UNIVERSIDADE FEDERAL DE MINAS GERAIS

FACULDADE DE EDUCAÇÃO

KARINA FLAVIANA RIBEIRO

ESTUDO DE CASO SOBRE O USO DA LINGUAGEM DE MODELAGEM UML NO PROCESSO DE LEVANTAMENTO DE REQUISITOS NO DESENVOLVIMENTO DE APLICAÇÕES DO LCC-UFMG

Belo Horizonte

2013

KARINA FLAVIANA RIBEIRO

ESTUDO DE CASO SOBRE O USO DA LINGUAGEM DE MODELAGEM UML NO PROCESSO DE LEVANTAMENTO DE REQUISITOS NO DESENVOLVIMENTO DE APLICAÇÕES DO LCC-UFMG

Trabalho apresentado ao curso de especialização Gestão de Instituições Federais de Educação Superior da Faculdade de Educação da Universidade Federal de Minas Gerais como requisito parcial à obtenção do título de especialista.

Linha de pesquisa: Gestão e Tecnologias

Orientador: Prof. Geraldo Magela Couto Oliveira

Belo Horizonte

2013

ESTUDO DE CASO SOBRE O USO DA LINGUAGEM DE MODELAGEM UML NO PROCESSO DE LEVANTAMENTO DE REQUISITOS NO DESENVOLVIMENTO DE APLICAÇÕES DO LCC-UFMG

Trabalho apresentado ao curso de especialização Gestão de Instituições Federais de Educação Superior da Faculdade de Educação da Universidade Federal de Minas Gerais como requisito parcial à obtenção do título de especialista.

Orientador: Prof. Geraldo Magela Couto Oliveira

Aprovado em _____ de _____ de 2013

BANCA EXAMINADORA

_____

Orientador Prof. Geraldo Magela Couto Oliveira – CEFET MG

_____

Profa. Rosilene Horta Tavares – UFMG

AGRADECIMENTOS

Agradeço a Deus, que é a luz que ilumina meu caminho, que nunca me abandona nas horas de prova e sofrimento.

A minha família, em especial meu filho, pela paciência e compreensão de ter uma mãe quase sempre ausente.

A bibliotecária chefe da Faculdade de Filosofia e Ciências Humanas, Vilma Carvalho, pelas orientações e informações utilizadas como base deste trabalho e pela constante disposição em me ajudar.

Ao meu orientador, Geraldo Magela, que com sua boa vontade e conhecimento assumiu a responsabilidade de me guiar nesta tarefa.

As pessoas que direta ou indiretamente me ajudaram neste trabalho.

A todos,

Muito obrigada!

Dedicatória

A minha mãe, cuja existência foi marcada pela alegria e pelo entusiasmo de viver! Pelos momentos inesquecíveis vividos ao seu lado. E que, onde estiver, sei que comemora cada vitória alcançada em minha vida. Para sempre te amarei!

“A parte mais árdua na construção de um software consiste exatamente em identificar “o que” construir. Nenhuma outra parte do trabalho compromete tanto o resultado do trabalho se elaborado de forma incorreta. Nenhuma outra parte oferece tanta dificuldade para efetuar correções posteriores.”.

Fred Brook (“No Silver Bullet”, 1986)

RESUMO

Este trabalho visa apresentar um estudo de caso sobre a utilização da linguagem de modelagem UML em uma alteração no Sistema Opus e na Biblioteca Digital de Teses e Dissertações da UFMG. Essa alteração surgiu em função de uma demanda da Biblioteca Universitária. Esta demanda consiste na necessidade de apoio e controle de situações em que o autor da produção solicita o embargado da mesma, de modo a não disponibilizar, por tempo pré-definido, o seu texto completo na Biblioteca Digital de Teses e Dissertações. Atualmente este controle é realizado de maneira manual pelas bibliotecárias, o que torna o processo sujeito a falhas. O objetivo principal do estudo é a introdução da linguagem de modelagem UML (*Unified Modeling Language*) no processo de Levantamento de Requisitos do Setor de Desenvolvimento de Sistemas do LCC-UFMG afim de que esta seja realizada, cada vez mais, de forma mais fácil, eficaz e eficiente. Para isso utilizou-se como estudo a implementação de um processo automatizado para o controle de produções que forem embargadas. A coleta de dados e informações foi feita, principalmente, através de entrevistas com usuários do sistema. Após a fase de levantamento de requisitos foi realizada a modelagem dos mesmos em UML. Além disso, este trabalho descreve, sucintamente, os conceitos de Engenharia de Software, Engenharia de Requisitos, suas etapas e os diagramas utilizados pela UML.

Palavras-chave: Engenharia de Software, Engenharia de Requisitos, Levantamento de Requisitos, UML.

LISTA DE FIGURAS

LISTA DE SIGLAS

BDTD – Biblioteca de Teses e Dissertações.

Dspace- *Institutional Digital Repository System.*

IBICT - Instituto Brasileiro de Informação em Ciência e Tecnologia.

JAD - *Joint Application Design.*

LCC - Laboratório de Computação Científica.

UML - *Unified Modeling Language.*

XML - *Extensible Markup Language.*

XSLT – *Extensible Stylesheet Language for Transformation.*

Sumário

1. Introdução

1.1. LCC – Laboratório de Computação Científica da UFMG

O LCC - Laboratório de Computação Científica da UFMG - foi criado em 1981. Até então o órgão encarregado de oferecer serviços computacionais para todos os fins era o CECOM, o Centro de Computação. Por iniciativa de um grupo de professores dos Departamentos de Física e de Ciência da Computação, foi proposta a criação do LCC, com a missão de oferecer apoio computacional às atividades de ensino e pesquisa.

Atualmente o LCC possui a seguinte estrutura interna (figura1):

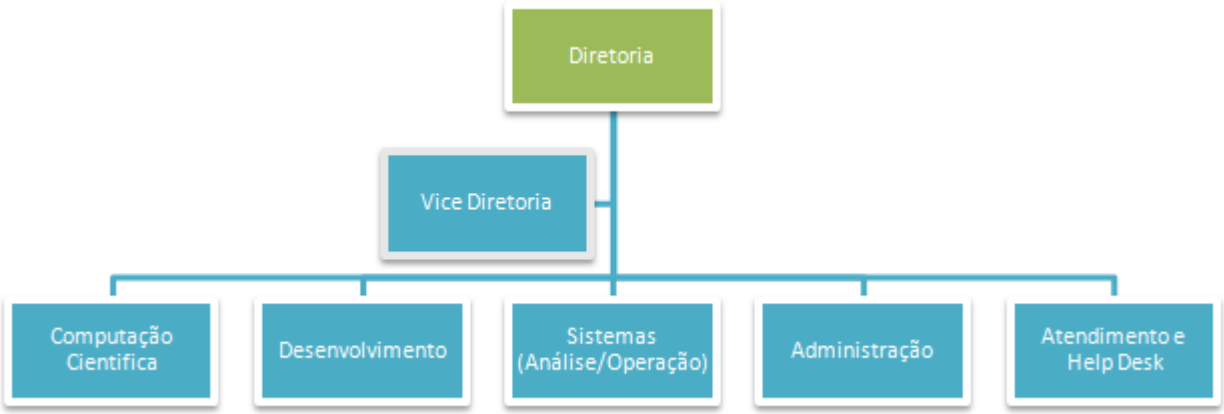

Figura 1- Estrutura interna do LCC. Fonte: http://www.lcc.ufmg.br/index.php/sobre-o-lcc/estrutura-interna, acessado em 11/02/2013.

O LCC é responsável pelo desenvolvimento e manutenção de algumas das aplicações utilizadas por vários setores da UFMG. Entre os serviços e sistemas que o LCC oferece pode-se citar: o Moodle, Opus, Sistema Perfil, SICS – Sistema de Bolsas da Iniciação Científica, Workflow de Química, Canções Brasileiras, Catálogo de Registro Sonoros Musicais Indígenas, Processamento de Questionários (QPL), Suporte ao minhaUFMG, Repositório de Objetos de Aprendizagem, BDTD – Biblioteca Digital de Teses e Dissertações.

