UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCIANA DA CONCEIÇÃO FERREIRA MENDES

DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE:
UMA SOLUÇÃO ESTRATÉGICA

São Luís
2015

LUCIANA DA CONCEIÇÃO FERREIRA MENDES

DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE:
UMA SOLUÇÃO ESTRATÉGICA

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profª. Ma. Maria Auxiliadora Freire

São Luís
2015

Mendes, Luciana da Conceição Ferreira.

Desenvolvimento Distribuído de Software: Uma Solução Estratégica/ Luciana da Conceição Ferreira Mendes. – São Luís, 2015.

76 f.

Orientadora: Profa. Msc. Maria Auxiliadora Freire

Monografia (Graduação) – Universidade Federal do Maranhão,
Curso de Ciência da Computação, 2015.

1. Desenvolvimento distribuído de software 2. Processo de desenvolvimento de software 3. Colaboração 4. Project-Open I. Título.

CDU 004.438

LUCIANA DA CONCEIÇÃO FERREIRA MENDES

DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE: UMA SOLUÇÃO ESTRATÉGICA

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 17/04/2015

BANCA EXAMINADORA

Profª. Ma. Maria Auxiliadora Freire (Orientadora)
Mestra em Ciência de Engenharia
Universidade Federal do Maranhão

Prof. Me. Carlos Eduardo Portela Serra de Castro
Mestre em Informática
Universidade Federal do Maranhão

Prof. Dr. Samyr Beliche Valle
PhD em Ciência da Computação
Universidade Federal do Maranhão

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me abençoar nessa caminhada e permitir que conseguisse chegar até o final.

Aos meus pais, que acreditaram em mim, e sempre estiveram dispostos a me ajudar em tudo que fosse preciso.

Ao meu noivo, Luis Fernandes, que foi muito compreensivo nos momentos difíceis e incentivou o meu crescimento pessoal e profissional.

Aos meus professores, pelo conhecimento que me proporcionaram, e principalmente, á minha orientadora Maria Auxiliadora, pela valiosa ajuda nessa fase final do curso.

Aos meus colegas de curso em especial á minha amiga Débora Penha pelo companheirismo durante todo o curso.

E a todos que de alguma forma contribuíram para que a realização desse trabalho fosse possível.

“Vivemos todos sob o mesmo céu, mas nem todos temos o mesmo horizonte”.

(Konrad Adenauer)

RESUMO

Uma das áreas da tecnologia mais dinâmica é a área de desenvolvimento de sistemas, que exige de seus profissionais um contínuo processo de aprendizagem. E para que os projetos de software sejam produzidos com qualidade é necessário o uso de metodologias que conduzam o processo de maneira eficiente e produtiva.
Com a globalização, o crescimento da importância dos sistemas de informação nas organizações e os processos de terceirização, surge o desenvolvimento distribuído de software. Diante deste cenário, o presente trabalho tem como objetivo a implementação de um projeto de software usando o modelo distribuído. No desenvolvimento do projeto distribuído fez-se necessário o uso de uma ferramenta que tornasse o processo mais eficiente, a escolhida foi *Project-Open*, por se encaixar melhor nos requisitos necessários para sua realização.

Palavras-chave: Desenvolvimento distribuído de software. Processo de desenvolvimento de software. Colaboração. Project-Open.

ABSTRACT

One of the most dynamic technology areas is the system development, that requires its practitioners a continuous learning process. And for that software projects are produced with quality methodologies with leading the process of efficiently and productively is required.

With globalization, the growing importance of information systems in organizations and outsourcing processes, distributed development of software arises. Against this background, this study aims to implement a software project using the distributed model. In developing the project distributed became necessary to use a tool that would make the process more efficient, the choice was Project-Open, because it fits better on the requirements necessary for its realization.

Keywords: distributed software development. Software development process. Collaboration. Project -Open .

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CASE – *Computer Aided Software Engineering*

COBIT – *Control Objectives for Information End Relatet Technology*

DDS – Desenvolvimento Distribuído de Software

IDE – *integrated Development environment*

ITIL – *Information Technology Infrastructure Library*

RCS – *Revision Control System*

TI – Tecnologia da Informação

UFMA – Universidade Federal do Maranhão

SUMÁRIO

1 INTRODUÇÃO

Pode-se perceber nos últimos anos um grande avanço em direção á globalização dos negócios. E na área de desenvolvimento de software isso não é diferente. O software tem se tornado um componente estratégico para diversas áreas de negócio. Especialmente na área de Engenharia de Software. Mercados nacionais tem se transformado em mercados globais, criando novas formas de cooperação e competição que vão além das fronteiras dos países (Prikladnic; Audy, 2007).

Mas o deslocamento de um mercado nacional para um mercado global gera novas formas de competição e cooperação que cruzam os limites nacionais. Isto tem um profundo impacto não só na distribuição de produtos de software, mas na forma como eles são concebidos, projetados, construídos, testados e entregues aos clientes (Prikladnic; Audy, 2007).

Sendo assim as empresas tem adotado o desenvolvimento distribuído de software (DDS) com o intuito de minimizar as dificuldades encontradas nesse novo cenário de competição, dentre elas a demanda de custos, a necessidade de rapidez de resposta ao mercado, o próprio mercado e presença global, a escala e o rigor e experiência no desenvolvimento.

Contudo este novo ambiente de desenvolvimento também gerou novas dificuldades e desafios, dentre eles podemos citar desafios com relação a pessoas, processos, tecnologia, gestão e comunicação. Portanto, uma proposta de solução para esses novos desafios é não só importante, mas fundamental para que o desenvolvimento distribuído de software torne-se eficiente.

Sendo assim, este trabalho busca mostrar através do uso da ferramenta *Project-Open,* o desenvolvimento de um projeto de software de forma distribuída com o intuito de apresentar soluções para minimizar os problemas relacionados à comunicação, coordenação e controle, decorrentes da distância temporal, geográfica e sociocultural presentes no desenvolvimento distribuído de software.

1.1 Motivação

Apresentar soluções para minimizar os problemas relacionados à comunicação, coordenação e controle, decorrentes da distância temporal, geográfica e sociocultural presentes no desenvolvimento distribuído de software.

1.2 Objetivos

Este trabalho tem como objetivo principal apresentar uma solução estratégica para o suporte do Desenvolvimento Distribuído de Software (DDS).

Os objetivos específicos são listados:

a) Evidenciar a estrutura usada para desenvolvimento distribuído de software diferenciando da utilizada em ambientes co-localizados.
b) Identificar os desafios do desenvolvimento distribuídos de software com base nos estudos efetuados.
c) Apresentar soluções para minimizar os problemas relacionados a comunicação,coordenação e controle, decorrentes da distância temporal, geográfica e sociocultural presentes no desenvolvimento distribuído de software**.**

1.3 Organização do Trabalho

A apresentação detalhada dos aspectos deste trabalho está distribuída da seguinte forma:

Neste primeiro capítulo foi realizada a introdução deste trabalho.

O segundo capítulo apresentará as principais características do desenvolvimento distribuído de software com o intuito de diferenciá-lo dos modelos tradicionais.

O terceiro capítulo apresenta uma análise de algumas ferramentas de acordo com os recursos necessários para colaboração no DDS, que são: cooperação, coordenação, comunicação, suporte e gerenciamento.

O quarto capítulo apresenta um estudo de caso onde serão seguidas as fases genéricas do desenvolvimento de software, com intuito de mostrar o desenvolvimento de um projeto de software usando a ferramenta *Project-Open*.

O quinto capítulo apresenta a conclusão do trabalho, ressaltando as dificuldades e vantagens do uso do DDS no estudo de caso.

2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Os modelos de desenvolvimento de software vão desde os mais clássicos até os mais contemporâneos. O modelo a ser usado depende muito de alguns fatores, como por exemplo, custos, relação de negócio, incentivos fiscais, proximidade do cliente entre outros.

O modelo de desenvolvimento de software tradicional é realizado por pessoas próximas fisicamente, o que torna mais fácil a comunicação e troca de informações pertinentes, mas em algumas circunstancias esse processo tende a não ser o mais adequado.

Em alguns casos, demandas, rapidez de resposta ao mercado rigor e experiência no desenvolvimento pode levar uma organização a desenvolver o software em outra cidade, estado, país ou até mesmo continente. Nesses casos em que as pessoas envolvidas no processo de desenvolvimento são impedidas de trabalhar em proximidade física, é necessário fazer uso de outra forma de desenvolvimento: desenvolvimento distribuído de software (DDS) (SIQUEIRA, 2005).

Esse tipo de desenvolvimento surge em um cenário onde mercados nacionais se tornam mercados globais e a competição toma novas formas. O desenvolvimento de software vinha passando por diversas crises, tanto com relação a falhas em projetos como em relação a crescente demanda. Nessas circunstancias muitas organizações encontraram no DDS uma alternativa de desenvolver software (PRIKLADNICKI, 2003).

2.1 Fatores geradores do DDS

Anteriormente o desenvolvimento de software era realizado apenas de maneira centralizada e por pessoas com auto grau de especialização, que dispunham de grandes centros de processamentos de dados. Mas atualmente o quadro é diferente. O que se vê é um crescimento maior do desenvolvimento distribuído de software. E esse crescimento não se deve a um fator em especial, mas a um conjunto de fatores que de maneira independente ou conjunta influenciam esse tipo de desenvolvimento. Alguns fatores geradores do DDS são:

Globalização do comercio de serviços: com a globalização dos negócios e ambiente empresarial competitivo, o software se tornou um serviço indispensável. Diante dessas circunstancias cresce também o mercado de software. Muitas empresas de software investem em DDS, pois dessa maneira é possível acompanhar o mercado global e ficar próximas dos seus consumidores. (PRIKLADNIC; AUDY, 2008).

Custos mais baixos: a globalização dos negócios na área de engenharia de software torna muito custoso desenvolver software de maneira centralizada. O avanço da tecnologia na comunicação torna possível o desenvolvimento distribuído de software, pois os custos de comunicação diminuíram muito. Observa-se também um gasto menor com viagens e transferência de funcionários. Têm-se ainda as melhorias em ferramentas e métodos de desenvolvimento. Logo se torna menos custoso desenvolver software de maneira distribuída.

Proximidade ao cliente: Durante o desenvolvimento é importante à proximidade com o cliente. O levantamento de requisitos e possíveis dúvidas que surjam durante o processo só podem ser revolvidos com o cliente, o que torna indispensável ter uma equipe próxima a ele.

Falta de espaço no escritório ou escala: em algumas situações o tamanho do projeto necessita de muitas pessoas trabalhando nele. O que nesse caso inviabiliza o desenvolvimento em um mesmo local e até mesmo prejudica a gerencia do mesmo. Nesses casos o melhor é distribuir, dividido em grupos. Para que não só haja mais espaço para o trabalho no ambiente, mas também favoreça o desenvolvimento e gerencia do projeto.

Política do país: alguns países exigem que haja uma instalação da empresa no país para fechar o contrato. Mas mesmo nesses casos algumas empresas podem preferir executar algumas tarefas em suas sedes (SIQUEIRA, 2005).

Incentivos fiscais: a cobrança de menos impostos e mão de obra barata também são grandes atrativos para o DDS, pois em alguns lugares a mão de obra ou abrir uma empresa se torna mais caro. Normalmente são escolhidos países em desenvolvimento como Índia, por exemplo, que não só possui mão de obra mais barata, mas também qualificada.

Acesso à mão de obra especializada: a falta de mão de obra especializada em algumas tarefas específicas leva uma organização à busca de profissionais em

outros locais. Isso ocorre porque algumas empresas buscam profissionais que além de uma boa formação básica, tenham pós-graduação e fluência em um idioma específico, o mais indicado é o inglês.

2.2 **Dispersão geográfica**

A distância física é um dos fatores que mais caracteriza o DDS, pois em algumas circunstancias não é possível desenvolver software em um mesmo local, algumas vezes devido a custos, mão de obra ou mesmo interesse dos *stakeholders*. AUDY; PRIKLADNICKI (2008) divide a distância física ou níveis de dispersão dos *stakeholders* em quatro tipos: mesma localização, distância nacional, distância continental e distância global, mas nesse trabalho serão consideradas apenas três níveis, mesma localização, distância nacional e distância global.

2.2.1 Mesma localização

Situação em que a organização possui todos os atores no mesmo local. Nessas situações reuniões ocorrem sem dificuldades, pode haver uma maior interação por parte dos membros da equipe. Não existe diferença de fuso horário e as diferenças culturais raramente envolvem a dimensão nacional. As dificuldades são as mesmas encontradas no desenvolvimento de software centralizado**.**

2.2.2 Distância nacional

Situação caracterizada por ter os atores localizados dentro de um mesmo país, com possibilidade de reunir-se em pequenos intervalos de tempo. Dependendo do país pode haver diferenças em relação ao fuso horário, e as diferenças culturais podem ser maiores em comparação com a situação em que os atores desenvolvem em uma mesma localização.

2.2.3 Distância global

Nesta situação os atores se encontram localizados em países diferentes e em continentes diferentes. As reuniões face a face são mais raras e normalmente ocorrem no inicio do projeto.

Há dificuldades na comunicação e a diferença cultural é ainda mais notável o que pode ser uma barreira pra o trabalho. Mas a maior dificuldade é a diferença de fuso horário, pois pode impedir interações entre a equipe.

Nos casos de distância nacional, continental e global verifica-se o desenvolvimento distribuído de software, pois segundo AUDY; PRIKLADNICKI (2007) o desenvolvimento distribuído de software ocorre quando pelo menos um dos atores envolvidos no processo estiver distante fisicamente dos demais.

2.3 **Desafios do DDS**

Assim como o software tradicional, o distribuído também possui diversos obstáculos em seu desenvolvimento, alguns deles são: dispersão geográfica (distância física), dispersão temporal (diferença de fuso horário), e diferenças socioculturais (idioma, costumes, traições, normas e comportamentos) (PRIKLADNIC; AUDY, 2008).

Segundo PRIKLADNIC; AUDY (2008) essas mesmas características acrescentam outros desafios para o DDS, pois criam um grande impacto na forma como o produto é concebido, desenvolvido e testado. Sendo assim, os desafios e dificuldades de DDS são divididos em quatro grupos: Desafios relacionados à gestão, a processos, a pessoas, a tecnologia e a comunicação.

2.3.1 Desafios relacionados á gestão

Os desafios relacionados á gestão tomam maiores proporções quando se trata de software distribuído, pois as equipes não estão todas em um mesmo local (dispersão física), e nem todas falam o mesmo idioma, tem a mesma cultura e nem o mesmo fuso horário, o que torna o controle e a coordenação mais difícil por parte dos gestores do projeto.

Alguns desses problemas estão relacionados à gestão de portfólio de projetos, gerenciamento de projetos que exige uso de algumas técnicas especiais, ou seja, não usadas no desenvolvimento tradicional. Pode-se citar também o gerenciamento de riscos, pois se sabe que a gestão de risco é indispensável em qualquer projeto, e no projeto distribuído esse risco pode não ser muito visível (PRIKLADNIC; AUDY, 2008).

2.3.2 Desafios relacionados á processos

Processos são uma sequência de atividades que se realizam sem sequencia para obtenção de um objetivo comum. E para que esse objetivo seja alcançado no processo de desenvolvimento de software é fundamental que esses passos sejam realizados de maneira correta. Um dos processos mais importante é a engenharia de requisitos, ou seja, a primeira etapa na construção de um software.

