressman, Roger S. Software Engineering: a practitioner's approach. 5th ed. McGraw Hill. 2001. 860p

[6] Berry, Daniel; Lawrence, Brian. Guest Editors' Introduction - Requirements Engineering. IEEE Software. 1998. pp 26-29.

[7] Sommerville, Ian; Sawyer, Peter. Requirements Engineering – a good practice guide. Wiley, 1997.

[8] Kruchten, Philippe. The Rational Unified Process: An Introduction. 2a. Edição. Addison-Wesley. 2001. 298p

[9] IEEE, Institute. IEEE Std 830-1998. Recommended Practice for Software Requirements Specifications. Institute of Electrical and Eletronic Engineers. Inc. 1998

[10] Paulk, Mark; Curtis, Bill; Chrissis, Mary; Weber, Charles. "Capability Maturity Model, Version 1.1", IEEE Software, Vol. 10. No. 4. 1993.

[11] Prikladnicki, Rafael. ArcoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software. Seminário de Andamento, Mestrado em Ciência da Computação. PUCRS. 2002.

[12] Damian, Daniela; Zowghi, Didar. An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. Proceedings of the 36th Hawaii International Conference on Systems Sciences (HICSS'03). IEEE. 2002.10p

[13] Lloyd, James W.; Rosson, Mary B.; Arthur, James D. Effectiveness of Elicitation Techniques in Distributed Requirements Engineering. Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02). IEEE. 2002. 8p.