A equipe é composta pelos seguintes profissionais:

- Oito analistas/desenvolvedores responsáveis pelo levantamento, desenvolvimento e manutenção dos sistemas e serviços.
- Um administrador de banco de dados, responsável por gerenciar e monitorar as atividades de banco de dados.
- Uma gerente, responsável pelo planejamento e controle da execução dos trabalhos de toda a equipe e as atividades relativas ao setor.

O LCC desenvolve tanto sistemas e/ou serviços que visam atender necessidades específicas de um setor da UFMG como sistemas e/ou serviços que serão de uso de várias unidades/departamentos em conjunto ou até mesmo de uso de toda a comunidade universitária. Cada aplicação é desenvolvida utilizando uma linguagem de programação e/ou banco de dados mais apropriada, afim de melhor atingir as expectativas do projeto. Levando em consideração o aspecto citado anteriormente, a equipe de profissionais não é homogênea. No grupo temos desenvolvedores com conhecimento em Java, Php, Webdesign, Lotusscript, Oracle e MySQL.

1.2. Contexto

O sucesso no desenvolvimento de um projeto e/ou software depende, fundamentalmente, do Levantamento de Requisitos e da sua gerência ao longo do processo, que são as bases que permitirão à equipe tirar conclusões sobre situações, problemas ou fenômenos e, assim, sugerir propostas que possam contribuir para uma solução adequada.

Em um Processo de Desenvolvimento de Software, a etapa de Levantamento de Requisitos é a primeira atividade importante, pois a partir da mesma o software será modelado e desenvolvido.

Percebendo a importância desta etapa, a indústria de software tem demonstrado crescente interesse em Engenharia de Requisitos. Busca-se, sobretudo, cada vez mais, entender o que se deseja construir antes de começar a fazê-lo já que, o tempo destinado ao entendimento do problema é um excelente investimento.

O contexto de um setor de desenvolvimento de sistemas de uma Instituição Pública não difere das organizações privadas de desenvolvimento de softwares. Os problemas enfrentados e as preocupações em garantir a qualidade final do produto também fazem parte do ambiente público.

A adoção de um processo definido e adaptado à realidade da Instituição Pública e do seu setor desenvolvedor é um fator decisivo para o sucesso dos projetos de softwares. Deste modo, é possível garantir sistemas confiáveis, que realmente atendam as necessidades dos usuários e que sejam desenvolvidos dentro de um cronograma e orçamentos previsíveis. Assim como o setor privado atentou para relevância desta etapa para o sucesso do projeto, a mesma também deve ser estudada e aprimorada nos setores públicos responsáveis por desenvolvimento de sistemas.

1.3. Problema

Os requisitos de um sistema representam todas as funcionalidades que o sistema englobará. Eles devem descrever o entendimento dos objetivos dos usuários e serem convertidos em ações dos sistemas, algo passível de execução.

Aos requisitos estão associados os principais problemas de desenvolvimento de software. Requisitos que não refletem as reais necessidades dos usuários, incompletos, inconsistentes, mudanças em requisitos já acordados, falta de comunicação de usuários e analistas, são as principais dificuldades encontradas ao desenvolver um software. Como consequência tem-se a geração de softwares com baixa qualidade, retrabalho por parte dos desenvolvedores, atrasos no cronograma, aumento nos custos e insatisfação dos usuários. Muitos desses erros poderiam ser evitados se as organizações dispusessem de um processo mais claro, objetivo e definido de Levantamento de Requisitos.

Segundo Peters & Wiltold (2001), o tempo gasto em detecção e correção de erros, quando percebidos na fase de requisitos, é mínimo. A correção de erros durante a fase de desenvolvimento é quase 100 (cem) vezes mais trabalhosa e custosa se comparada à fase inicial do projeto. A figura 2 representa o crescimento dos custos para correções conforme as fases:

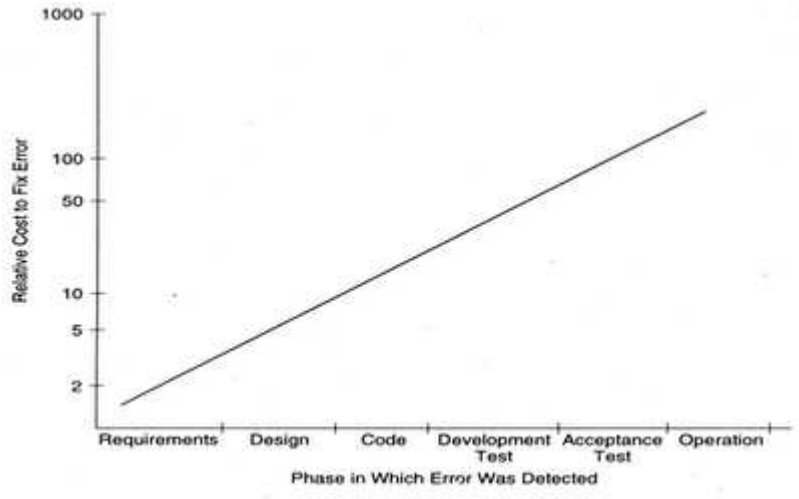

Figura 2 - Gráfico representativo de custos para correção de erros por fase. Fonte: http://www.cs.colorado.edu/~kena/classes/5828/s07/lectures/04/index.html#%2812%29, acessado em 28/02/2013.

Em seu trabalho, Filho (2003) apresenta dados que justificam a melhoria e o investimento nos processos de software, principalmente nas etapas que envolvem requisitos, entre os quais: captar um requisito correto é de 50 a 200 vezes mais barato do que corrigi-lo durante a implementação ou operação. A engenharia e a gestão de requisitos estão entre as práticas de maior retorno de investimento. Retirando a fase de manutenção, refazer defeitos de requisitos, desenho e código consomem 40% a 50% do custo total dos projetos. Portanto, a garantia da qualidade se paga rapidamente, à medida que diminui a necessidade de refazer. Cada hora gasta em prevenção de defeitos representa menos 3 a 10 horas de correção de defeitos. Entre as atividades de qualidade, as que estão relacionadas com a prevenção de defeitos são mais eficazes do que aquelas que focalizam a correção.

Porém, muitos usuários e alguns desenvolvedores, no entanto, não percebem a importância da elicitação de requisitos, por crerem que a mesma representa um gasto supérfluo.

Em se tratando de uma Instituição Pública de Ensino, esta atividade torna-se mais complexa e trabalhosa. Em virtudes de criações e/ou modificações em leis, decretos, ou até mesmo regimento interno, os sistemas precisam ser adaptados a nova realidade. O profissional responsável por este levantamento de informações precisa ter um conhecimento da Instituição como um todo, de modo a prever possíveis interfaces com outros sistemas já existentes ou em desenvolvimento, assim como impactos em outros departamentos ou unidades acadêmicas.

1.4. Justificativa

Dada à importância que as organizações tem dado a etapa de Levantamento de Requisitos, devido às vantagens trazidas pela mesma, é necessário um estudo sobre como incorporar melhorias a essa etapa nos setores de desenvolvimento de softwares das Instituições Públicas de Ensino.

Atualmente a metodologia utilizada pela equipe de desenvolvimento do LCC-UFMG para Levantamento de Requisitos consiste em reuniões com usuários chaves. Após esses encontros é desenvolvido um documento denominado Plano de Projeto. Este documento tem como principais objetivos descrever a situação atual e propor uma situação para a resolução do problema. Este documento é enviado aos usuários chaves para aprovação. Após a aprovação, é iniciado o desenvolvimento do sistema. Para descrever as alterações e/ou implementações da etapa de desenvolvimento, a equipe utiliza um documento denominado Especificação Técnica. As características principais deste documento é descrever, para modificações em sistemas existentes onde e o que foi alterado, e para novos sistemas o que foi implementado. Essas alterações ou criações são referentes a bibliotecas, classes, formulários, agentes. Não se utiliza uma linguagem de modelagem, casos de uso, ou documentação específica para requisitos. De modo geral, o documento de Especificação Técnica é gerado durante a Etapa de Desenvolvimento (codificação), sendo o mesmo modificado durante toda a duração desta etapa.