A engenharia de requisitos se divide em quatro fases: elicitação analise e negociação, especificação e validação. Cabe à elicitação a tarefa de identificar os fatos que compõem os requisitos do sistema, de forma a prover o mais correto e mais completo entendimento do que é demandado daquele software. Na fase de analise e negociação são descobertos problemas de inconsistência e incompletude dos requisitos e se forem encontrados deve voltar ao cliente para resolvê-los através de um processo de negociação.

Na especificação é produzido o documento que muitas vezes serve como o contrato de serviço entre as partes. Já na fase de validação procura-se demonstrar se o documento produzido é exatamente o que foi proposto pelo cliente.

Diante de todo esse processo é possível notar desafios com relação a todas as fases. Na fase de elicitação têm-se os desafios relacionados á comunicação, que podem levar a problemas relacionados ao entendimento do projeto. Na fase de análise e negociação têm-se desafios relacionados á consistência dos requisitos, dificultada pela distância entre os atores, também desafios relacionados á compreensibilidade entre os requisitos, já que nem todos os atores participam da elicitação dos requisitos, e por ultimo desafios relacionados á validação de requisitos, pois no DDS os clientes e desenvolvedores podem não estar no mesmo local, o que pode ocasionar uma dificuldade em acordar com o projeto final.

2.3.3 Desafios relacionados á pessoas

Desafios relacionados a pessoas também são muito encontrados nesse tipo de desenvolvimento, tem-se a falta de confiança, conflitos, diferenças culturais, liderança e etc.

No desenvolvimento distribuído de software, onde as equipes estão dispersas é fundamental que haja confiança entre os participantes do projeto. Pois se não existe confiança em uma equipe as chances de falha no projeto são muito altas. É importante que existam pelo menos algumas vezes durante o projeto encontros face a face, principalmente no inicio do projeto, para que haja uma maior socialização.

Alguns conflitos em projetos é algo normal, pois nem todas as pessoas estão de acordo com alguns aspectos do mesmo, mas em DDS pode haver um agravamento de conflitos ou pode levar um tempo maior para que eles sejam solucionados, pois quando a distância entre os *steakholders* é muito grande, pode haver uma diferença de fusos horários o que muitas vezes torna impossível resolver logo aquele conflito.

Outro desafio á ser observado com relação ás pessoas é o desafio na comunicação, pois a comunicação em projetos de software é fundamental. Vai do levantamento de requisitos á manutenção do software. Sem uma boa comunicação não é possível desenvolver um software de qualidade. Mas essa comunicação em DDS pode ser afetada pela dispersão física (distância física), temporal, fusos horários e idiomas diferentes.

A distância física entre os envolvidos no projeto caracteriza os diversos níveis de dispersão. As dificuldades encontradas no nível de dispersão global tende a ser bem maior que nos demais níveis. Os níveis de dispersão podem ajudar a caracterizar melhor a distribuição física das equipes e consequentemente ajudar a identificar possíveis fontes de problemas (Gärtner, 2011). É imprescindível entender os níveis de dispersão, pois segundo PRIKLADNIC; AUDY (2008) quando a distância atinge 30 metros ou mais a frequência de comunicação tende a se tornar igual ao nível de colaboradores que estão a quilômetros de distância.

Outro fator que pode afetar a comunicação é a diferença de idioma, pois é necessário muitas vezes que as equipes envolvidas no desenvolvimento se reúnam

para tirar alguma dúvida entrar em acordo sobre algum aspecto do projeto ou mesmo consultar um documento.

Segundo SIQUEIRA (2005) as diversas formas de comunicação são divididas em dois grupos: a comunicação síncrona e a assíncrona. Na comunicação assíncrona o receptor tem um tempo para ler a mensagem e analisá-la, ou seja, a resposta não é imediata. São exemplos de comunicação assíncrona o correio eletrônico, grupos de discussão e até mesmo o correio tradicional.

A comunicação síncrona é o contrário da assíncrona, ou seja, o emissor espera uma resposta imediata, logo não há tempo para fazer analise da mensagem. Acontece nos casos de conversa por telefone, vídeo conferencia e diálogos face a face. Nesse caso é fundamental que se conheça o idioma falado, caso contrário ficará impossível à compreensão da mensagem e em consequência a comunicação.

Logo, percebe-se que tanto na comunicação síncrona quanto assíncrona é necessário que haja por parte dos interlocutores um entendimento, não só superficial da língua em questão, mas um grau que permita uma boa comunicação, e no desenvolvimento distribuído de software torna-se mais importante ainda o domínio do idioma que será usado no desenvolvimento das atividades, pois um pequeno mal entendido pode causar grandes prejuízos.

2.3.4 Desafios relacionados á tecnologia

Diante da dispersão física imposta pelo DDS, surge a necessidade de comunicação eletrônica entre as equipes envolvidas no projeto. Para esse tipo de comunicação usa-se mais comumente a teleconferência e o correio eletrônico. Mas há ainda a necessidade de algumas ferramentas que auxiliem no desenvolvimento, nesses casos são usadas ferramentas cases.

Por outro lado é fundamental ter uma boa infraestrutura Como infraestrutura entende-se não só hardware, o software, as ferramentas, as técnicas, os padrões e as instalações envolvidas, mas também a infraestrutura engloba as técnicas, os padrões e as instalações envolvidas no desenvolvimento, a operação e a manutenção de produtos de software (SIQUEIRA, 2005).

A infraestrutura tecnológica no processo de desenvolvimento de software é a base à vida de qualquer sistema. A decisão sobre a infraestrutura tem função estratégica que poderá limitar ou potencializar o desenvolvimento do software.

Muitas vezes os funcionários de uma determinada empresa trabalham em casa, ou em hotéis, o que torna difícil montar uma infraestrutura adequada para o DDS. Em outros casos há mais de uma empresa envolvida no projeto, o que gera uma heterogeneidade na infraestrutura o que pode levar a conflitos no projeto.

Logo, torna-se indispensável um planejamento não só quanto ao desenvolvimento de software, mas também com relação a Infraestrutura, já que ela reflete diretamente não só na qualidade do software, mas também no processo como um todo.

A Infraestrutura é um fator que não depende de distância física. Pois se pode ter uma Infraestrutura idêntica quando se deseja. O fator Infraestrutura está ligado ao fato de se trabalhar em alguns casos com empresas diferentes no caso de desenvolvimento distribuído.

2.4 Modelos de negócio

Com o surgimento do DDS surge também para as organizações a possibilidade do desenvolvimento de software utilizando equipes distribuídas geograficamente, pois em um projeto onde o DDS é utilizado tem-se a possibilidade de distribuição em um mesmo local, em países diferentes ou mesmo em continentes diferentes.

Dessa maneira têm-se diferentes culturas organizacionais, ou seja, modelos inventados por uma determinada organização, que tendo funcionado bem o suficiente é considerado válido e sendo assim é passado para os demais membros da organização e visto como a maneira correta de agir com relação aos determinados problemas.

Logo, a cultura organizacional é algo que varia de organização para organização, e como não é algo estático, pois é um aprendizado constante da organização, muda até mesmo, com o passar do tempo, dentro de uma organização. Ou seja, novos

valores e comportamentos são agregados e alguns costumes e comportamentos antigos são substituídos.

Sendo assim, é um fator que influencia o desenvolvimento distribuído de software, já que a cultura de uma empresa pode diferir da cultura de outra. Algumas práticas comuns em uma organização podem não ser toleradas por outras. E é possível notar essa diferença na cultura organizacional mesmo no caso de distância nacional.

Essa diferença na cultura organizacional pode afetar bastante o andamento de um projeto distribuído, pois para que um projeto seja realizado é necessário que haja acordo entre as partes interessadas, e essa diferença de cultura organizacional pode gerar conflitos, pois uma atitude pode ser percebida de maneira diferente pelas pessoas, o que algumas consideram normal, outras podem ter outra opinião. Isso pode ocorrer, pois a maioria dos sistemas distribuídos envolve mais que uma organização no projeto.

Segundo SIQUEIRA (2005), a cultura organizacional está dividida em três níveis distintos, são eles: os artefatos, os valores expostos e os pressupostos básicos. Os artefatos seriam os elementos visíveis, como por exemplo, o ambiente de trabalho, os produtos, a tecnologia, as estórias e até mesmo a linguagem comum á empresa. Os valores expostos, como o próprio nome já diz, seriam os objetivos, estratégias e filosofias seguidas pelos funcionários. Serve como um guia no caso de duvidas. Mas quando esses valores expostos funcionam, ou seja, tem sucesso repetidamente torna-se um pressuposto básico. Nesses casos as pessoas já assumem essas ideias como verdadeiras e raramente discutem sobre elas.

Ainda segundo SIQUEIRA (2005), essas ideias são incorporadas na organização de maneira paulatina, até que todos os funcionários assumam como verdadeiras e as mesmas sejam adotadas sem hesitação. Diante desse cenário surgem algumas modelos de negócios utilizados pelas organizações, são eles: *outsourcing, join-venture, insourcing, offshore, crowdsourcing, onshore, offshore outsourcing, offshore insourcing, onshore insourcing* e *onshore outsourcing*, que estão representados na tabela 2.1.

Tabela 2.1 - Modelo de negócios para DDS, adaptado de Prikladnic; Audy, 2008.

	LOCALIZAÇÃO GEOGRÁFICA	
CONTROLE E RELAÇÃO ENTRE AS EMPRESAS	Onshore "mesmo país"	Offshore "outro país"
Outsourcing "terceirizar"	Onshore Outsourcing	Offshore Outsourcing
Insourcing "departamento ou subsidiária"	Onshore Insourcing	Offshore Insourcing

- *Outsourcing* (terceirização): uma das formas mais simples de ser implantadas, pelo fato de ser mais simples de gerenciar. Nesse caso uma organização outorga a outra a gestão das atividades em questão a uma empresa contratada.

- *Join-venture*: neste caso ocorre a junção de duas ou mais organizações que unem seus recursos na criação de uma associação que ficará responsável pela criação e execução de alguns projetos por certo período.

- *Insourcing*: neste caso as organizações criam seus próprios núcleos de desenvolvimento de software. Entre as razões para se utilizar esse tipo de desenvolvimento pode-se citar redução de custos, maior agilidade e o fato de não haver problemas com relação à cultura da empresa.

- *Onshore*: nesse caso o desenvolvimento ocorre no mesmo país do cliente e da matriz da empresa contratada. Nesse contexto podem

acontecer duas situações: no primeiro caso, o desenvolvimento ocorre na empresa contratada, mas no mesmo país onde está o cliente, apenas distante fisicamente dele. No segundo caso o desenvolvimento acontece dentro da empresa do cliente, mas com todos os recursos da empresa contratada. Há a vantagem da proximidade do cliente, onde ele pode controlar as atividades e ser requisitado no caso onde há a necessidade de sua atenção imediata.

- *Onshore insourcing*: nesse caso a demanda de serviços de desenvolvimento de software é suprida por um departamento ou subsidiária criada pela própria empresa. O departamento ou subsidiária está localizada no mesmo país da empresa.

- *Onshore outsourcing*: ocorre a contratação de uma empresa terceirizada, que suprirá a demanda de serviços de desenvolvimento de software da empresa cliente.

- *Offshore*: nesse caso o desenvolvimento de software ocorre em um país diferente da empresa contratante e da contratada, ou seja, a própria organização pode montar sua equipe de desenvolvimento ou pode contratar outra, mas o que caracteriza essa prática é o fato de o desenvolvimento acorrer em um país diferente da matriz da contratante, ou se for o caso de terceirização, diferente da empresa contratada.

    - *Offshore outsourcing*: o desenvolvimento é realizado por uma empresa contratada, ou seja, terceirização, sendo que a empresa contratada está em um país diferente da empresa cliente.

- *Offshore insourcing*: a empresa cria uma subsidiária para suprir as necessidades de desenvolvimento de software. Essa subsidiária está impreterivelmente localizada em país diferente da matriz da empresa. (AUDY, 2008).

É importante observar que atualmente com as novas tendências criadas neste cenário, surge ainda uma nova visão de modelo de negócios entre as empresas chamada de *crowdsourcing, que consiste em um c*onjunto de pessoas ou organizações que se juntam com o objetivo encontrar soluções para os mais variados desafios, entre eles é possível citar o desenvolvimento de software.

Finalmente, é possível observar que neste capitulo o objetivo foi conceder uma visão geral com relação às principais características do desenvolvimento de software para diferenciá-lo dos demais modelos, evidenciando os fatores que levaram ao seu surgimento, os desafios que surgiram e as relações que foram criadas entre as empresas á partir do surgimento desse novo modelo.

3 FERRAMENTAS DE APOIO AO DDS

Durante o processo de desenvolvimento de software é imprescindível o uso de ferramentas, pois elas funcionam como suporte para o desenvolvimento do projeto, proporcionando um conjunto de serviços que colaboram com o processo de desenvolvimento. Dessa maneira as ferramentas reduzem o tempo de desenvolvimento do software sem, contudo prejudicar a qualidade do mesmo.

Ferramentas não são todas iguais e são usadas em fases diferentes durante o processo de desenvolvimento, mas algumas de suas vantagens são comuns, dentre elas pode-se citar: aumento da produtividade, melhor qualidade, facilidade de gerenciamento, redução de custo e facilidade de manutenção.

É importante observar que a escolha da ferramenta adequada para uso em uma determinada organização deve se adequar ás necessidades da mesma, pois a escolha da ferramenta errada pode impactar diretamente e de maneira negativa toda a estrutura do projeto.

O uso de ferramentas no DDS é parte fundamental do desenvolvimento, não só na comunicação, mas também no gerenciamento do projeto, controle de versão, edição e geração de código e testes. Dessa maneira o objetivo deste capítulo é fazer uma análise de algumas ferramentas de acordo com os recursos necessários para colaboração no DDS, que são: cooperação, coordenação, comunicação, suporte e gerenciamento.

3.1 Cooperação

Cooperar é trabalhar juntamente com outros para obtenção de um objetivo comum. Sendo assim, no DDS a cooperação é parte componente do processo, pois se trata de um modelo de desenvolvimento em grupo, ou seja, durante a cooperação os participantes do projeto trabalham em um espaço compartilhado, produzindo, manipulando e refinando objetos como planilhas, arquivos e códigos.

No DDS é possível citar três formas diferentes de cooperação: agendamento de atividades, edição colaborativa de código e compartilhamento de arquivos.

É essencial para uma ferramenta colaborativa possuir a função de agendamento de atividades, pois não só facilita o trabalho em grupo, mas também possibilita uma melhor gestão do projeto.

A edição colaborativa de código é parte fundamental de uma ferramenta colaborativa de software, pois possibilita que os desenvolvedores que se encontram dispersos geograficamente possam contribuir com desenvolvimento do produto.

O compartilhamento de arquivos em qualquer projeto que envolva mais de uma pessoa é necessário para que todos do grupo se mantenham atualizados do andamento do mesmo. No DDS esse compartilhando se torna mais essencial pelo fato dos atores envolvidos estarem distantes uns dos outros e na maioria das vezes essa distancia não é local, mas global.