Durante e após o desenvolvimento do sistema são realizados testes pela equipe desenvolvedora. Terminada esta fase, são realizados testes pelos usuários. Caso os usuários aprovem o que foi implementado, o projeto é colocado em produção para uso dos mesmos.

A equipe de desenvolvimento é considerada pequena diante da demanda de projetos, sendo assim os profissionais exercem múltiplos papéis. Na maioria dos projetos, o analista é também o programador do sistema, ou seja, aquele que participa da fase de desenvolvimento de requisitos e da programação, incluindo desenvolvimento, testes e implementação. Este papel de analista/programador requer que o profissional esteja constantemente atualizado.

Novos projetos sempre fazem parte do cenário da UFMG. Em sua maioria, estes projetos impactam diretamente nas atividades desenvolvidas pelo LCC-UFMG, pois envolvem a unificação dos bancos de dados, reestruturação de sistemas existentes e desenvolvimento de novos sistemas. O setor de desenvolvimento de aplicações do LCC-UFMG deve conter em seu quadro de recursos humanos, profissionais capazes de identificar e definir estratégias. Na etapa de Levantamento de Requisitos estas habilidades e conhecimentos são fundamentais para o sucesso dos projetos.

A análise do uso da linguagem de modelagem UML (*Unified Modeling Language*) será proposta para uso no setor de desenvolvimento do LCC-UFMG. De acordo com o descrito anteriormente, as aplicações são desenvolvidas em plataformas diversas, conforme suas necessidades, e por consequência a equipe é composta por profissionais com conhecimentos em diversas linguagens de programação. Apesar de todos utilizarem os mesmos modelos de documentos para seus projetos, não existem diagramas padronizados. Cada analista/desenvolvedor faz sua documentação conforme seus projetos.

A escolha da etapa de Levantamento de Requisitos para estudo, além da sua importância, deve-se ao fato que é uma etapa que independe da linguagem de programação na qual será implementado o projeto. Assim, será possível que a maioria dos participantes da equipe de desenvolvimento possa utilizar o resultado deste estudo.

O uso de uma linguagem de modelagem permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados.

Espera-se que este estudo possa contribuir para as atividades do setor, objetivando a melhoria dos processos já existentes, de desenvolvimento e manutenção de softwares, como agilidade, confiabilidade e qualidade.

1.5. Objetivos e Metas

1.5.1. Objetivos Principais

Este trabalho visa apresentar um estudo sobre a introdução da linguagem de modelagem UML no processo de Levantamento de Requisitos do Setor de Desenvolvimento de Sistemas do LCC-UFMG.

1.5.2. Objetivos Específicos

- Pesquisar e descrever a etapa de Levantamento de Requisitos e benefícios;
- Pesquisar e descrever a linguagem de modelagem UML, que tem como foco principal os diagramas para o Levantamento de Requisitos;
- Utilizar UML no Levantamento de Requisitos de uma aplicação do LCC-UFMG. Nesse estudo, será realizado um Levantamento de Requisitos para a implementação de uma nova funcionalidade de uma aplicação já existente e utilizada pela UFMG. Não será levada em consideração a plataforma em que está desenvolvida esta aplicação;
- Demonstrar a importância do Levantamento de Requisitos para garantir a qualidade do produto a ser desenvolvido e o uso da UML para atingir tal objetivo.

2. Referencial Teórico

Para melhor entendimento deste trabalho, é necessária a compreensão de termos que serão citados durante o seu desenvolvimento. Este capítulo visa conceituar:

1 - Engenharia de Software;

2 - Engenharia de Requisitos;

3 - UML.

Os conceitos serão abordados de maneira simples (superficial e de fácil entendimento), proporcionando ao leitor um necessário embasamento teórico.

2.1. Engenharia de Software

A Engenharia de Software pode ser entendida como uma disciplina que envolve o desenvolvimento e manutenção de software, objetivando através de métodos definidos, a construção viável de software de qualidade. Para tanto, a Engenharia de Software requer atividades que sejam realizadas de maneira planejada, com equipe capacitada, estimando prazos e custos, além de um gerenciamento em todas as etapas e contando com a utilização de teorias, técnicas, métodos, modelos e ferramentas adequadas.

Essa abordagem da engenharia caracteriza-se por seguir processos, que são formas de realizar uma atividade seguindo padrões e normas. De acordo com Peters e Wiltold (2001), um processo de software pode ser entendido como uma sequencia de atividades que produzem uma variedade de documentos, culminando em um programa satisfatório e executável.

Enquanto um processo de software define atividades a serem realizadas, o modelo de processo de software define a sequencia na qual estas atividades serão desenvolvidas, bem como o pessoal envolvido e os tipos de artefatos gerados por cada atividade.

Segundo Pressman (1995) um processo de desenvolvimento de software contém três fases genéricas, independente do modelo de processo escolhido, sendo estes: definição, desenvolvimento e manutenção.

De modo geral, em cada fase de um processo de software são executadas as atividades básicas para que sejam atingidos os objetivos propostos. Estas atividades constituem um conjunto mínimo para se obter um produto de software.

Em seu trabalho, Pressman (1995) cita as seguintes atividades:

- Análise e Engenharia de Sistema: envolve uma visão geral: interface com outros sistemas, hardware, pessoas e banco de dados.
- Análise de Requisitos: levantamento das necessidades do software a ser implementado. Os requisitos são documentados e revistos com o cliente.

- Projeto: neste passo concentram-se quatro atributos distintos do programa: estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização da interface.
- Codificação: a implementação do sistema em si, em uma linguagem de programação.
- Testes: Validação do que foi codificado. Realização de testes para verificar presença de erros e comportamentos inadequados do sistema, além de verificar interações entre diferentes módulos e, quando aplicável, com sistemas externos.
- Manutenção e Evolução: o software poderá sofrer mudanças depois que for entregue. Essas mudanças podem ser:
    - Corretivas: A manutenção corretiva muda o software para corrigir defeitos.
    - Adaptativas: A manutenção adaptativa muda o software para acomodar mudanças em seu ambiente.
    - De Melhoramento Funcional: a medida que o software é usado, o usuário reconhecerá funções adicionais que oferecerão benefícios.

Nesta fase, o software em geral entra em um ciclo iterativo que abrange todas as fases anteriores.

A definição e utilização de um processo de software são fundamentais para sucesso no desenvolvimento de sistemas. Em seu trabalho, Filho (2003) faz ainda uma importante observação, ao afirmar que processos rigorosamente definidos, mas não alinhados com os objetivos das organizações, são impedimentos burocráticos, e não fatores de produção. Sendo assim, o modelo a ser seguido pela empresa/instituição vai depender das suas reais necessidades, condizente com a realidade na qual está inserida.

Para tornar uma organização mais madura e capacitada é preciso, na verdade, melhorar a qualidade de processos de software, de forma que possibilitem a entrega de produtos com qualidade, por um menor custo e em prazos mais curtos.

2.2. Engenharia de Requisitos

Um requisito pode ser entendido como a descrição de um serviço, uma funcionalidade ou uma limitação que seja essencial ou desejável ao funcionamento do sistema.

Para o IEEE[1] um requisito é definido como:

1. Uma condição ou capacidade necessitada por um usuário para resolver um problema ou atingir um objetivo;
2. Uma condição ou capacidade que deve ser atingida ou possuída por um sistema ou componente de sistema para satisfazer um contrato, padrão, especificação, ou outro documento de formalidade;
3. Uma representação documentada de uma condição ou capacidade como em (1) ou (2).

Complementando a definição do IEEE, Peters e Wiltold (2001), descrevem Requisito de Software como:

> ... uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamentos e atributos. Em suma, um requisito de software fornece uma estrutura básica para o desenvolvimento de um produto de software.

De modo geral, os requisitos classificam-se em:

- Requisitos Funcionais: aqueles que definem parte da funcionalidade do sistema, ou seja, descrevem as funções que o software deve executar. Por exemplo, um software de gerenciamento de uma escola deve permitir o cadastro de alunos, professores, cursos, turmas, gerações de relatórios, dentre outras funcionalidades.
- Requisitos Não Funcionais: aqueles que dizem respeito a restrições, aspectos de desempenho, interfaces com o usuário, confiabilidade, segurança, mantenabilidade, portabilidade, padrões. Por exemplo, o sistema de matrícula de alunos deve ser implementado de forma a ser acessado, sem erro, na maioria dos browsers disponíveis no mercado ou o tempo de retorno de uma consulta ao banco de dados não pode ser superior a 3 segundos.

[1] Institute of Electrical and Electronics Engineers (http://www.ieee.org/)

- Requisitos de domínio: estes requisitos podem ser funcionais ou não e refletem o domínio do sistema. Podem ser requisitos funcionais novos, restrições sobre requisitos existentes ou computações específicas. Por exemplo, um aluno somente pode se matricular em uma determinada disciplina se tiver sido aprovado em outra disciplina considerada pré-requisito.

Os requisitos são a base para a definição da arquitetura do sistema, para a implementação propriamente dita, para a geração dos casos de testes e para a validação do sistema junto ao usuário. A especificação de requisitos é um grande desafio para o projeto de desenvolvimento de um software. Nesse contexto, a Engenharia de Requisitos auxilia a delimitar o escopo do projeto e estabelecer a base comum de comunicação para todas as atividades envolvidas no projeto de software.

A Engenharia de Requisitos é definida por Filho (2003) como um conjunto de técnicas de levantamento, documentação e análise dos requisitos dos produtos de software. Ainda de acordo com Peters e Wiltold (2001), a Engenharia de Requisitos pode ser definida em função das suas atividades principais: entendimento dos problemas, determinação de soluções e especificação de uma solução que seja testável, compreensível, passível de manutenção e que satisfaça as diretrizes de qualidade do projeto.

De acordo com Peters e Wiltold (2001), o foco da Engenharia de Requisitos é a definição e a descrição do que um sistema de software pode satisfazer aos requisitos informais fornecidos por um relatório de necessidades. Para ele, o pensamento na análise de requisitos deve voltar-se, principalmente, para o problema e não para a solução.

Segundo Kotonya e Sommerville (1998), os Processos de Engenharia de Requisitos são um conjunto estruturado de atividades que são seguidas para derivar, validar e manter o documento de requisitos do sistema. Essas atividades de processo incluem:

- ✓ **Levantamento de Requisitos**

A elicitação de requisitos consiste no levantamento e identificação das necessidades dos usuários, quais são os objetivos do sistema ou produto, o que deve ser acompanhado, como o sistema ou produto se encaixa no contexto das necessidades do negócio e como será a utilização do sistema ou produto.

- ✓ **Análise e Negociação de Requisitos**

Uma compreensão completa dos requisitos é fundamental para o desenvolvimento do software, já que a atividade de análise consiste em categorizar e organizar os requisitos com base nos documentos gerados na etapa anterior. Essa atividade verifica a consistência, a omissão e a ambiguidade de cada requisito. Com base nas necessidades dos clientes/usuários, os requisitos são priorizados e organizados em subconjuntos relacionados, de modo que cada requisito tenha relacionamento com os demais.

Esta etapa também engloba a negociação, junto aos clientes/usuários, do tamanho do escopo do software, além da resolução e definição para requisitos ambíguos ou conflitantes.

- ✓ **Especificação de Requisitos**

De acordo com Filho (2003), a Especificação de Requisitos visa descrever de forma detalhada um conjunto dos requisitos que devem ser satisfeitos por uma solução implementável para o problema. Nesta definição, ainda acrescenta que a Especificação dos Requisitos é o documento oficial de descrição dos requisitos de um projeto de software, ou seja, pode se referir a um produto indivisível de software ou a um conjunto de componentes de software que formam um produto quando usados em conjunto.

Para Peters e Wiltold (2001), a Especificação de Requisitos é a descrição de um produto de software, programa ou conjunto que executa uma série de funções no ambiente de destino. Conforme Filho (2003), a Especificação de Requisitos deve incluir as seguintes características:

- Funcionalidade: o que o software deve fazer;
- Interfaces externas: como o software interage com as pessoas, hardware, outros sistemas e com outros produtos;
- Desempenho: diz respeito à velocidade de processamento, tempo de resposta e outros parâmetros de desempenho requeridos pela aplicação;
- Outros atributos: portabilidade, manutenibilidade e confiabilidade;
- Restrições impostas pela aplicação: existem padrões e outros limites a serem estabelecidos, como linguagem de programação, banco de dados, limites de recursos, etc.

Peters e Wiltold (2001) completam afirmando que o grau de compreensibilidade, precisão e rigor da descrição fornecida por um documento de requisitos de software tende a ser diretamente proporcional ao grau de qualidade do produto resultante.

✓ **Modelagem do Sistema**

Nesta atividade são criados modelos a fim de facilitar o entendimento por parte dos analistas e desenvolvedores do sistema a ser criado. Esses modelos visam descrever, em forma de notação gráfica e textual, a função, comportamento e outras características do sistema.

✓ **Validação de Requisitos**

Pressman (1995) afirma que a revisão da Especificação de Requisitos avalia a exatidão da definição contida no documento e Peters e Wiltold (2001), acrescentam que, a validação deve identificar relações para consistência, correção, completude e acurácia dos requisitos levantados, garantindo que a atenção seja focada em aspectos específicos dos requisitos. Esta atividade engloba uma revisão de todos os requisitos levantados e negociados, assim como uma prototipagem e validação de modelos e teste de requisitos. De modo geral, após analisar e documentar os requisitos levantados, o analista realiza uma reunião com os *Stakeholders*[2] para validação dos mesmos. A revisão não deve incidir sobre os requisitos individualmente, mas sim nas relações dos requisitos.

[2] Stakeholders é qualquer pessoa ou organização que tenha interesse, ou seja, afetado pelo projeto, como por exemplo, analistas, gerentes, usuários, clientes, patrocinador.

Para auxiliar o processo de validação de requisitos, técnicas podem ser utilizadas, como exemplo pode-se citar:

- Análise manual sistemática dos requisitos;
- Uso de modelo executável do sistema para avaliar requisitos;
- Desenvolvimento de testes específicos para os requisitos a fim de avaliá-los.

✓ **Gerenciamento de Requisitos**

Segundo Filho (2003), a Gerência de Requisitos pode ser definida como a disciplina da Engenharia de Software que procura manter sob controle o conjunto dos requisitos de um produto, mesmo diante de algumas alterações inevitáveis.

Segundo Thayer e Dorfman (1997), a Gerência de Requisitos tem como objetivo principal controlar a evolução dos requisitos seja por constatação de novas necessidades, seja por constatação de deficiências nos requisitos registrados até o momento.

A Gerência de Requisitos possui uma abordagem sistemática para levantar, documentar, organizar e rastrear mudanças em requisitos.

A Gerência de Requisitos visa:

- Manter registro do status de cada requisito em relação ao processo de desenvolvimento;
- Controlar as modificações aceitas e razões associadas ao processo de desenvolvimento, tendo como referência a baseline[3] de requisitos;
- Estabelecer uma visão comum entre o cliente e a equipe de projeto em relação aos requisitos que serão atendidos pelo software;
- Documentar e controlar os requisitos alocados numa baseline para uso gerencial e da engenharia de software;

[3] Baseline pode ser entendida como um conjunto de especificações ou produtos de trabalho que foram formalmente revisados e acordados. A baseline servirá de base para desenvolvimento posterior. Somente mudanças autorizadas podem ser efetuadas através dos procedimentos de controle de mudanças.

- Manter planos, artefatos e atividades de software consistentes com os requisitos alocados.

As atividades da Gerência de Requisitos não terminam com a entrega da Especificação de Requisitos, mas continuam durante todo o ciclo de vida do software.

De acordo com Leite (2003), a implementação de boas práticas de Gerência de Requisitos de software constitui uma das prioridades na implantação de melhoria do Processo de Software.