Um exemplo a ser citado é a uma ferramenta elaborada por GÄRTNER (2011) chamada *idDE*, que foi idealizada para diminuir problemas no DDS. Funciona como um *plugin* para a IDE (Ambiente Integrado de Desenvolvimento) NetBeens que adiciona a mesma as funcionalidades de chat, conversação por áudio e desenvolvimento cooperativo.

3.2 **Comunicação**

Como já foi dito no capítulo anterior, a comunicação é indispensável em processos que envolvem atividades em grupo, pois através dela se torna possível acordar e discordar de propostas e opiniões mantendo não só a cooperação, mas também a qualidade do projeto.

Dessa forma é possível citar algumas formas de comunicação que devem existir em ferramentas voltadas para a colaboração no DDS: *chat*, fórum, mural de recados, email, vídeo conferência, conferencia por voz, quadro branco e identificação de usuários online.

Com o surgimento da internet, surgiu também o *chat,* que é uma conversação em tempo real, imprescindível quando se trata de alguma necessidade pontual (instantânea).

Já o fórum permite comunicação dinâmica na foram de enquetes, ou discussões. Destinados á pessoas com interesse comuns e consiste em informação não pontual (não instantânea), ou seja, uma pergunta feita em um fórum pode ser imediatamente respondida, ou pode levar dias para que se obtenha uma resposta.

O mural de recados permite que sejam agendadas tarefas e compromissos, dessa maneira fica mais intuitivo para os participantes do projeto visualiza-las, já que essas atividades estão em uma área especifica.

A vídeo conferência e a conferência por voz permitem que sejam feitas reuniões sem que os participantes estejam necessariamente no mesmo local, além de proporcionarem comunicação em tempo real.

Têm-se as vantagens de economia de tempo e recursos, pois não é necessário que um participante se desloque para que a reunião seja possível. A diferença entre a conferência por voz e vídeo é que na primeira não há visualização entre os participantes.

Outra forma de comunicação voltada para ambientes colaborativos é o quadro branco, que consistem em uma área compartilhada onde os participantes podem trocar informações, como códigos e arquivos de texto.

Um recurso muito importante na comunicação em um ambiente colaborativo é a identificação de usuários online, que possibilita comunicação assíncrona, auxiliando o emissor que tem uma necessidade pontual a escolher a forma de comunicação que melhor atenderá sua necessidade.

Um exemplo de ferramenta de comunicação é o *Skype*, que permite que seus usuários se comuniquem pela internet através de conexões de voz e vídeo, além de possibilitar conferência com até cem pessoas ao mesmo tempo.

3.3 **Coordenação**

Para garantir que os compromissos assumidos na comunicação serão cumpridos e a cooperação de cada membro da equipe seja possível , é necessário que haja coordenação para que as tarefas sejam realizadas de maneira adequada sem desperdício de tempo e recursos.

De acordo com RAPOSO & FUKS (2002, pag.4).

> A coordenação envolve a pré-articulação das tarefas, o gerenciamento do andamento das mesmas e a pós-articulação. A pré-articulação envolve as ações necessárias para preparar a colaboração, normalmente concluídas antes do trabalho colaborativo se iniciar: identificação dos objetivos, mapeamento destes objetivos em tarefas, seleção dos participantes, distribuição das tarefas entre eles, etc. A pós-articulação ocorre após o término das tarefas, e envolve a avaliação e análise das tarefas realizadas e a documentação do processo de colaboração (memória do processo).

Dessa forma é imprescindível gerenciar o agendamento de tarefas, pois é a etapa que mais muda ao longo do projeto, precisando ser renegociada ao longo de todo processo, pois além das mudanças continuas as tarefas possuem uma relação de interdependência uma com as outras.

Uma forma de controlar as atividades é utilizando o gráfico Gantt, que pode ser usado para controle do ciclo de vida de cada tarefa. Através dele é possível visualizar as tarefa de maneira individual e o tempo gasto por cada um colaboradores em uma tarefa especifica. Dessa maneira pode-se medir o desempenho dos colaboradores envolvidos, além de obter conclusões sobre os gastos no projeto.

Outra maneira de coordenar um projeto colaborativo é usar ferramentas de controle de versão, um sistema que seja capaz de registrar as mudanças feitas ao longo do tempo no projeto de forma que versões anteriores do projeto possam ser recuperadas, além de proporcionar controle das modificações realizadas no projeto no decorrer do tempo, dessa maneira é possível garantir a rastreabilidade e através do histórico que o sistema armazena.

Esse tipo de controle é considerado uma extensão natural no DDS, pois nesse tipo de desenvolvimento várias pessoas em lugares diferentes (países ou continentes diferentes) participam do desenvolvimento e só através dessa ferramenta é possível garantir que as modificações no projeto sejam feitas de maneira uniforme e harmoniosa.

O repositório do sistema de controle de versão funciona como seu núcleo, pois nele é que ficam armazenadas todas as modificações feitas durante o

desenvolvimento do projeto. Em geral o desenvolvedor não interage de modo direto com o repositório durante operações de escrita, apenas durante a operação de leitura.

Todas as modificações feitas nos projetos são realizadas sobre uma cópia do arquivo, dessa maneira o sistema realiza uma comparação entre o arquivo original e a cópia, a partir desse ponto é possível detectar as modificações feitas e integrá-las ao arquivo original. Mas em algumas tarefas como adição e cópias de arquivos precisam ser expressamente informadas em alguns controles de versão.

Quando os desenvolvedores alteram o mesmo arquivo, mas não a mesma linha de código não há nenhum problema, pois o sistema de controle de versão mescla as duas versões no arquivo e cria um novo arquivo atualizado com as duas versões. Mas quando os desenvolvedores alteram a mesma linha de código e de maneira diferente o sistema gera um conflito que só pode ser resolvido pelo desenvolvedor que fez as ultimas alterações, que deverá verificar as alterações feitas anteriormente e solucionar o possível conflito.

Existem três tipos de sistemas de controle de versão: sistema de controle de versão local, sistema de controle de versão centralizado e o sistema de controle de versão distribuído.

3.3.1 Sistema de controle de versão local

No sistema de controle de versão local os arquivos que são produzidos são armazenados em locais diferentes, ou seja, são armazenados em diretórios diferentes em um mesmo computador. Nesse tipo de controle de versão pode haver acidentes, pois arquivos podem ser copiados para diretórios errados, subescritos, ou mesmo apagados. Há também dificuldades na hora de encontrar uma versão antiga ou um arquivo modificado.

Para lidar com os problemas encontrados no controle de versão local, foram desenvolvidos controles de versão locais que controlam melhor as alterações feitas. Um exemplo é o *Revision Control System* (RCS) ou Sistema de Controle de Revisão que facilita a recuperação e registro das versões de arquivos. A figura 3.1 traz uma representação do sistema de controle de versão local.

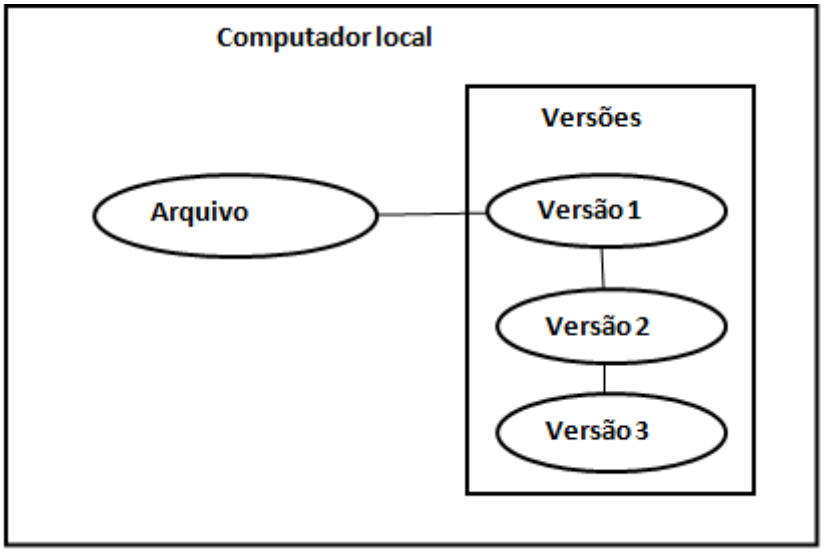


Figura 3.1 – Sistema de controle de versão local, adaptado de Chacon; Straub [3.1]

3.3.2 Sistema de cotrole de versão centralizado

O sistema de controle de versão centralizado foi idealizado para o caso em que há necessidade de trabalhar em grupo com outros desenvolvedores que utilizam outros sistemas. Nesse tipo de sistema um único servidor central possui todas as versões dos arquivos, e o desenvolvedor pode consultar a versão que desejar. Um exemplo desse tipo de sistema é o *Subversion*.

A grande desvantagem de usar um sistema de controle de versão de centralizado é fato do servidor central se tornar um ponto único de falha, ou seja, no caso de haver um problema com o servidor o trabalho será interrompido, se o problema for no disco do servidor perdem-se os dados, em alguns casos em que os desenvolvedores não possuem copias em suas máquinas locais, pode-se até perder todo o projeto .

A figura 3.2 traz uma representação do sistema de controle de versão centralizado.

[3.1] Disponível em:<http://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o> Acesso em out. 2014.

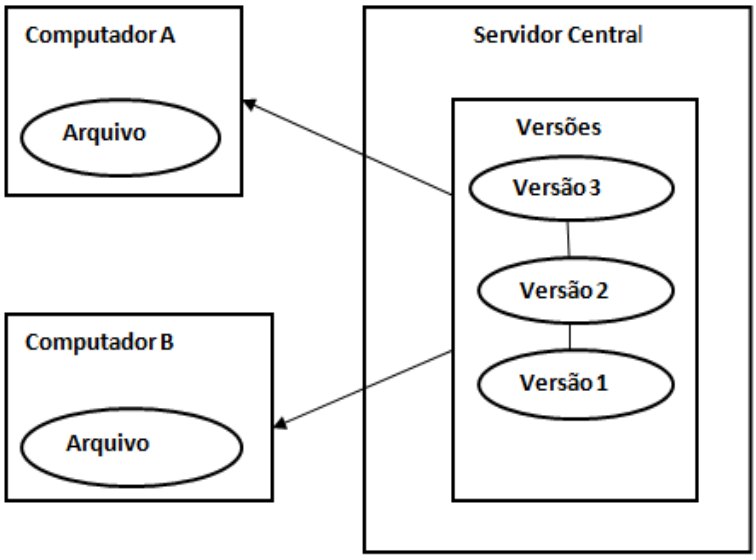


Figura 3. 2 – Sistema de controle de versão centralizado, adaptado de CHACON; STRAUB [3.2.]

3.3.3 Sistema de controle de versão distribuído

Neste cenário surgem os sistemas de controle de versão distribuídos, onde os desenvolvedores fazem cópias completas das ultimas versões dos arquivos para seus repositórios pessoais. Nesse caso se houver falha do servidor, qualquer um dos membros do grupo de desenvolvedores pode restaurar os arquivos para o servidor copiando o arquivo de volta.

Os sistemas de controle de versão distribuídos lidam bem com o fato de ter vários repositórios remotos com os quais pode colaborar, dessa maneira é possível trabalhar com pessoas no mesmo grupo ou em grupos diferentes ao mesmo tempo e no mesmo projeto.

Desse modo, segundo DIAS (2009) as principais vantagens de um controle de versão distribuído do ponto de vista do desenvolvedor são:

[3.2] Disponível em:<http://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o> Acesso em out. 2014.

- Rapidez (as operações são processadas localmente, não sendo necessário passar pelo servidor central para fazer um *commit*).
- Autossuficiência (não é necessário conexão com a rede para trocar revisões com outros repositórios, é possível trabalhar desconectado).
- Ramo privado (o desenvolvedor não necessita de sincronização entre os repositórios antes de cada *commit*).
- Simplicidade na mesclagem do código (rastreamento automático).

A figura 3.3 traz uma representação do sistema de controle de versão distribuído.

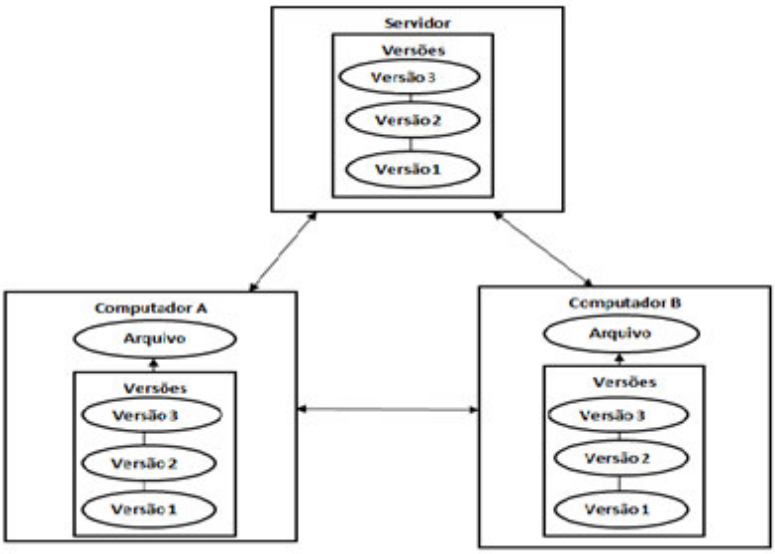


Figura 3.3 – Sistema de controle de versão distribuído, adaptado de CHACON; STRAUB [3.3]

[3.2] Disponível em:<http://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o> Acesso em out. 2014.

Um exemplo de sistema de controle de versão distribuído é o GIT, que não trata a informação como um conjunto de arquivos que é atualizado à medida que sofre algum de mudança, mas age como se tirasse uma foto do arquivo toda vez que o ele é atualizado, ou seja, salva (*commit*) uma referência para essa captura. No caso de não haver nenhuma modificação a informação não é novamente armazenada, mas cria-se um *link* para o arquivo anterior que já foi armazenado (CHACON; STRAUB, 2014).

A maioria das operações do GIT necessita apenas de recursos e arquivos locais para funcionar, ou seja, não é necessária outra informação de algum computador da rede. No caso de haver necessidade de navegação no histórico do projeto o GIT lê diretamente do banco de dados local do desenvolvedor, dessa maneira o histórico pode ser visto quase imediatamente. O próprio GIT faz o cálculo da diferença da data localmente, sem necessidade de fazer qualquer solicitação ao servidor.

O GIT pode ser instalado com facilidade no Linux, Windows ou Mac á partir do código fonte ou através de pacotes disponibilizados para uma dessas plataformas. É imprescindível se cadastrar no GIT durante a instalação, pois cada *commit* feito estará associado as suas informações, ou seja, é possível saber quem modificou os arquivos e a data da modificação, bem como o percentual de modificação feito por cada pessoa participante do projeto.

3.4 Gerenciamento

Ferramentas de gerenciamento de projeto tornam possível o controle da equipe de desenvolvimento, auxiliam no controle de custos e nos prazos, além de tornar possível o acompanhamento de todo projeto, padronização de métodos e processos de trabalho.

Dessa maneira a utilização de ferramentas de gerenciamento de projeto no DDS é indispensável, pois é através delas que as metas definidas no inicio do projeto poderão ser alcançadas, pois só dessa maneira os gerentes de projeto poderão coordenar toda uma equipe que se encontra fisicamente distribuída.