2.2.1. Modelo de Processo de Engenharia de Requisitos em Espiral

Conforme Sommerville (2007), as atividades relacionadas à obtenção, documentação e verificação de requisitos visam criar e manter o documento de requisitos. O processo usado na Engenharia de Requisitos varia bastante, dependendo do domínio da aplicação, das pessoas envolvidas e dos interesses da organização.

Sommerville (2007) apresenta uma alternativa do Processo de Engenharia de Requisitos, denominado Modelo em Espiral. Este modelo mostra o processo como uma atividade de três estágios, na qual as atividades estão organizadas como um processo interativo em espiral. No modelo em espiral os requisitos são desenvolvidos para diferentes níveis de detalhes. No início do processo, a maior parte do tempo e esforço será direcionada no entendimento de requisitos de alto nível de negócios, requisitos não funcionais e requisitos de usuários. Próximo ao fim do processo, nas partes mais externas da espiral, um esforço maior será empregado à engenharia de requisitos e modelagem de sistemas. Neste modelo, conforme a figura 3, os números de inteirações podem variar.

Figura 3 - Modelo em Espiral dos Processos de Engenharia de Requisitos. Fonte da imagem: Ian Sommerville, Engenharia de Software, 8 edição, capítulo 7.

Serão utilizados, para desenvolvimento deste trabalho, os processos de levantamento, análise e negociação, especificação e modelagem. Como produto final, espera-se a geração do documento de Especificação de Requisitos contendo os processos citados acima, bem como a inclusão da modelagem em UML de uma demanda de trabalho do LCC.

2.2.2. Levantamento de Requisitos

Os requisitos frequentemente são expressos em linguagem natural, portanto essa é a atividade onde se obtém maior interatividade com o cliente. A comunicação contínua com o usuário, aliada a técnicas e métodos, contribui para o efetivo levantamento dos requisitos. Por ser uma atividade complexa, existem dificuldades para a sua realização.

Essas dificuldades estão associadas a problemas de definição de escopo, compreensão e volatilidade. Os problemas de escopo estão associados aos limites do sistema, definições incompletas que podem confundir os objetivos gerais do sistema. Os problemas de compreensão envolvem a comunicação entre desenvolvedor e usuários. Muitas vezes os usuários não estão certos de suas necessidades ou não entendem (por não terem conhecimento técnico) limites de ambientes computacionais tais como de linguagem de programação, hardware, plataforma. Aqui também se enquadra um problema grave, quando usuários especificam requisitos que são conflitantes com interesses de outros usuários. Além disso, muitos omitem informações importantes, por julgá-las obvias. Já os problemas de volatilidade envolvem mudanças nos requisitos já acordados.

Nessa atividade os profissionais de sistemas trabalham em conjunto com os usuários finais, aqueles que irão utilizar a aplicação. O objetivo é aprender sobre o domínio do sistema, quais as funcionalidades que o sistema deve englobar, o desempenho esperado, restrições de ambiente, dentre outros.

A figura 4 apresenta um modelo de elicitação e análise de requisitos proposta por Sommerville (2007).

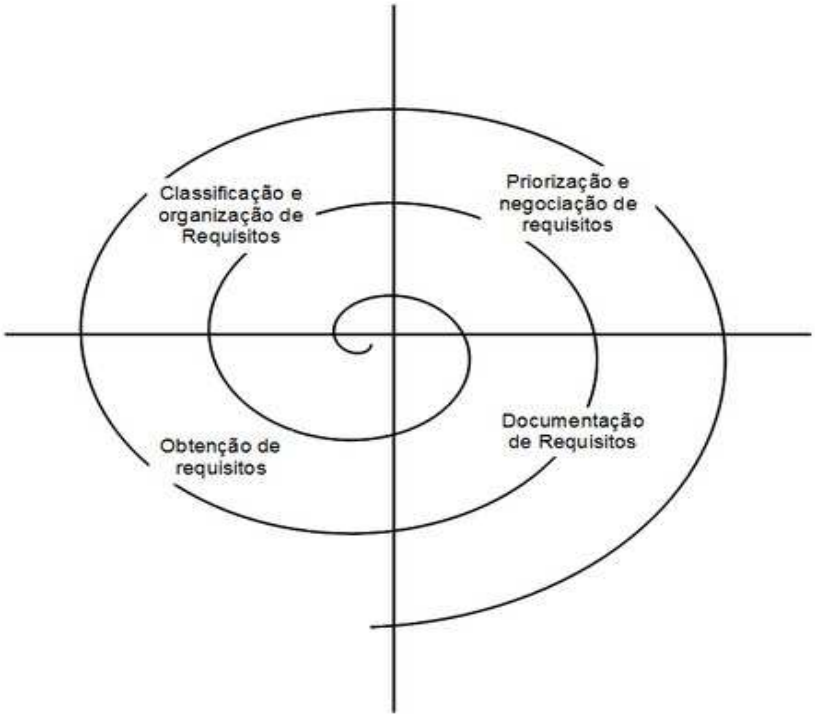

Figura 4- Processo de elicitação e análise de requisitos. . Fonte da imagem: http://progridbb.wikidot.com/engenhariadesoftware acessado em 24/03/2013

Como visto na figura acima e descrito por Sommerville (2007), são quatro as atividades de processo:

1 - Obtenção de Requisitos: consiste no processo que reúne informações sobre o sistema proposto e os existentes para obter os requisitos de usuários e sistemas.

2 - Classificação e Organização de Requisitos: agrupa requisitos relacionados e organiza-os em conjuntos coerentes.

3 - Priorização e Negociação de Requisitos: priorização de requisitos e resolução de conflitos de requisitos.

4 - Documentação de Requisitos: Os requisitos são documentados e colocados na próxima volta da espiral.

Em todo desenvolvimento de software, um aspecto fundamental é a captura dos requisitos dos usuários. Para apoiar este trabalho, diversas técnicas podem ser utilizadas. Abaixo, algumas:

- Amostragem: é o processo de seleção sistemática de elementos representativos de uma população.
- Investigação: consiste na analise de documentos. Através de investigação, podemos obter mais facilmente informações, tais como tipos de documentos e problemas associados, informação financeira e contextos da organização.
- Entrevista: levantamento de informações através de uma conversa direcionada com um propósito específico, que utiliza um formato “pergunta-resposta”.
- Cenários: são simulações de como um sistema poderá ser usado.
- Etnografia: consiste na observação de modo a compreender os requisitos sociais e organizacionais da empresa.
- *Workshops*: utiliza o Brainstorming, que é uma técnica para geração de idéias. Fazem uma ou várias reuniões que permitem que as pessoas sugiram e explorem idéias.
- Observação: observar o comportamento e o ambiente do usuário é uma forma de levantar informações não levantadas por outras técnicas.
- Prototipagem: tem por objetivo explorar aspectos críticos dos requisitos de um produto, implementando de forma rápida um pequeno subconjunto de funcionalidades deste produto.
- JAD (*Joint Application Design*): é uma técnica para promover cooperação, entendimento e trabalho em grupo entre os usuários desenvolvedores.

Todos os métodos de levantamento de requisitos possuem vantagens e desvantagens e nenhum pode ser considerado completo. Muitas variáveis podem decidir na escolha do método apropriado para levantamento: complexidade, perfil dos usuários, situações de hierarquia e política na empresa, nível de qualificação dos profissionais, dentre outros. O analista deve ter um perfil adequado para decidir qual(is) técnica(s) serão utilizadas. A combinação de técnicas pode ser interessante, tendo que vista que uma técnica pode ter características complementares à outra.

2.2.3. Análise e Negociação de Requisitos

Após a identificação dos requisitos do sistema, segue-se à etapa de análise dos requisitos e negociação. A análise e negociação de requisitos é processo complexo, pois neste momento é preciso definir as fronteiras do sistema, verificar se há requisitos conflitantes ou que não condizem com a necessidade da organização naquele instante.