A escolha da ferramenta certa faz toda a diferença durante o processo, pois pode proporcionar tanto vantagens como desvantagens. É importante escolher uma ferramenta que seja adequada ao processo, e esse discernimento muitas vezes só se alcança com experiência na área de gestão.

Mas alguns requisitos são indispensáveis quando se trata de desenvolvimento colaborativo. Uma ferramenta deve proporcionar acoplamento com outras ferramentas diferentes que possam ajudar no desenvolvimento das tarefas, assim como deve ser possível a adição de novas funcionalidades de acordo com a necessidade da equipe.

Entre as principais ferramentas de gerenciamento de projeto está o *Project-Open* que é um aplicativo baseado na web e de código aberto que permite que o projeto seja gerenciado através da utilização do gráfico de Gantt, que é usado para exibir o progresso de cada etapa do projeto. Dessa maneira é possível calcular os custos e ganhos com cada projeto e empresa.

As principais finalidades do *Project-Open* são a colaboração entre os membros das equipes do projeto através de *chats* e página Web editável pelo usuário (*Wiki*) e controle de gastos. Mas a ferramenta possui outras funcionalidades como noticias e pesquisas.

O acesso a cada projeto ou tarefa pode ser controlado através das permissões, dessa maneira os envolvidos no projeto só podem ver as conversas, listas de clientes e relatórios financeiros do projeto se for concedida autorização pelo gestor. É possível também verificar a contribuição de cada um dos participantes da equipe e esse controle é feito através de perfis.

O *Project-Open* possui uma arquitetura padrão de aplicativos Web, ou seja, um servidor web de código aberto com um banco de dados SQL também aberto, além de um *kit* de ferramentas chamado *OpenACS* que é usado na construção de aplicativos escaláveis orientados para a comunidade web. Dessa maneira é possível observar na figura 3.4 como a arquitetura do Project-Open está organizada.

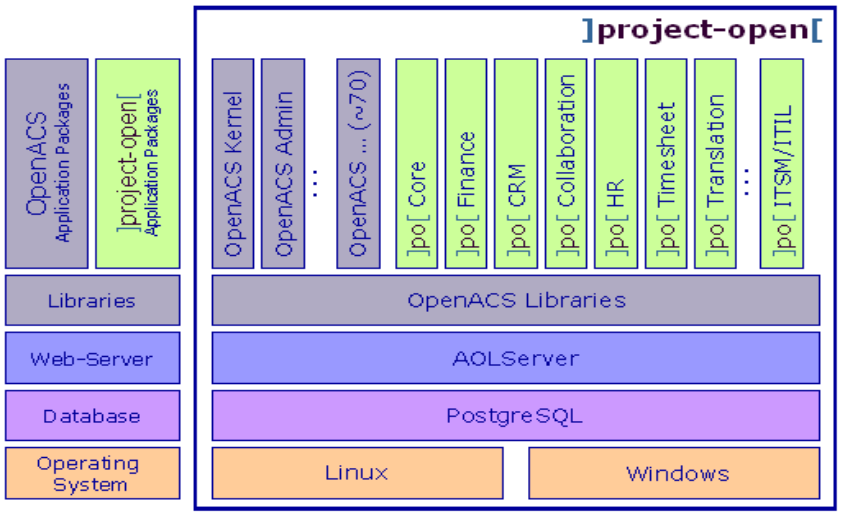


Figura 3.4 – Arquitetura Project Open, segundo PROJECT-OPEN.ORG [3.4]

O *Project-Open* possui suporte tanto para os sistemas operacionais Windows e Linux. O banco de dados é o *PostgreSQL*, que é *open source*, mas também há suporte para *Oracle*. O servidor de web usado é o *AOLserve*r e na camada acima usa uma biblioteca que consiste em um conjunto de ferramentas que ajudam na construção de aplicativos que são voltados para comunidade web.

Project-Open está dividido em pacotes que podem ser instalados e desinstalados de acordo com as necessidades da equipe, através do gerenciador de pacotes. Um pacote gerencia um ou mais processos de negócios. Alguns pacotes estão disponíveis para download, exceto alguns que não são de código aberto. Dessa maneira os pacotes são agrupados por função, dentre eles é possível citar:

- Núcleo: define a interface gráfica do usuário, exibindo os projetos, as empresas com as quais se relaciona, além de menus e componentes.

[3.4] Disponível em:< http://www.project-open.org/en/project_open_architecture> Acesso em fev. 2015.

- Finanças: calcula o lucro e a perda em tempo real, com relatórios e funções estatísticas. Contém uma grande quantidade de serviços de recursos humanos não só para os funcionários que estão desenvolvendo o projeto, mas também para os profissionais associados, dessa maneira o módulo está divido em dois paradigmas: o virtual onde a importância da presença física é diminuída devido o uso da internet, e o paradigma onde cada funcionário é responsável pelo processo de recursos humanos.

- Gestão de quadro de horários como o próprio nome já supõe, gerencia o quadro de horários para um melhor controle do projeto.
- Gerenciamento de serviço de TI: possui funcionalidades específicas de acordo com os padrões de projetos UTIL, COBTI e PRINCE, sendo possível escolher entre eles. Além desses padrões de projeto outros podem ser instalados de acordo com a necessidade do projeto.
- Relatórios: á partir desse módulo é possível obter relatórios corporativos do banco de dados de acordo com os parâmetros definidos.
- Recursos humanos: pacote base para o armazenamento e gerenciamento de informações do usuário, tanto os da empresa quando dos clientes. Além disso, armazena também as informações de empregados temporários, suas habilidades e o trabalho desenvolvido na empresa.
- Gestão de conhecimento: não existe nenhum módulo exclusivo, ou seja, deve haver compartilhamento de conhecimento em todos os módulos. Dessa maneira cada módulo possui algumas características que permitem esse compartilhamento, são elas: fórum de discussão, email, edição colaborativa de documentos (*wiki*), buscas simples, índices, calendário, avaliação, aprovação e implementação de ideias.

O *Project-Open* suporta idiomas como o inglês, francês, alemão, italiano, húngaro, russo e búlgaro. Mas alguns idiomas como o português do Brasil, japonês e turco ainda estão em fase de desenvolvimento.

Os idiomas como português de Portugal, árabe, finlandês, coreano, romeno esloveno e sueco ainda estão em fase de análise para serem suportados pela ferramenta.

Mas por ser de código aberto, o módulo de tradução pode ser modificado para se adaptar á necessidade de cada organização através de um guia de localização fornecido pela ferramenta, que possibilita aos interessados contribuir com a tradução para a língua desejada.

3.5 Suporte

No desenvolvimento de software de em geral, é necessário que haja suporte para armazenamento de tudo o que produzido por cada membro da equipe. No DDS esse suporte é ainda mais fundamental dado o fato de que os componentes do grupo encontram-se dispersos geograficamente, nesse caso é essencial que exista um espaço pessoal para armazenamento de arquivos, para que futuramente possa ser compartilhado com todo equipe.

Outro desafio encontrado no DDS é o suporte para trabalho móvel. Algumas ferramentas de comunicação como o *Skype* podem ser usadas aliadas á ferramentas que oferecem espaço para armazenamento pessoal, para que o usuário possa ter o suporte necessário para desenvolver seus projetos com mobilidade.

3.6 Análise das ferramentas

Não existe atualmente uma ferramenta que possa ser usada em todas as fases do desenvolvimento de software, o tem-se é o uso em conjunto dessas fermentas. A análise das ferramentas foi feita baseado nas funcionalidades ou recursos necessários para colaboração no DDS segundo SILVA (2013).

É possível observar na figura 3.3 que a ferramenta que possui os recursos necessários para colaboração no DDS entre as ferramentas citadas neste capítulo é a Project Open. Logo, foi a ferramenta escolhida para aplicação em um estudo de caso a ser realizado no próximo capítulo.

Tabela 3.6 – Análise de ferramentas, adaptado de Silva, 2013.

			idDE	GIT	Project Open	SKYPE
Funcionalidades/recursos necessários para colaboração no DDS	Cooperação	Agendamento de atividades	x			
		Edição colaborativa de código	x	x	x	
		Compartilhamento de arquivos	x	x	x	
	Comunicação	Chat	x		x	x
		Fórum			x	
		Mural de recados			x	
		Email vídeo conferência				x
		Conferência por voz	x			x
		Quadro branco			x	
		Identificar usuário online	x			x
	Coordenação	Gerenciamento de atividades	x		x	
		Visualização de atividade através do Gantt			x	
		Controle de versão		x		
	Gerenciamento	Escolhas das ferramentas			x	
		Adicionar novas funcionalidades			x	
		Acoplamento de ferramentas diferentes				
	Suporte	Suporte para trabalho móvel	x	x		x
		Espaço pessoal para armazenar arquivos		x	x	

4 ESTUDO DE CASO - DESENVOLVENDO O SISTEMA SGDU UTILIZANDO O MODELO DISTRIBUÍDO

No estudo de caso proposto neste trabalho o objetivo é descrever o processo de desenvolvimento de software distribuído, utilizando como objeto de estudo o projeto Sistema de Gerenciamento de Documentos UFMA (SGDU) desenvolvido anteriormente usando o modelo tradicional, ou seja, onde os atores estão próximos fisicamente. Para isso utilizou-se a ferramenta de gerenciamento de projeto *Project-Open* como ferramenta CASE para desenvolver as atividades necessárias do projeto, desde a fase de definição até a fase de suporte.

O Sistema de Gerenciamento de Documentos UFMA (SGDU), surgiu da necessidade de uma ferramenta capaz de facilitar a organização, compartilhamento, e o acesso a documentos da Universidade Federal do Maranhão (UFMA). Dessa maneira essa solução afeta tanto alunos como servidores da UFMA (Apêndice A).

É importante ressaltar que foi usada a versão *demo* da ferramenta, ou seja, uma versão de demonstração para possíveis futuros clientes. O motivo do uso dessa versão e não a completa, é que apesar de ser uma ferramenta *open source* alguns pacotes importantes não são gratuitos, dessa maneira a demonstração completa do uso da ferramenta só seria possível através da versão demo disponível no site (http://www.project-open.org/).

Neste estudo de caso serão seguidas as fases genéricas do desenvolvimento de software, ou seja, fases que compõem todos os modelos de desenvolvimento e podem ser aplicadas em qualquer domínio. São elas: definição, desenvolvimento e suporte.

4.1 Definição do projeto

Nessa fase o objetivo é ter uma visão geral do que será desenvolvido, encontram-se três etapas: análise do sistema (define a função que o software irá desempenhar), planejamento de projeto de software (análise de custos, riscos e

alocação de recursos) e análise de requisitos (informação detalhada do domínio e da função do software).

Sendo assim, para dar inicio á fase de definição do projeto utilizando o *Project-Open* é necessária à criação de um projeto, que pode ser vista na figura 4.1, onde a ferramenta oferece várias opções de projetos, nesse caso será escolhida a opção desenvolvimento de software.

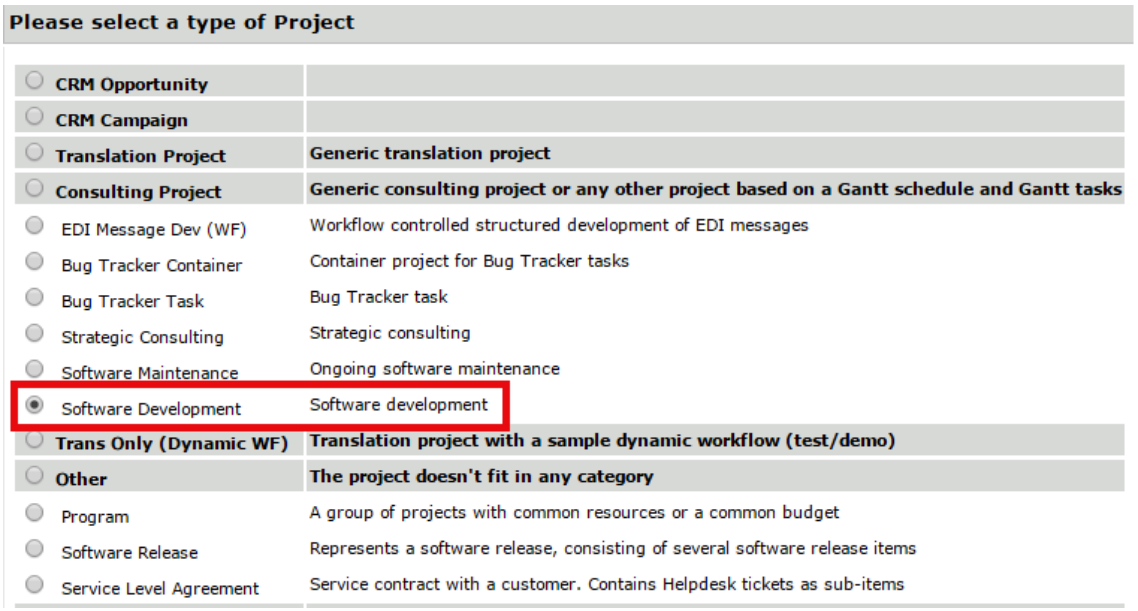


Figura 4.1 – Criação de um projeto usando Project Open.

A seguir são necessárias informações, como inicio e fim do projeto, nome do projeto, cliente, responsável pelo gerenciamento do mesmo entre outras informações que são fundamentais para identificá-lo. Segue a descrição na figura 4.2.