Segundo Júnior (2010), algumas das atividades podem ser envolvidas na análise de requisitos:

- Classificação: agrupamento de requisitos em "módulos" para facilitar a visão global do funcionamento pretendido para o sistema;
- Resolução de conflitos: dada a multiplicidade e diversidade de papéis das partes interessadas envolvidas na captura e análise de requisitos, é inevitável a existência de conflitos nos requisitos identificados; é importante resolver estes conflitos o mais breve possível;
- Priorização: consiste na atribuição de uma "prioridade" a cada requisito (por exemplo elevada/média/baixa); obviamente, este pode ser um fator gerador de conflitos;
- Confirmação: é confirmada com as partes interessadas a completude dos requisitos, sua consistência e validade (de acordo com o que se pretende do sistema).

Os processos de elicitação, análise e negociação ocorrem de forma intercalada. Para se chegar a um documento de requisitos que satisfaça aos diversos usuários do sistema, normalmente cada uma das atividades deve ser realizada várias vezes. A meta é estabelecer um consenso sobre os requisitos que estão sendo definidos.

2.2.4. Especificação de Requisitos

A Especificação emerge dos componentes de análise de problemas. Peters & Wiltold (2001) afirmam que, ao escrever uma Especificação de Requisitos, o profissional irá lidar com cinco questões básicas, descritas a seguir:

1. Funcionalidade: O que o software pretende fazer, seus objetivos, funções e estados.
2. Interfaces externas. Como o software interage com as pessoas, com o hardware do sistema, outros hardwares e outros softwares.
3. Desempenho: performance, velocidade, disponibilidade, tempo de resposta, tempo de restauração, recuperação das várias funções do software.
4. Atributos: portabilidade, conectividade, manutenção, segurança, qualidade, estabilidade, rastreabilidade.
5. Restrições: limites traçados estabelecidos na implementação. Padrões de qualidade, linguagem implementada, políticas de integridade para Banco de Dados, limites de recursos e ambientes operacionais.

Pressman (1995), em seu livro, analisa os princípios para uma boa especificação:

1. Separar funcionalidade de implementação. A especificação é uma descrição daquilo que é desejado, e não de como tem de ser realizado (implementação).
2. Utilizar uma linguagem de especificação orientada ao processo. Todas as partes interagentes devem ser especificadas, e não apenas um componente do sistema.
3. Fazer a especificação que abranja o sistema como um todo. Somente dentro do contexto de todo o sistema e da interação entre suas partes é que o comportamento de um componente em específico pode ser definido.
4. Inserir na especificação o ambiente no qual o sistema opera.
5. Especificar o sistema da forma como ele é percebido pelos usuários.
6. Descrever a especificação de modo que a mesma seja completa, formal, para que possa ser usada para determinar se a implementação proposta a satisfaz.
7. Descrever uma especificação de modo que a mesma possa ser expansível, pois a mesma é um modelo (abstração) de uma situação real, sendo assim nunca será totalmente completa.

8. Descrever a especificação de modo que suporte mudanças exigidas pelo tempo, sem comprometer uma funcionalidade (parte de um sistema) que não será mudada.

   O analista pode utilizar um modelo específico para elaborar este documento. A estrutura do documento vai depender da metodologia escolhida e do tempo para seu desenvolvimento. No geral, o modelo é basicamente textual, e utiliza o mínimo possível de termos técnicos. A especificação pode ser ilustrada com desenhos e diagramas que representem a compreensão que os analistas tiveram dos requisitos do sistema a ser desenvolvido.

2.2.5. Modelagem

Esta atividade visa construir modelos gráficos que representem de modo explicativo as características ou o comportamento do sistema. Estes modelos visam:

- Estabelecer uma visão comum do ambiente antes da automação;
- Servir como suporte para negociação e especificação de requisitos e possibilidade futura;
- Representar, avaliar e refinar conceitos do projeto;
- Tratar a complexidade do problema por níveis de abstração;
- Promover facilidades para geração de testes de aceitação.

2.3. *Unified Modeling Language* (UML)

Segundo Pressman (2005), os métodos da Engenharia de Software muitas vezes utilizam uma notação gráfica ou orientada a linguagem de modo como forma de propiciar apoio ao desenvolvimento e manutenção de sistemas.

A UML é uma linguagem de modelagem de sistemas. A UML não é uma metodologia de desenvolvimento, ou seja, não propõe passos e atividades a serem seguidas no desenvolvimento de um software. O objetivo da mesma é auxiliar os analistas a definir as características do sistema, seus requisitos, como irá se comportar e a dinâmica dos processos.

Segundo Booch et al. (1998) a UML pode ser entendida como:

> ... uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos de um sistema de software intensivo. A UML oferece uma maneira padrão para escrever projetos de um sistema, que abrange a parte conceitual, tais como negócios processos e funções do sistema, bem como coisas concretas, como as classes escritas em uma linguagem de programação específica, esquemas de banco de dados e componentes de software reutilizáveis.

Ainda de acordo com Booch et al. (1998), a UML aborda a documentação da arquitetura de um sistema e todos os seus detalhes. A UML também fornece uma linguagem para expressar requisitos e auxiliar os testes. Finalmente, a UML fornece uma linguagem para modelar as atividades de planejamento e gestão de projetos de lançamento, pois os artefatos resultantes são cruciais para controlar, medir e comunicar sobre um sistema durante o seu desenvolvimento e após a sua implantação.

Essa linguagem vem consolidando-se como uma linguagem padrão de modelagem de software, adotada pela grande maioria das empresas de desenvolvimento de sistemas.

Para representar as diferentes partes de um sistema, a UML conta com elementos de modelo. Estes elementos são usados para criar diagramas, que representam determinada parte ou ponto de vista do sistema.

Um diagrama é a apresentação gráfica de um conjunto de elementos. Um diagrama é utilizado para visualizar um sistema de diferentes perspectivas, ou seja, um diagrama pode ser entendido como uma projeção do sistema. Cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada visão, alguns tendem a focar no sistema de forma mais geral, apresentando uma visão externa, já outros oferecem um enfoque mais técnico ou uma característica mais específica do processo. A utilização dos diagramas permite ao analista ter uma visão do sistema de modo a descobrir falhas e incoerências, diminuindo ou até mesmo evitando a ocorrência de erros.

De acordo com Booch et al. (1998), a UML inclui nove tipos de diagramas, que serão mostrados a seguir.

2.3.1. Diagrama de caso de uso

Este diagrama possibilita apresentar o sistema através da visão do usuário. É o diagrama mais informal e flexível da UML. Geralmente é utilizada no início da modelagem do sistema, nas fases de levantamento e análise de requisitos do sistema, porém pode vir a ser consultado durante todo o desenvolvimento do projeto.

Um diagrama de casos de uso mostra um conjunto de casos de uso e atores e seus relacionamentos. Atores são entidades externas que interagem com o sistema. Os atores representam pessoas, usuários, dispositivos, hardwares e/ou software. Cada ator pode participar de vários casos de uso. Os casos de uso descrevem funcionalidades estimuladas pelos atores, enfim modela a interação entre atores e sistemas. Abaixo, a figura 5 mostra um diagrama de caso de uso em UML.