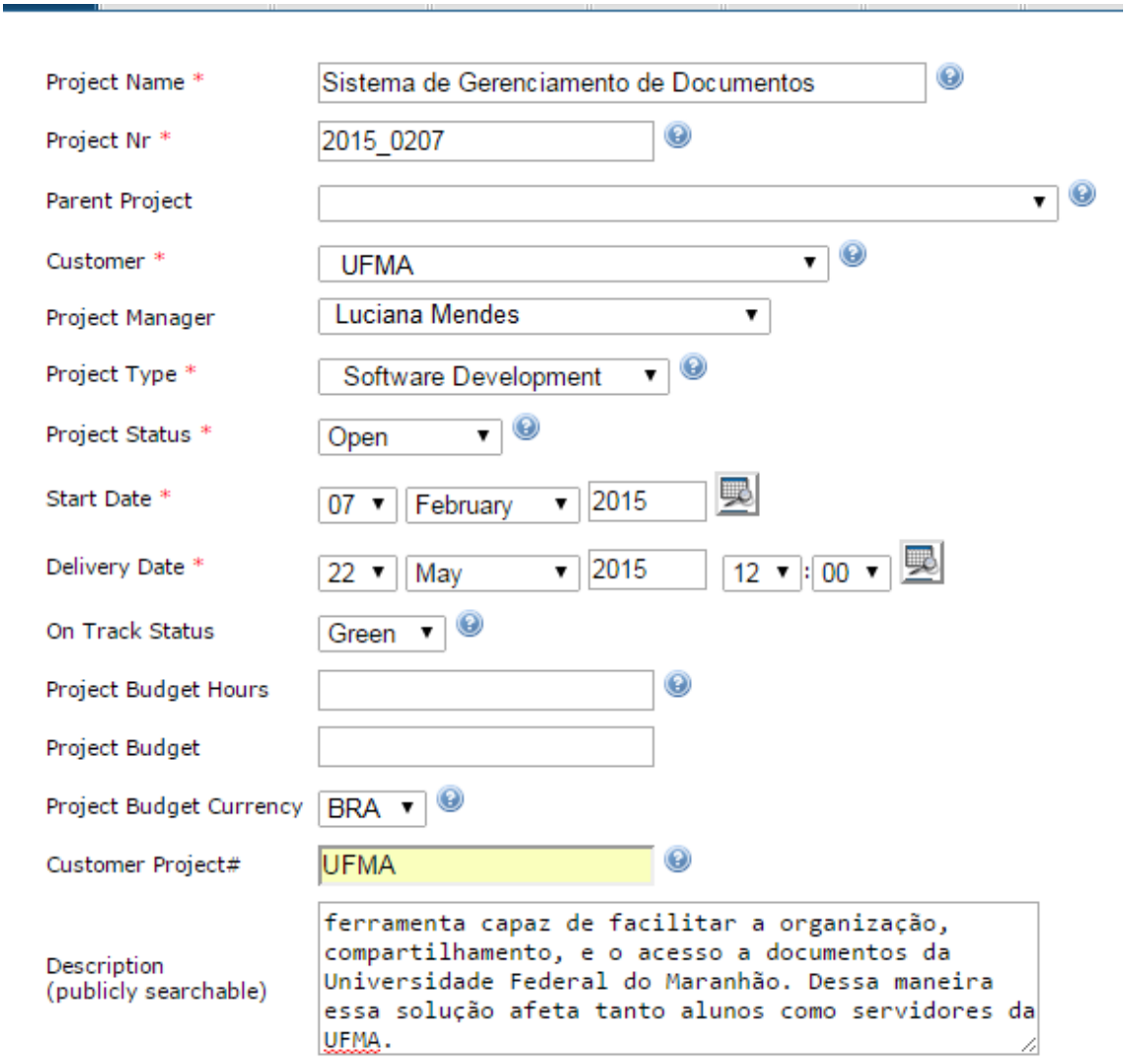


Figura 4.2 – Descrição do projeto no Project Open.

Com a criação do projeto, segue-se a alocação de mão de obra para desenvolvê-lo, cada um com permissão de acordo com seu perfil. O *Project-Open* tem perfis de administrador, chefe geral do projeto, desenvolvedor entre outros. Todos no projeto terão o mesmo perfil, exceto a pessoa que cria o projeto, pois a ferramenta automaticamente atribui a ele o perfil de gestor do projeto, figura 4.3. Aos demais coube o perfil de administrador, para que todos possam alterar qualquer parte do projeto e criar discussões sem precisar solicitar permissão ao gestor do mesmo.

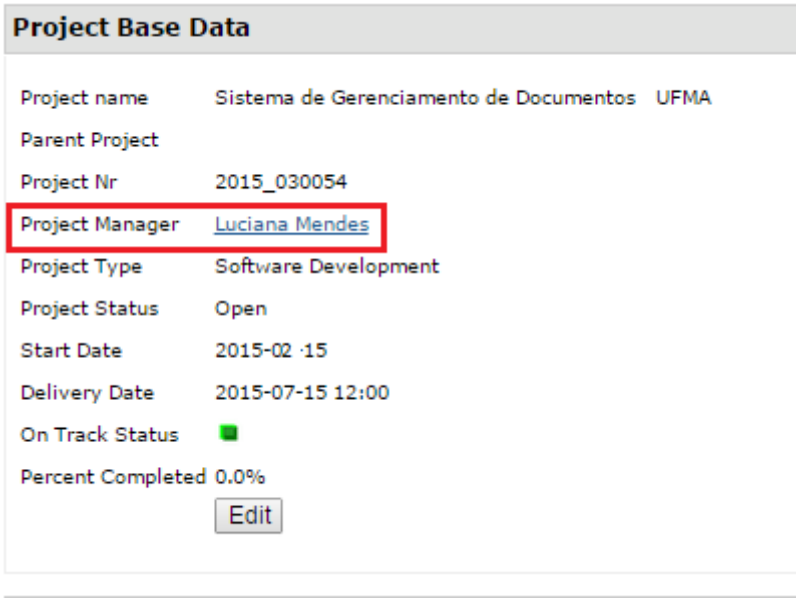


Figura 4.3 - Perfil de gestor do projeto

Sendo assim, a inserção de participantes no projeto é feita, Figura 4.4, escolhendo-se a opção **Add Members** no menu direito vertical **Project Members**. Na mesma opção é possível também apagar membros do grupo.

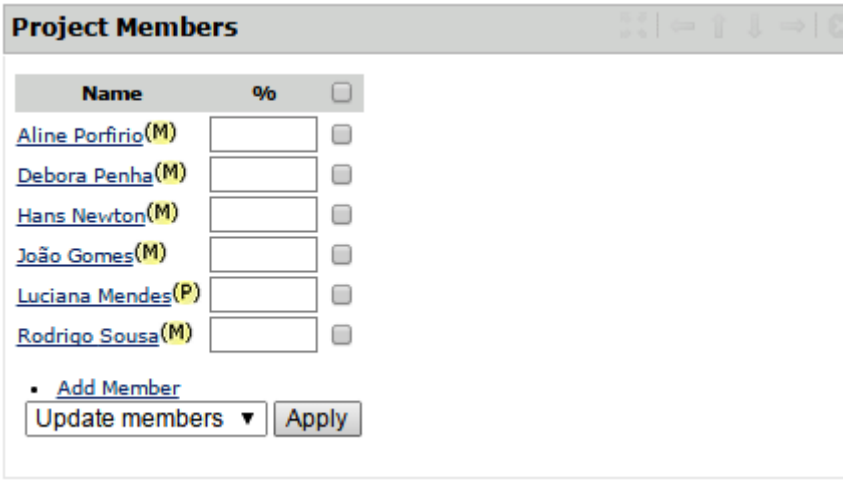


Figura 4.4 – adição de membros no Project Open.

Em seguida, de acordo com a fase de definição do projeto é feita a análise do sistema. Como essa etapa consiste em definir as funções que o software irá desempenhar, foi importante a participação de todos os membros da equipe.

Dessa maneira, foi criado um fórum de discussão para que os requisitos funcionais do projeto pudessem ser levantados e discutidos com todos, figura 4.5.

Após todos concordarem com os requisitos, foi criado um documento de requisitos que foi colocado em uma pasta, figura 4.6, compartilhada dentro do projeto para que pudessem ser consultada sempre que necessário.

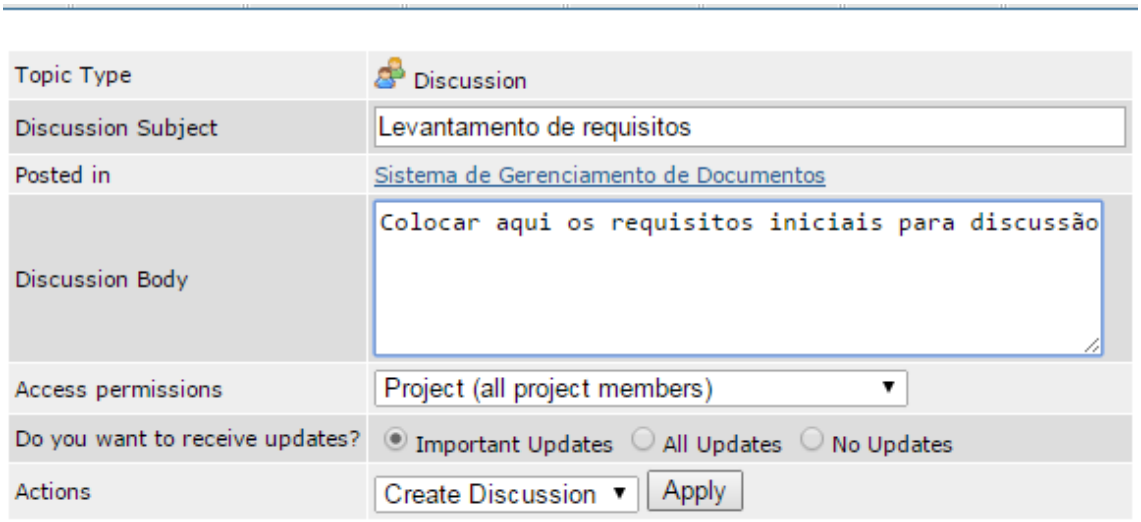


Figura 4.5- Criar fórum de discussão no Project Open.

Figura 4.6-Organização dos requisitos em pastas

Após o levantamento dos requisitos é fundamental que seja feita a análise dos mesmos. Dessa maneira foram designados três membros da equipe para essa tarefa. A eles coube a elaboração do diagrama de casos de uso (apêndice A), para que á partir dele as principais funcionalidades do sistema ficassem claras. Houve ainda a necessidade do diagrama de sequencia de algumas funcionalidades para que elas pudessem ser melhor entendidas, essa tarefa coube ao restante da equipe.

Ainda com relação ao levantamento de requisitos todos os participantes da equipe podem enviar sugestões através do fórum de discussão, figura 4.7. A ferramenta permite a identificação dos membros que enviarem resposta aos tópicos criados.

Figura 4.7 – Requisitos funcionais do projeto SGDU.

4.2 Desenvolvimento

Nessa fase de desenvolvimento a concentração é em como tornar o projeto real, ou seja, definir como os dados serão estruturados, que arquitetura será usada, se os testes serão realizados e como serão realizados.

Sendo assim, toda definição de arquitetura do projeto foi feita por meio de discussão no *chat.* Em seguida gerou-se um documento onde constam os casos de usos, assim como todo detalhamento dos mesmos (Apêndice A). Á partir desse ponto criaram-se classes, pois o projeto foi feito usando o modelo de orientação a objetos, onde cada classe representa um caso de uso.

O *Project-Open* dispõe de alguns módulos chamados de integração que permite que várias ferramentas sejam a ele incorporadas. Um desses módulos possui um *plugin* que permite a integração com a ferramenta eclipse, que foi a escolhida para edição de código do projeto, figura 4.8. O plugin chama-se *Mylyn* e é a estrutura de gerenciamento de ciclo de vida de tarefa e de aplicação para a ferramenta eclipse usada pelo *Project-Open*. Através dele os desenvolvedores se conectam a um servidor *Project-Open* para revisar, atualizar e criar tarefas.

Sendo assim, as tarefas do projeto foram distribuídas entre os membros da equipe, onde cada membro só tem acesso à tarefa atribuída a ele. Ao final de cada tarefa cumprida por um membro, a equipe era avisada via chat.

Ao final de uma tarefa por um membro da equipe, uma nova tarefa era repassada ao mesmo, até que todas as tarefas fossem finalizadas. Em seguida, todo o código foi organizado para que fosse possível observar e analisar o resultado, dando inicio á fase de testes.

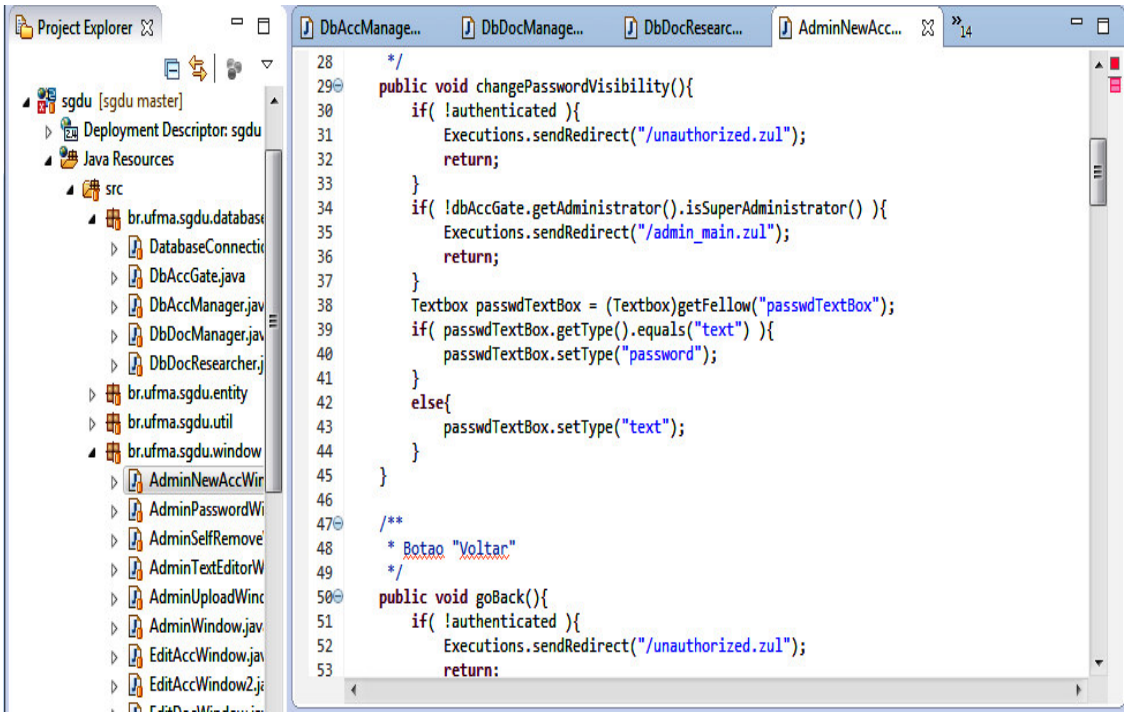


Figura 4.8 Desenvolvimento do projeto usando eclipse.

Na definição dos testes, ficou acordado que seriam realizados teste de segurança, para verificar se apenas os autorizados teriam acesso ao sistema, testes funcionais, para observar se os requisitos funcionais são atendidos, testes de volume, para observar o funcionamento do sistema operando com volume de informações considerado normal e teste de usabilidade que tem como foco a experiencia dos usuários, nesse caso cada um dos membros testou o software com relação ao *layout* e acesso ás funcionalidades. Para tais atividades não foi feito o uso da ferramenta *Project-Open,* pois os testes descritos anteriormente não exigiram o uso da mesma. Mas a ferramenta possui um *plugin* que permite sua integração com ferramentas de testes, entre elas é possível citar o *Bugzilla*.

4.3 Suporte

Mesmo após a fase de testes de software, alguns erros só surgem depois da implantação e uso operacional do mesmo. Em alguns casos novas funcionalidades são requeridas pelos usuários. Nesse caso é necessário aplicar novamente os ciclos tanto da fase de definição quanto do desenvolvimento.

Como visto anteriormente na figura 4.2, no *Project-Open* é possível adicionar uma empresa cliente e associá-la ao projeto de forma que a empresa cliente possa contatar a empresa prestadora de serviço.

Durante todas as fases do projeto a empresa cliente pode acompanhar o desenvolvimento do mesmo. No caso em que há problemas no software ou mesmo solicitação de adição de alguma funcionalidade, a empresa cliente pode entrar em contato com a empresa prestadora de serviço e informar através de *chat* sua solictação. Após a criação de um novo tópico no fórum os participantes do projeto podem ser escolhidos e um email é enviado para que sejam notificados da abertura de uma nova discussão,Figura 4.9.

Figura 4.9 - Envio de notificação para os participantes do projeto

Ao final de todas as etapas descritas anteriormente neste capítulo, o projeto desenvolvido apresentou alguns problemas com relação á interface, pois a mesma não foi considerada intuitiva o suficiente por alguns membros da equipe. Dessa forma foi criado um novo tópico de discussão, figura 4.10, para que fossem dadas sugestões e fosse possível chegar a um acordo.

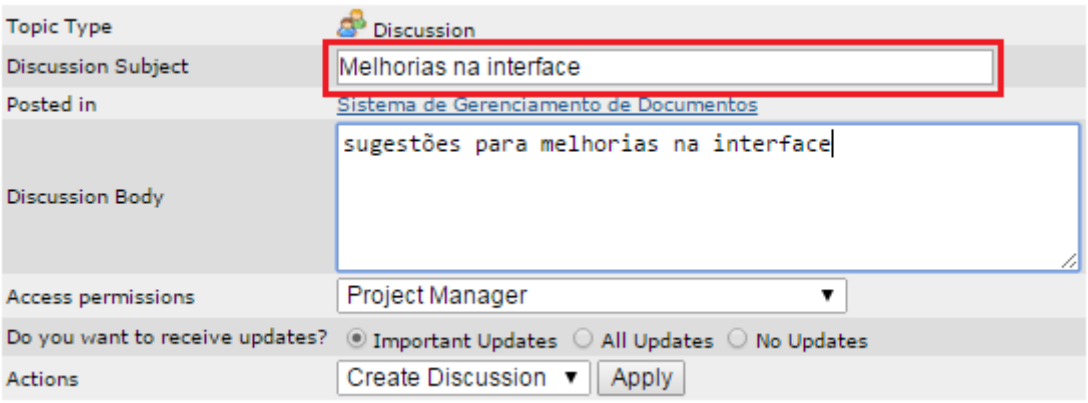


Figura 4.10 – Nova discussão sobre melhorias

Portanto, após a discussão foi possível entrar em acordo com relação á interface do projeto. Nesse caso não houve necessidade de aplicar novamente o ciclo da fase de desenvolvimento, pois essa tarefa coube ao membro que estava encarregado anteriormente dela. Dessa maneira foi possível chegar á finalização da fase de suporte, figura 4.11.

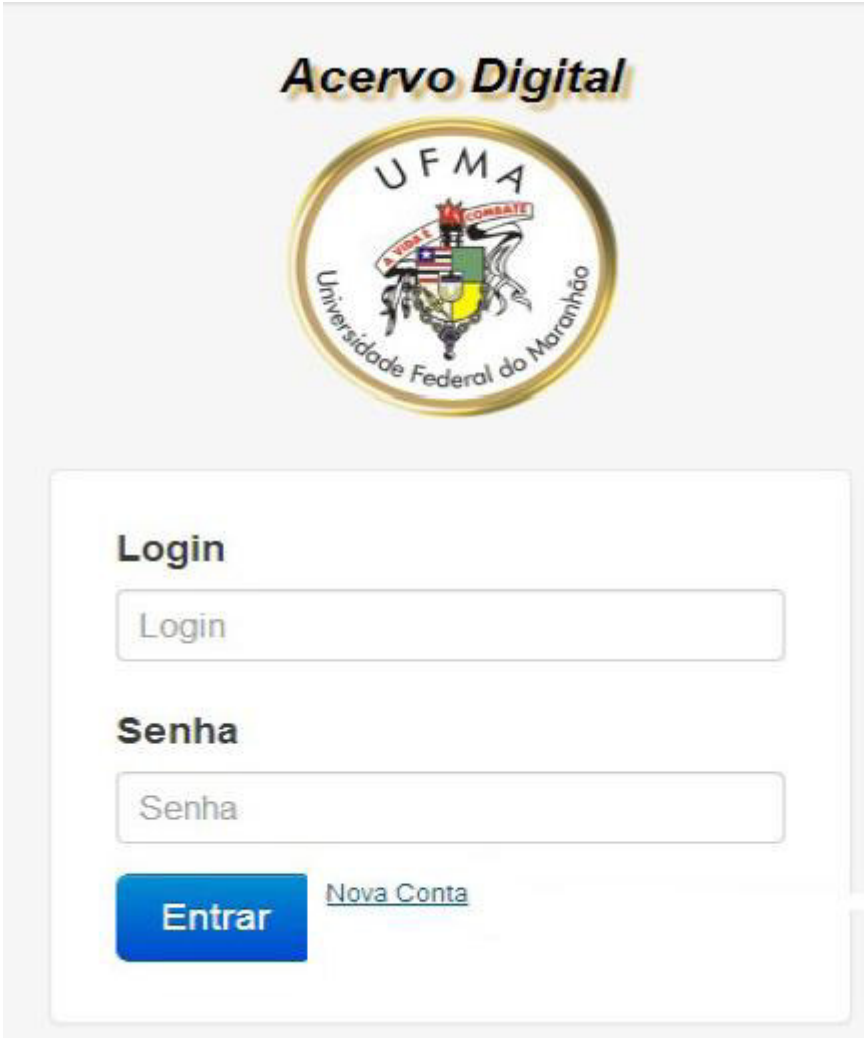


figura 4.11- Interface projeto SGDU.

Com a finalização do projeto foi possível alcançar o objetivo proposto neste capitulo, ou seja, desenvolver um software usando o modelo distribuído com o auxilio da ferramenta *Project-Open,* e dessa maneira não só apresentar as dificuldades, mas também as vantagens de fazer uso desse novo modelo.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um projeto de software, usando o modelo de desenvolvimento distribuído. Para isso, foi usada uma ferramenta parcialmente gratuita, *Project-Open*. A opção pela ferramenta em questão foi o fato da mesma ser completamente voltada para o trabalho colaborativo.

O passo inicial consistiu em cadastrar todos os participantes do projeto na ferramenta para que as tarefas pudessem ser distribuídas entre os mesmos. Após o cadastro de todos, foi feita a criação do projeto usando a ferramenta para que fosse dado inicio ao processo de desenvolvimento. Durante essa fase inicial não foram encontrados obstáculos.

Durante a fase de levantamento de requisitos toda comunicação foi feita usando o *chat* da ferramenta, o que causou um pouco de atraso na comunicação e definição dos requisitos, pois alguns dos participantes do projeto não puderam responder as perguntas solicitadas de maneira pontual por não estarem disponíveis.

A etapa de desenvolvimento foi feita de forma prática usando o *plugin* para a IDE eclipse, o mesmo possibilitou que as tarefas fossem divididas de maneira que os membros da equipe não ficassem sobrecarregados. Houve dificuldade em usar o controle de versão pelo fato de nenhum dos membros da equipe possuir conhecimentos suficientes sobre o mesmo.

A fase de testes não foi feita usando a ferramenta, pois não foi julgado necessário por não se tratar de um projeto grande. Sendo assim, os testes foram feitos por todos os membros da equipe de forma manual. Mas a ferramenta dispõe de *plugins* para integração com outras ferramentas de testes.

Na fase de suporte foram encontrados alguns problemas com relação á interface, pois a mesma não foi considerada intuitiva o bastante, contudo o problema foi resolvido sem muita dificuldade.

De maneira geral, a ferramenta se mostrou bastante eficiente mesmo em sua versão demo. Foi possível desenvolver todo o projeto de software de maneira

distribuída sem muita dificuldade, pois a ferramenta possui muitos recursos para o trabalho colaborativo. Também foi possível observar as vantagens do desenvolvimento distribuído de software, não só com relação á colaboração, mas também ao que se refere á qualidade, agilidade e customização do software.

É importante ressaltar o considerável valor do trabalho colaborativo no desenvolvimento de software, que possibilita que as tarefas sejam realizadas em um tempo mais curto e de maneira mais eficiente, principalmente o modelo *crowdsourcing,* ficando dessa forma a sugestão para a realização de trabalhos científicos sobre esse novo modelo no desenvolvimento distribuído de software.

REFERÊNCIAS

AFFONSO, Frank José. **Metodologia para Desenvolvimento de Software Reconfigurável Apoiada por Ferramentas de Implementação**: uma aplicação em ambiente de execução distribuído e reconfigurável. 2009. Tese (Doutorado em Processamento de Sinais de Instrumentação) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18152/tde-02072009-095730/>. Acesso em: 2014-10-30.

Bezerra, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. 2ª edição, Rio de Janeiro: Campus Elsevier, 2006

CAMARGO, Álvaro Antônio Bueno De. KHOURI, Lourdes Halim El e GIAROLA, Paulo César. **O Uso de Sistemas Colaborativos na Gestão de Projetos: Fatores Relevantes para o Sucesso.** Trabalho de Conclusão de Curso**.** Fundação Instituto de Administração – FIA. 2005.

ESPINDOLA, Rodrigo Santos et al. **Uma Abordagem Baseada em Gestão do Conhecimento para Gerência de Requisitos em Desenvolvimento Distribuído de Software**. In: **WER**.Rio Grande do Sul, 2005. p. 87-99.

GÄRTNER, Vilson Cristiano. **O desenvolvimento distribuído de software: Características e Recomendações**. Dissertação (Mestrado em Computação). Universidade do Vale do Rio dos Sinos, São Leopoldo, 2011.

HUZITA, Elisa Hatsue Moriya et al. **Um conjunto de soluções para apoiar o desenvolvimento distribuído de software**. In: Proceedings of the Workshop de Desenvolvimento Distribuído de Software-II WDDS. Salvador, 2008. p. 101-110.

Koscianski, André; Soares, Michel dos Santos. **Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2ª edição. São Paulo: Novatec, 2007.

MOLINARI Leonardo. **Teste de Software** – Produzindo sistema melhores e mais confiáveis. 4ª ed. São Paulo: Editora Érica Ltda., 2012, 228 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis Nicolas. **Desenvolvimento distribuído de software**. Elsevier, 2008.

Pimental, Mariano; Hugo Fuks. **Sistemas Colaborativos** – Rio de janeiro: Campus, 2011.

Site: GIT- Documentação. Acesso em 20 de Outubro de 2014, disponível em: http://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Versão.

Site: Projetc-open. Acesso em 20 de março de 2015, disponível em: http://www.project-open.org/

PRIKLADNICKI, R.; AUDY, J. L. N. **Um Modelo de Referência para Desenvolvimento Distribuído de Software**. Dissertação (Mestrado em Ciência da Computação). Pontifícia Universidade Católica do Rio Grande do Sul, Faculdade de Informática Programa de Pós-Graduação em Ciência da Computação, Rio Grande do Sul, 2003.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis Nicolas. **Uma Análise Comparativa de práticas de Desenvolvimento Distribuído de Software no Brasil e no exterior**. XX Simpósio Brasileiro de Engenharia de Software, Porto Alegre, p. 255-270, 2006.

RAPOSO, H; FUKS, A.B. **O Modelo de Colaboração 3C e a Engenharia de Groupware.** Pontifícia Universidade Católica do Rio De Janeiro, Rio de Janeiro, 2002.

SIQUEIRA, Fábio Levi. **O Desenvolvimento Distribuído de Software: Características e Recomendações para Gerencia de Projetos**. Dissertação (Mestrado em Engenharia). Escola Politécnica da Cidade de São Paulo, São Paulo, 1995.

SIQUEIRA, Fábio Levy; SILVA, Paulo Sérgio Muniz. As características do desenvolvimento distribuído de software. **I Simpósio Brasileiro de Sistemas de Informação–SBSI**, p. 171-178, 2004.

SOUZA, Carla Pires; GASPAROTO, Angelita Moutin Segoria. **A Importância da Atividade de Teste no Desenvolvimento de Software.**. XXXIII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO, 2008.

SILVA, Maurício Severo da. CollabCode: **Ferramenta para Apoio ao Desenvolvimento Distribuído e Colaborativo de Software.** Trabalho de conclusão de curso, UNIVATES, Rio Grande do Sul, 2013.

Wazlawick, Raul Sidnei**. Análise e projeto de sistemas de informação orientados a objetos**. 2ª edição. Rio de Janeiro: Campus Elsevier, 2011.

APÊNDICES

APÊNDICE A- Documento de Requisitos do Sistema de Gerenciamento de Documentos (SGDU).

Documento de Requisitos

Projeto **Data**	Acervo Digital da UFMA 24/10/13
Responsável	Aline, Débora, Hans, João, Luciana, Rodrigo.
Versão	1.1.0

Histórico de Revisão

Data	Versão	Autor	Descrição
03/10/13	1.0.0	Aline, João, Débora, Luciana, Hans, Rodrigo	Iniciar a descrição; Especificação do problema e possível solução; Descrever stakeholders e suas responsabilidades; Especificar e detalhar requisitos.
18/10/13	1.1.0	Aline, Débora, Hans, João, Luciana, Rodrigo.	Inclusão nos requisitos funcionais, criação de casos de uso.
24/10/13	1.1.1	Aline, Débora, Hans, João, Luciana, Rodrigo.	Alterações nos requisitos, alterações nos casos de uso, alterações no detalhamento dos casos de uso.

1 DEFINIÇÃO DO PROBLEMA

Conforme foi proposto para a equipe da necessidade de organização e acesso aos documentos da Universidade Federal do Maranhão é fundamental o desenvolvimento de um ambiente digital que facilite e permita o gerenciamento e acesso a esses documentos desta instituição. Em síntese pode-se visualizar o problema, os afetados e a solução na Tabela 1.

Tabela 1 – Análise do problema identificado

O Problema	Necessidade de uma ferramenta capaz de facilitar a organização, compartilhamento, e o acesso à documentos da Universidade Federal do Maranhão.
Quem é afetado	Servidores da UFMA e alunos.
Uma solução	Construção de seu acervo digital que consiste em um repositório de documentos e arquivos categorizados em diversas comunidades e será acessível a todos os usuários que poderão além de consultar, submeter documentos acadêmicos com a supervisão do administrador do acervo.

Fonte: arquivo do autor (2013)

2 STAKEHOLDERS

Na engenharia de requisitos, os *stakeholders* são definidos como pessoas ou organizações que serão afetadas pelo sistema e que direta ou indiretamente tem influência sobre os requisitos, ou seja, os interessados pelo sistema.

Basicamente, os *stakeholders* do Projeto *Acervo Digital da UFMA* podem ser divididos em dois grupos: os **desenvolvedores** (responsáveis pelo projeto e desenvolvimento de artefatos do sistema) e os **usuários finais** (utilizarão o sistema já desenvolvido), e são listados a seguir:

- **Desenvolvedores**: programadores, administradores do banco de dados, e gerente de projeto;
- **Usuários finais**: servidores da UFMA e alunos.

3 LEVANTAMENTO DE REQUISITOS

É necessário reunir informações que são de extrema importância para atingir o objetivo de produzir um software de qualidade e que realiza as tarefas que o cliente necessita. Dessa forma inicia-se a coleta dos requisitos funcionais e não funcionais.

3.1 REQUISITOS FUNCIONAIS

O autor Wazlawick (pg. 22, 2011) explica que o conjunto de funções que o software deve realizar, ou seja, operações que venham a constituir a *funcionalidade* do sistema podem ser entendidas como requisitos funcionais. Assim na Tabela 1 podem ser visualizados os requisitos funcionais do sistema a ser desenvolvido.