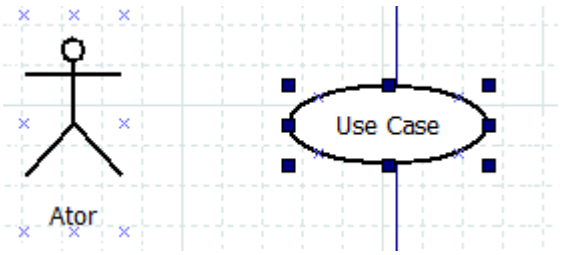

Figura 5 - Representação do ator e use caso em UML

As interações podem ocorrer entre atores, entre atores e casos de uso e entre casos de usos. As relações entre atores expressam as relações de negócios dentro da empresa (figura 6):

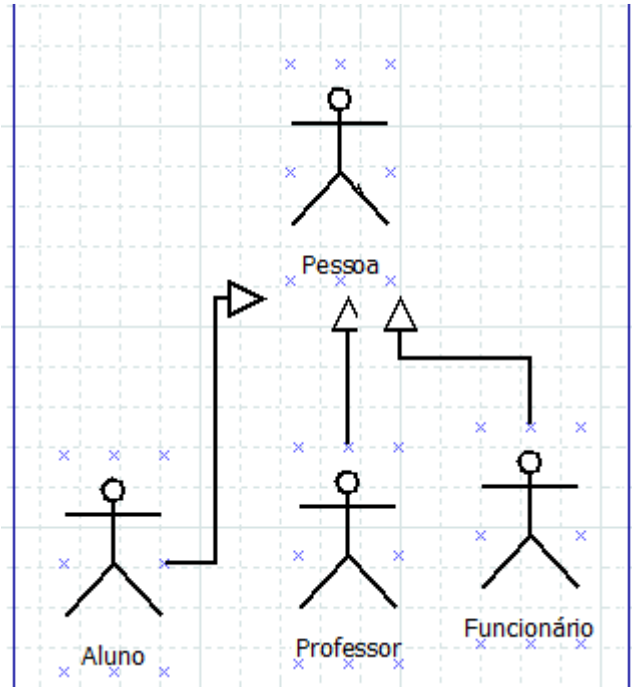

Figura 6 - Representação entre relacionamento entre atores em UML.

As relações entre atores e casos de uso expressam a comunicação entre os mesmos. Estas comunicações podem ser entendidas como estímulos dos atores aos casos de uso. A notação UML para este tipo de relacionamento é um segmento de reta ligando ator e caso de uso, representado pela figura 7.

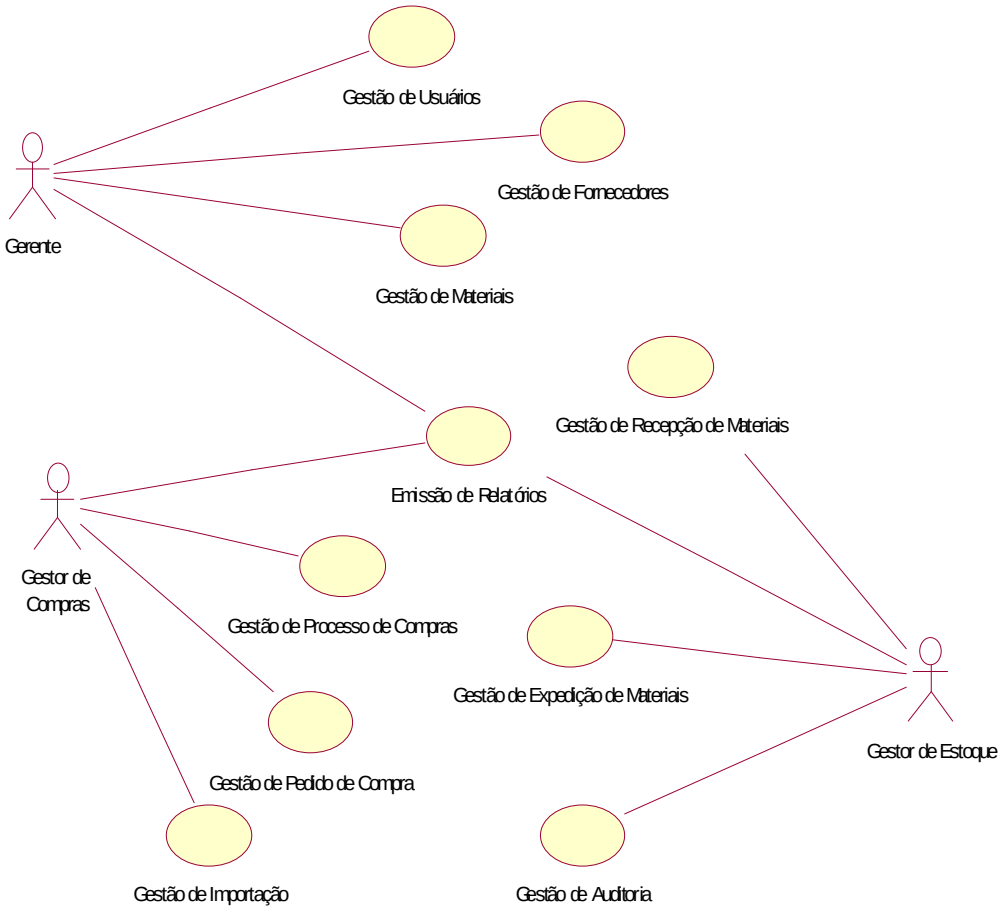

Figura 7 - Representação entre relacionamento entre ator e caso de uso em UML.

Os relacionamentos entre casos de uso são estruturais e não de comunicação. Os três tipos mais comuns de relacionamento entre casos de uso são:

Inclusão: especifica que um caso de uso inicia ou inclui funcionalidade de outro caso de uso. Os relacionamentos de inclusão indicam obrigatoriedade na execução.

Extensão: descrevem situações opcionais, em que situações específicas um caso de uso pode ou não utilizar uma funcionalidade de outro caso de uso.

A figura 8 mostra a notação gráfica destes relacionamentos.

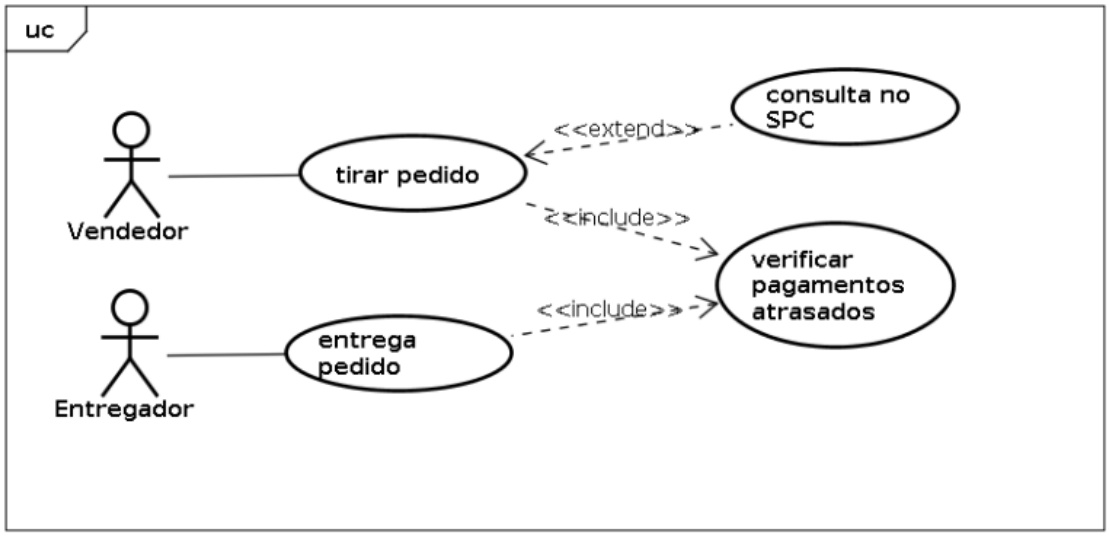

Figura 8 - Exemplo de relacionamento entre casos de uso - inclusão e extensão. Fonte: http://arezi.wordpress.com/2010/10/20/casos-de-uso-diferencas-entre-include-extend-e-generalizacao/ acessado em 07/04/2013.

Generalização: especifica que um caso de uso herda as características de outro caso de uso considerado base. Este caso de uso geral agrupa funcionalidades comuns a outros casos de uso em questão. A generalização pode ser entendida como uma herança. A generalização está representada na figura 9.