Tabela 1 – Requisitos funcionais do sistema de acervo digital da UFMA

ID	Requisito	Descrição
[RF01]	Submeter cadastro ao sistema	Visitantes podem submeter um cadastro, que deve ser validado antes de confirmado.
[RF02]	Autenticar ao sistema	Usuários que possuam login e senha válidos podem ser autenticados no sistema para acessá-lo.
[RF03]	Indexar arquivos e extrair informações	Permitir que usuários do sistema enviem arquivos à um servidor central e retirar o texto do documento e armazená-lo para posteriores consultas.
[RF04]	Alocar arquivos em diretórios e associar um tipo.	Armazenar documentos de acordo com o diretório e com o tipo de documento que é definido pelo usuário no momento do envio do arquivo e é auxiliado com a inserção de *tags* que representam partes do documento.
[RF06]	Gerenciar Diretórios	Criação de diretórios acadêmicos (departamentos, setores, categorias, etc.) pelo moderador, de modo a realizar o agrupamento de documentos, organizar e auxiliar a busca dos mesmos.

[RF07]	Realizar busca de arquivos	Permitir que os próprios servidores e os alunos realizem a busca de arquivos através da definição de informações relevantes ao documento a ser pesquisado. O sistema fará uma busca geral pelo conteúdo do arquivo que pode ser por diretório acessado ou por termo ou expressão.
[RF08]	Gerenciamento de contas de usuários	Possibilita a manipulação (criar, excluir, listar, alterar) de contas dos usuários a fim de prover uma maior interação com o sistema através da submissão de trabalhos mediante a supervisão e aprovação do administrador.
[RF09]	Manipulação de arquivos	Na área principal ou depois de realizada busca de arquivos, o sistema deverá permitir que os usuários visualizem os documentos encontrados e também façam download desses arquivos.
[RF10]	Cadastrar no sistema	Permitir que visitantes submetam cadastro para acessar ao sistema.
[RF11]	Gerenciar contas de usuários	Permitir a manipulação (Criar, Atualizar, Excluir, Permissão de cadastro de visitante, Listar) das contas de usuários por parte de moderadores e administradores do sistema.
[RF12]	Gerenciar contas de moderadores	Permitir a manipulação (Criar, Atualizar, Excluir, Permissão de cadastro de visitante, Listar) das contas de usuários por parte de moderadores e administradores do sistema.
[RF13]	Link direto aos diretórios	Os diretórios devem possui um link direto. Ex.: considerando que o nome do sistema fique Acervo Digital <www.acervo.ufma.br/deinf > (neste caso diretório DEINF)
[RF14]	Associação de Coletâneas	Permitir associação de documentos em coletâneas. A ideia é associar por exemplo a monografia, com a apresentação e a ata de defesa em apenas um grupo de documentos. Outro exemplo seria associar um oficio de cobrança com a resposta demonstrando a execução completa do problema.

Fonte: arquivo do autor (2013)

3.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não-funcionais são aqueles que não estão especificamente relacionados com a funcionalidade do sistema. Eles impõem restrições no produto a ser desenvolvido e/ou no processo de desenvolvimento do sistema como também especificam restrições externas as quais o produto precisa atender.

Eles referem-se a questões como: segurança, confiabilidade, usabilidade, desempenho, entre outros.

3.2.1 Usabilidade

Conforme Soares e Koscianski (2007, págs. 213 a 214) a usabilidade representa o quão fácil é usar o produto. Esta é uma questão a ser tratada não somente no levantamento de requisitos, mas em fases posteriores como projeto de interface, implementação, validação e testes. A Tabela 2 mostra os requisitos não funcionais relacionados à usabilidade.

Tabela 2 – Requisitos não funcionais relacionados à usabilidade

ID	Descrição
[RNF01]	**Facilidade de Uso**: O usuário do sistema deve ter facilidade de uso do sistema. Para confirmação disso, será realizado um teste de usabilidade.
[RNF02]	**Interface WEB**: O usuário utilizará o sistema através de um *web browser*.
[RNF03]	**Página principal:** Visitantes podem visualizar algumas informações disponibilizadas pelo sistema.
[RNF04]	**Recomendação de publicações:** Recomendação das publicações mais recentes
[RNF04]	**Tamanho máximo de arquivo:** Será definido um tamanho máximo para envio de arquivo.

Fonte: arquivo do autor (2013)

3.2.2 Desempenho

O software que é desenvolvido deve possuir um bom desempenho, assim, foi definido um requisito não funcional que estabelece um critério para decidir entre o bom ou possível mal funcionamento do sistema que é apresentado na Tabela 3.

Tabela 3 – Requisitos não funcionais relacionados à confiabilidade

ID	Descrição
[RNF03]	A busca de documentos deverá ser realizada de forma que não ultrapasse um tempo de resposta mínimo.

Fonte: arquivo do autor (2013)

3.2.3 Confiabilidade

Outra questão relacionada aos requisitos não funcionais é a confiabilidade na qual Soares e Koscianski (2007, pg. 215) explicam que habitualmente um produto é confiável quando não falha. E como todo software está sujeito a ocorrência de falhas é necessário definir um parâmetro de aceitação do software desenvolvido. A tabela 4 mostra essa questão relacionada com o tempo disponível para produção do software.

Tabela 4 – Requisitos não funcionais relacionados à confiabilidade

ID	Descrição
[RNF04]	O tempo de desenvolvimento do software não deverá ultrapassar um semestre.

Fonte: arquivo do autor (2013)

3 GERENCIAMENTO DE REQUISITOS

O gerenciamento de requisitos se dá a partir de uma solicitação de mudança do cliente para a equipe de desenvolvimento.

3.1 GERENCIAMENTO DE MUDANÇAS DE REQUISITOS

Para que os requisitos possam ser atualizados ou alterados é necessário seguir um conjunto de atividades, listadas a seguir:

1. O cliente solicita uma mudança de requisito aos desenvolvedores;
2. O responsável da equipe por receber essa mudança sugerida é o Programador;
3. O Programador juntamente com o Gerente de Projeto analisarão a mudança e avaliarão o impacto da mesma no sistema;
4. O Gerente do Projeto juntamente com o cliente negociam a mudança pretendida;
5. Como resultado dessa negociação ocorrerá ou não a mudança solicitada.

4. CASOS DE USO

Dando continuidade ao processo de desenvolvimento do software, após o recolhimento dos requisitos, é possível organizá-los em casos de uso, uma especificação de uma sequência de interações entre um sistema e os agentes externos que utilizam este sistema sem mostrar o comportamento interno do sistema (BEZERRA, 2006, pg. 46). Dessa forma foi feito utilizando a ferramenta **astah** na qual o diagrama de casos de uso do sistema e é apresentado na Figura 1.

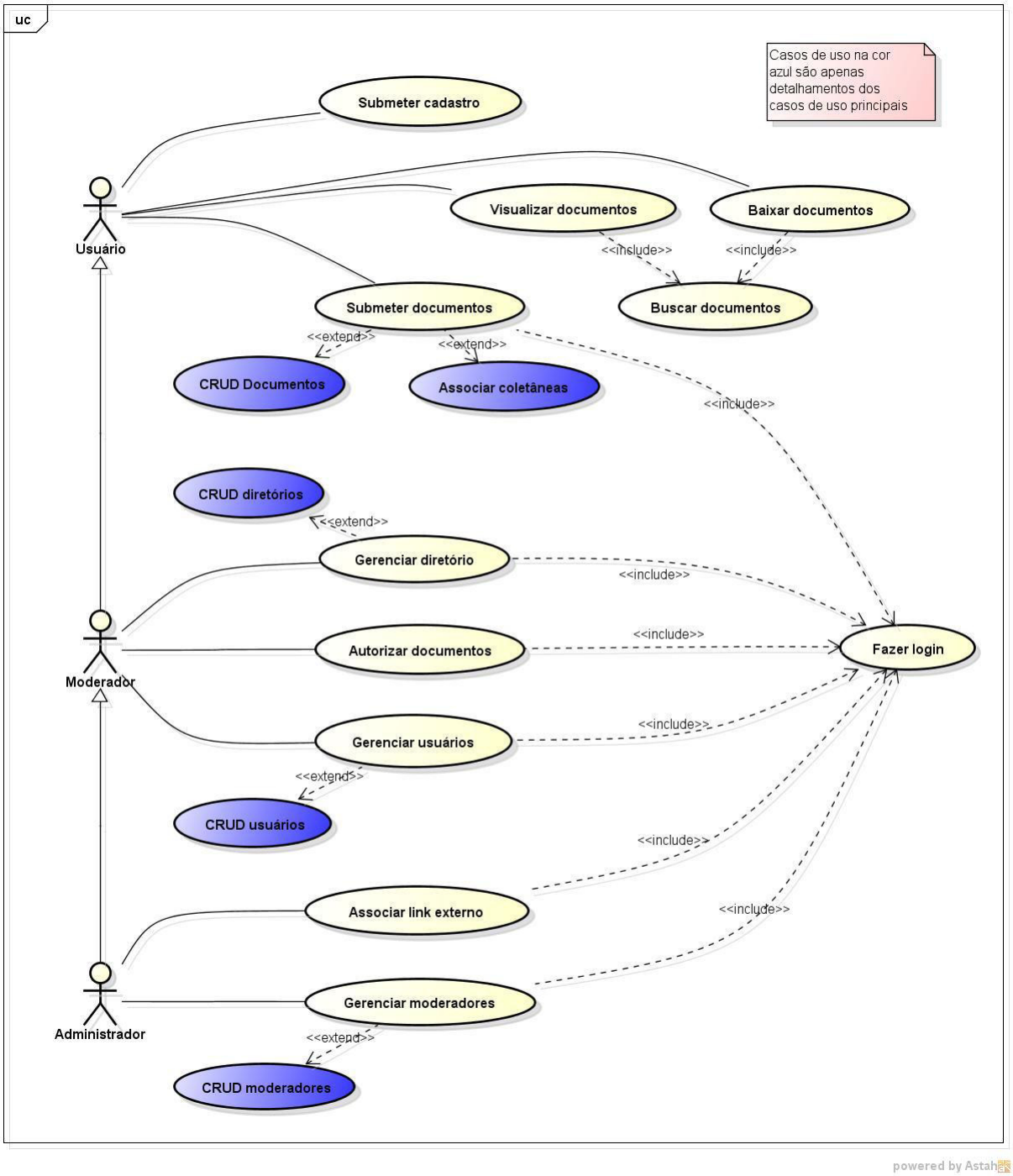
4.1 ATORES

1. **Usuário:** Este ator representa todos que querem utilizar o sistema esse ator pode visualizar ou baixar documentos, bem como submeter um formulário para obter mais privilégios como o de enviar arquivos.

2. **Moderador:** Este ator representam os coordenadores que terão todas as funcionalidades do ator anterior e ainda podem gerenciar as contas dos usuários, gerenciar diretórios e gerenciar os documentos enviados ao sistema.

3. **Administrador:** Este ator representa os chefes de departamento, é o maior nível de acesso do sistema, nele todas as funcionalidades do ator anterior são incluídas além da capacidade de gerenciar moderadores e excluir diretórios.

Figura 1 – Casos de uso do sistema de acervo digital da UFMA



Fonte: arquivo do autor (2013)

5 DETALHAMENTO DOS CASOS DE USO

Um caso de uso especifica o sistema num alto nível, mas existem situações particulares que podem acontecer que são chamadas de cenários, os quais devem ser detalhados, pois estes revelam informações relevantes do funcionamento do sistema através da iteração de atores.

5.1 SUBMETER CADASTRO

Ator: Usuário

Descrição: Ao visualizar a página principal do sistema o usuário poderá submeter um formulário com informações pessoais de forma a ter acesso a novas funcionalidades do sistema.

Pré-condição: O visitante deve redirecionar para a página raiz do sistema e ao visualizar a tela de login, selecionar o botão de “Cadastrar” e deve preencher corretamente os campos do formulário ao qual é redirecionado.

Pós-condição: O usuário recebe uma mensagem de confirmação, que os dados foram devidamente enviados e cadastrados no banco e um e-mail é enviado ao e-mail cadastrado pelo usuário para finalizar confirmação o cadastro.

Cenários

1. Submete cadastro - *Principal*
    a. O usuário seleciona o botão de cadastro do sistema;
    b. O usuário preenche as informações necessárias;
    c. O usuário envia o formulário;
    d. Uma mensagem de cadastro com sucesso é enviada.

2. O usuário esquece algum campo – *Exceção 1*
    a. No passo **b** do cenário **1** o usuário esquecer algum campo obrigatório, como nome, curso, e-mail, telefone, endereço;
    b. É exibida uma mensagem de atenção para que ele preencha os dados corretamente.

3. O usuário não recebe retorno do sistema – *Exceção 2*

a. Se por algum motivo o passo **d** do cenário **1** não ocorrer, algum problema do sistema gerenciador de banco de dados não conseguir incluir as informações, enviar uma mensagem de erro de sistema, e contatar os administradores do sistema.

5.2 SUBMETER DOCUMENTOS

Atores: Usuário, Moderador, Administrador.

Descrição: Compreende a ação de poder enviar ao servidor um arquivo no formato PDF, para que este seja armazenado e esteja disponível para busca.

Pré-condição: Realizado o caso de uso **"Fazer Login"**, o ator clicará na seção Documentos e selecionará a opção enviar documento, na qual uma nova página permitirá ao ator selecionar o(s) documento(s) no formato PDF.

Pós-condição: O sistema enviará uma mensagem de sucesso depois de enviado(s) todo(s) o(s) arquivo(s), e colocará os arquivos disponíveis para gerenciamento.

Cenários

1. Submete documento - *Principal*
    a. O ator seleciona a seção de Documentos e clica no botão de enviar documentos do sistema;
    b. Uma nova página é carregada onde é possível escolher documentos no formato PDF à serem enviados;
    c. O usuário envia o(s) arquivo(s);
    d. Uma mensagem de envio com sucesso é enviada.

2. O ator não indica nenhum arquivo para envio– *Exceção 1*
    a. No passo **b** do cenário **1** o ator deixa o formulário em branco e envia o mesmo;
    b. O sistema deve sinalizar uma mensagem de atenção que nenhum arquivo foi selecionado.

3. Tamanho de arquivo excedido – *Exceção 2*
    a. No passo **c** do cenário **1** o ator envia um arquivo acima do esperado;

b. O sistema deve sinalizar uma mensagem de erro que o arquivo enviado é superior ao tamanho permitido.

4. Ator não recebe retorno do sistema – *Exceção 3*
    a. No passo **d** do cenário **1 o** ator não recebe a mensagem do sistema;
    b. Caso não consiga efetivamente receber o(s) arquivo(s) o sistema deve sinalizar uma mensagem de erro.

5.3 VISUALIZAR DOCUMENTOS

Atores: Usuário, Moderador, Administrador.

Descrição: Representa a ação de visualizar o documento em sua totalidade.

Pré-condição: Realizado o caso de uso **"Buscar Documentos"** e clica na opção visualizar documento.

Pós-condição: Assim o sistema abrir deverá retornar um link para download do arquivo original e outro link que possibilita a reconstrução do arquivo em uma nova página web.

Cenários

1. Visualiza documento - *Principal*
    a. Buscar Documentos – Principal;
    b. Uma página contendo os resultados da pesquisa é apresentada para o usuário contendo um botão de visualização e de download ao lado do seu nome para cada documento encontrado;
    c. O ator clica sobre o ícone de visualização/download do documento desejado;
    d. Se clicado no botão de visualização uma nova página será aberta, sendo de responsabilidade de o sistema reconstruir o arquivo de acordo com as informações extraídas;
    e. O sistema deve sinalizar uma mensagem de documento gerado com sucesso;
    f. Se clicado no botão download, uma cópia do arquivo será enviada para o solicitante;
    g. O sistema deve sinalizar uma mensagem de documento baixado com sucesso.

2. Arquivo corrompido – *Exceção 1*

a. No passo **a** do cenário **1** o ator tenta acessar a página de visualização, sem ter feito login;
b. O sistema deve sinalizar uma mensagem de erro usuário não autenticado.

3. Reconstrução inviável – *Exceção 2*
    a. No passo **d** do cenário **1** o sistema não consegue por algum motivo recuperar as informações do documento para reconstrução;
    b. O sistema deve sinalizar uma mensagem de erro documento não pode ser reconstruído contatar administrador.

4. Arquivo corrompido – *Exceção 3*
    a. No passo **f** do cenário **1** o arquivo não é encontrado no sistema;
    b. O sistema deve sinalizar uma mensagem de erro arquivo não encontrado.

5.4 BAIXAR DOCUMENTOS

Atores: Usuário/Moderador/ Administrador

Descrição: Esse caso de uso inicia-se quando o ator, após realizar a busca, deseja fazer o download de um documento retornado pela mesma. Ele clica no link de download ao lado do nome do arquivo e o gerenciador de download do computador do cliente é iniciado.

Pré-condição: O usuário deve ter realizado o caso de uso “Buscar Documento” com êxito, estando, portanto, na página web com a lista de resultados e então clicar no link de download do documento.

Pós-condição: O documento escolhido é salvo no disco rígido do computador do cliente.

Cenários

1. Download de Documento - Principal
    a. Usuário realiza a busca de documentos.
    b. Após os resultados serem exibidos, este clica no link de download, ao lado do nome do documento escolhido.
    c. O gerenciador de download é aberto.

2. Visualizar o Documento ao Clicar em Download - *Exceção*
    a. Usuário realiza a busca de documentos.

b. Após os resultados serem exibidos, este clica no link de download, ao lado do nome do documento escolhido.

5.5 BUSCAR DOCUMENTOS

Atores: Usuário/Moderador/Administrador

Descrição: Esse caso de uso se inicia quando o usuário, moderador ou administrador do sistema deseja encontrar algum documento específico. Ele entra na página principal e coloca as informações que ele tem do documento no campo de busca e clica em “Buscar”.

Pré-condição: Deve-se entrar na página inicial e preencher o campo de busca.

Pós-condição: O sistema retorna uma lista de documentos correspondentes às informações que foram preenchidas no campo de busca.

Cenários

1. Lista de Documentos Encontrados - Principal
    a. Usuário/Moderador/Administrador entra na página principal e preenche o(s) campo(s) de busca.
    b. O sistema retorna uma lista de documentos correspondentes à busca.

2. Nenhum documento encontrado – *Exceção 1*
    a. Usuário/Moderador/Administrador entra na página principal e preenche o(s) campo(s) de busca.
    b. O sistema retorna uma mensagem indicando que nenhum documento foi encontrado.

5.6 FAZER LOGIN

Atores: Usuário/Moderador/Administrador.

Descrição: Esse caso de uso inicia-se quando o usuário, moderador ou administrador do sistema deseja ter acesso aos seus privilégios. Ele entra na interface web do sistema e preenche os campos de login e senha e clica em “Entrar”.

Pré-condição: Deve-se estar na página de login e preencher os campos login e senha corretamente.

Pós-condição: O ator é redirecionado para uma página onde ele tem acesso a todos os seus privilégios.

Cenários

1. Login Sucesso - Principal
    a. Ator informa seu login.
    b. Ator informa sua senha.
    c. Sistema verifica os dados, confirma que é cadastrado e redireciona o cliente para a página administrativa.

2. Nenhum usuário encontrado- *Exceção 1*
    a. Ator informa seu login.
    b. Ator informa sua senha.
    c. Sistema verifica os dados e confirma que não está cadastrado. Nesse caso, uma mensagem de falha de login aparece na tela.

3. Login Inválido – *Exceção 2*
    a. Ator não informa ou login ou senha.
    b. O sistema exibirá uma caixa de diálogo alertando o usuário à preencher os dois campos.

5.7 GERENCIAR USUÁRIOS

Atores: Moderador, Administrador.

Descrição: Manipular contas de usuários através de um CRUD de contas de usuário, ou seja, criar, listar, atualizar e deletar contas de usuário.

Pré-condições: Ter realizado o caso de uso **“Fazer Login”,** selecionar a seção de Gerenciamento, e clicar em “Contas de Usuários”.

Pós-condições: O ator será redirecionado a uma página que possibilitará a manipulação das contas de usuários.

Cenários:

1. O moderador/administrador escolhe a opção “Gerenciar módulo de usuários”;
2. O sistema exibe um menu com as seguintes opções:
   a. Novo usuário <<extends – Criar usuário>>;
   b. Editar usuário <<extends – Editar usuário>>;
   c. Excluir usuário <<extends – Excluir usuário>>;
   d. Listar usuários <<extends - Listar usuários>>;
   e. Retornar a seção “Gerenciamento” – finaliza o caso de uso;

5.8 GERENCIAR DIRETÓRIOS

Atores: Moderador, Administrador.

Descrição: Manipular diretórios através de um CRUD de diretórios, ou seja, criar, listar, atualizar e deletar diretórios.

Pré-condições: Ter realizado o caso de uso **“Fazer Login”,** selecionar a seção de Gerenciamento, e clicar em “Diretórios”.

Pós-condições: O ator será redirecionado a uma página que possibilitará a manipulação dos diretórios.

Cenários:

1. O moderador/administrador escolhe a opção “Gerenciar módulo de diretórios”;
2. O sistema exibe um menu com as seguintes opções de acordo com o perfil do ator:
   a. Novo diretório <<extends – Criar diretório>>;
   b. Editar diretório <<extends – Editar diretório>>;
   c. Excluir diretório <<extends – Excluir diretório>>;
   d. Listar diretórios <<extends - Listar documentos>>;
   e. Retornar a seção “Gerenciamento” – finaliza o caso de uso;

5.9 GERENCIAR DOCUMENTOS

Atores: Moderador, Administrador.

Descrição: Manipular documentos através de um CRUD de documentos, ou seja, criar, listar, atualizar e deletar documentos.

Pré-condições: Ter realizado o caso de uso **"Fazer Login",** selecionar a seção de Gerenciamento, e clicar em "Documentos".

Pós-condições: O ator será redirecionado a uma página que possibilitará a manipulação dos documentos.

Cenários:

1. O moderador/administrador escolhe a opção "Gerenciar módulo de documentos";
2. O sistema exibe um menu com as seguintes opções de acordo com o perfil do ator:
    a. Novo documento <<extends – Enviar documento>>;
    b. Editar documento <<extends – Editar documento>>;
    c. Excluir documento <<extends – Excluir documento>>;
    d. Listar documentos <<extends - Listar documentos>>;
    e. Retornar a seção "Gerenciamento" – finaliza o caso de uso;

5.10 GERENCIAR MODERADORES

Atores: Administrador.

Descrição: Manipular contas de moderadores através de um CRUD de contas de moderador, ou seja, criar, listar, atualizar e deletar contas de moderadores.

Pré-condições: Ter realizado o caso de uso **"Fazer Login",** selecionar a seção de Gerenciamento, e clicar em "Contas de Moderadores".

Pós-condições: O ator será redirecionado a uma página que possibilitará a manipulação das contas de moderadores.

Cenários:

1. O administrador escolhe a opção "Gerenciar módulo de moderadores";
2. O sistema exibe um menu com as seguintes opções:
    a. Novo moderador<<extends – Criar moderador>>;
    b. Editar moderador <<extends – Editar moderador>>;
    c. Excluir moderador <<extends – Excluir moderador>>;

d. Listar moderadore <<extends - Listar moderadores>>;
e. Retornar a seção “Gerenciamento” – finaliza o caso de uso;

5.11 EXCLUIR DIRETÓRIO

Atores: Moderador, Administrador

Descrição: Exluir diretórios que o Moderador/Administrador tem permissão. No caso do moderador, ele só poderá excluir diretórios que ele criou ou diretórios que estiverem sob sua hierarquia.

Pré-condições: Fazer o login como Administrador/Moderador, possuir diretórios sob sua hierarquia, ir em diretórios > gerenciar > Excluir Diretório.

Pós-condições: Após excluir o diretório o Administrador/Moderador, o sistema envia mensagem de sucesso caso não dê nenhum erro, e mostra a lista de diretórios.

Cenários:

1. Diretório Excluído com Sucesso - Principal
    a. O ator realiza o caso de uso **“Fazer Login”,** no sistema.
    b. O ator seleciona a seção Gerenciar e clica em Diretórios, no qual todos os diretórios serão listados.
    c. O ator escolhe a opção excluir do diretório desejado.
    d. O sistema envia uma mensagem de “Diretório excluído com sucesso!”.
2. Moderador não tem permissão para excluir diretório – *Exceção 1*
    a. Moderador/Administrador loga no sistema.
    b. Moderador/Administrador vai em Diretórios e escolhe o diretório que deseja excluir
    c. O Moderador/Administrador escolhe a opção excluir.
    d. O sistema envia a mensagem “Você não tem permissão para excluir esse diretório”.

5.12 AUTORIZAR DOCUMENTOS

Atores: Moderador, Administrador;

Descrição: Este caso de uso representa atividade permitir que documentos submetidos ao sistema sejam efetivamente indexados e disponibilizados para busca.

Pré-condições: Ter realizado o caso de uso, "**Fazer Login**", selecionar a seção de Gerenciamento e clicar na opção Autorizar Documentos.

Pós-condições: O ator será redirecionado a uma página que aonde são visíveis os documentos submetidos por usuários. Estes terão links associados de visualizar, permitir ou negar ao lado de cada documento listado.

Cenários:

1. Autorizar Documentos - Principal
    a. O ator seleciona a opção Autorizar Documentos;
    b. Uma nova página é gerada listando todos os arquivos que ainda não foram disponibilizados para busca, em que cada registro deve possuir um botão de visualizar, permitir e negar associado ao documento.
    c. Caso seja clicado no botão de visualizar, uma cópia do documento deve ser disponibilizada para o ator, para devida validação do arquivo.
    d. Caso seja clicado no botão de permitir o sistema deve iniciar o processo de indexação e extração de informações e enviar uma mensagem de arquivo indexado com sucesso de ser enviada ao usuário.
    e. Caso o botão de negar seja clicado, o arquivo enviado deve ser removido e uma mensagem de arquivo removido com sucesso deve ser enviada ao usuário.

5.13 ASSOCIAR LINK EXTERNO

Ator: Administrador.

Descrição: Consiste em disponibilizar uma url para acesso de um diretório.

Pré-condições: O administrador necessita realizar o caso de uso "**Fazer Login**", clicar na seção Gerenciamento, clicar na opção de Diretórios e selecionar na lista de diretórios o botão associar link externo.

Pós-condições: Uma nova página será gerada onde o administrador pode escrever o nome o qual o diretório pode ser acessado externamente.

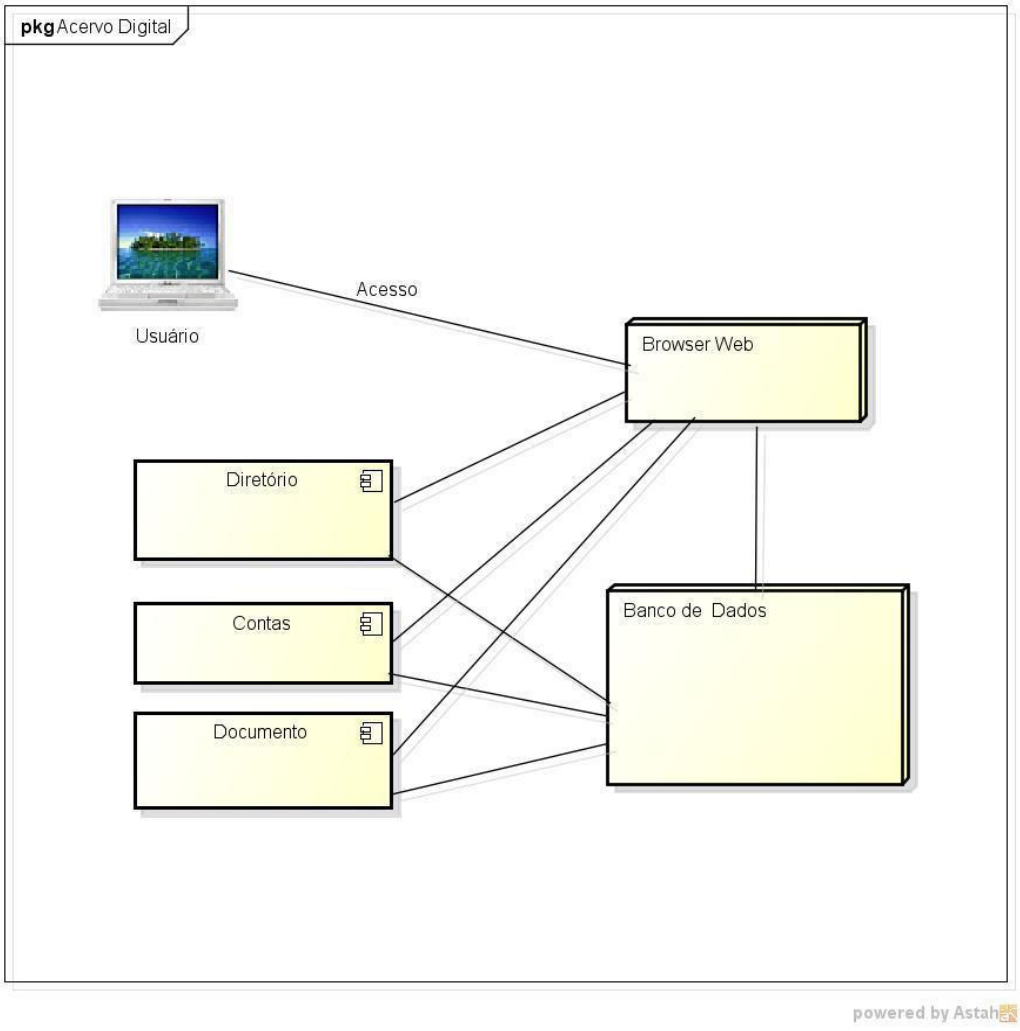
Cenários:

1. Associar link externo – Principal
    a. O administrador clica na seção de gerenciamento e clica na opção diretórios onde uma lista de todos os diretórios é mostrada;
    b. O administrador seleciona o botão associar link externo ao diretório desejado.
    c. O administrador digita no campo o nome o qual deseja q o diretório seja acessado.
    d. O sistema salva as alterações e envia uma mensagem de link externo criado com sucesso.

6 DIAGRAMA DE IMPLANTAÇÃO

A seguir na figura 4 apresentado o diagrama de sequência.

Figura 4 – Diagrama de implantação do sistema de acervo digital da UFMA



Fonte: arquivo do autor (2013)