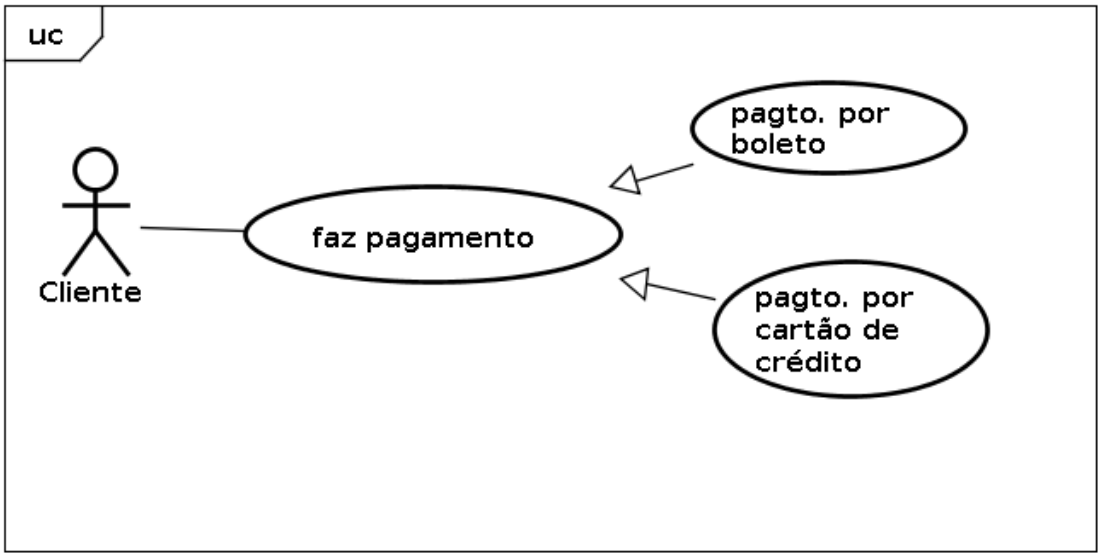

Figura 9 - Exemplo de relacionamento entre casos de uso - generalização. Fonte: http://arezi.wordpress.com/2010/10/20/casos-de-uso-diferencas-entre-include-extend-e-generalizacao/ acessado em 07/04/2013.

A especificação das funcionalidades de um sistema na forma de casos de uso permite uma visão mais abrangente das aplicações do sistema, favorecendo um levantamento mais completo e preciso de suas atribuições.

2.3.2. Diagrama de sequência

Um diagrama de sequência está associado à funcionalidade de um caso de uso. Este diagrama apresenta a funcionalidade de um caso de uso através de uma sequência de eventos. Um cenário em que é ilustrado um conjunto de interações entre objetos num determinado período de tempo. Um diagrama de sequencia contém atores, objetos, eventos e mensagens. A figura 10 apresenta um exemplo de diagrama de sequência.

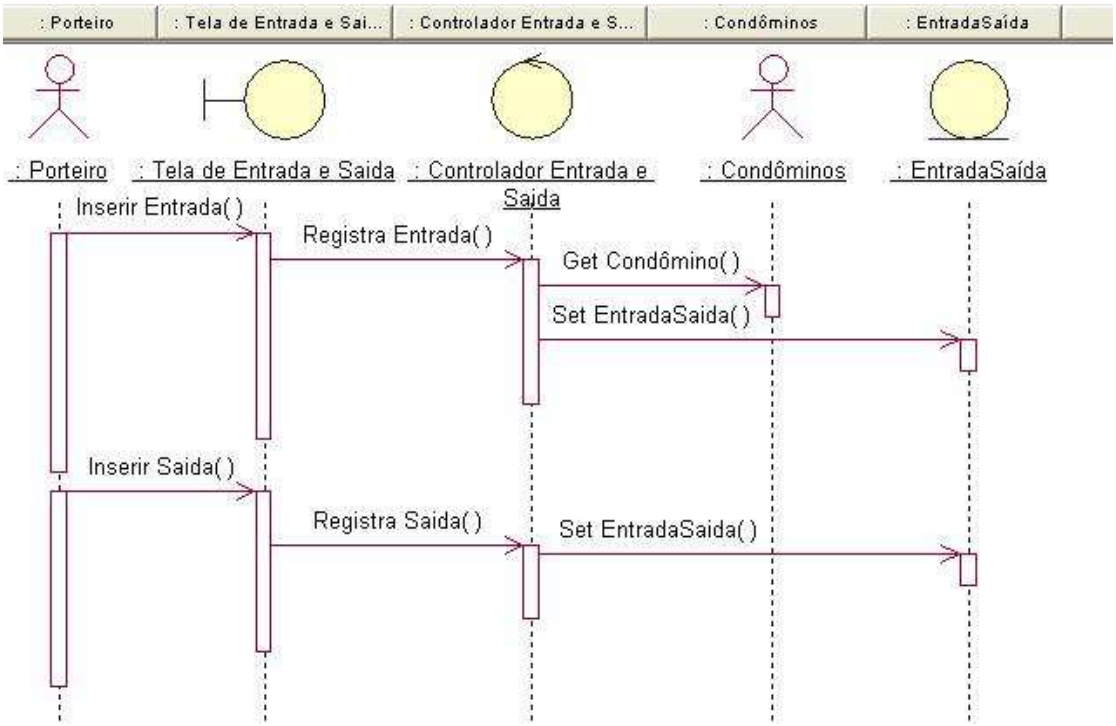

Figura 10 - Exemplo diagrama de sequencia de um sistema para entrada e saída de um condomínio.

2.3.3. Diagrama de colaboração

Um diagrama de colaboração, representado pela figura 11, está associado à funcionalidade de um caso de uso. Este diagrama apresenta uma visão da interação entre os objetos do domínio. Um diagrama de colaboração contém atores, objetos, ligações, mensagens e opcionalmente fluxos de dados.

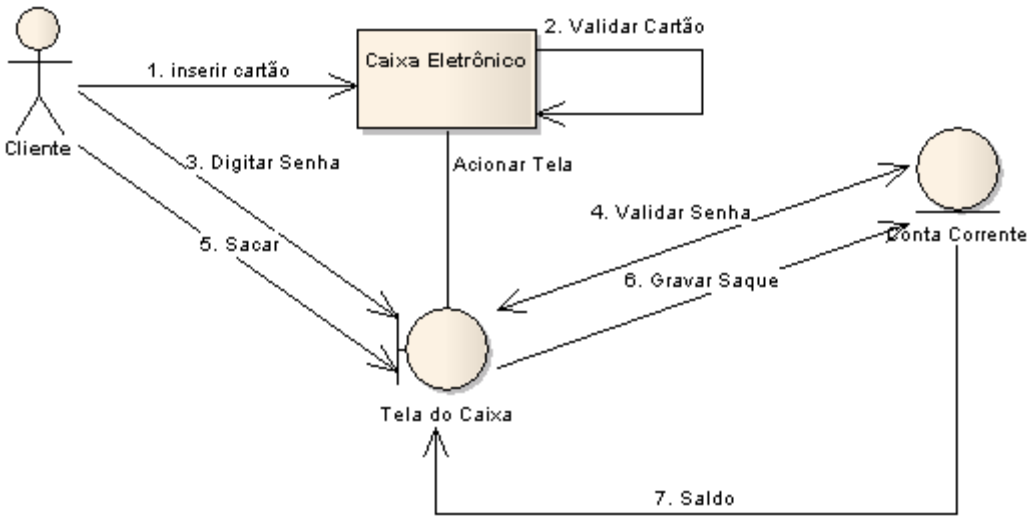

Figura 11 - Exemplo diagrama de colaboração em UML. Fonte http://techblog.desenvolvedores.net/tag/diagrama-de-colaboracao/ acessado em 07/04/2013.

2.3.4. Diagrama de classes

Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. Seu principal objetivo é permitir a visualização das classes que constituirão o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si. Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em como definir a estrutura lógica das mesmas. O diagrama de classe serve ainda como base para a construção da maioria dos outros diagramas da linguagem UML.

Os diagramas de classe são importantes não só para visualizar, especificar e documentar de modo estrutural os modelos, mas também para a construção de sistemas executáveis por meio de engenharia reversa.

Uma classe define os atributos e os métodos de um conjunto de objetos. Os objetos seriam instâncias desta classe e teriam o mesmo comportamento e atributos da classe. Na notação gráfica da UML as classes são representadas por retângulos, com seu nome, e podem também mostrar os seus atributos e operações. A notação gráfica de uma classe pode ser observada na figura 12:

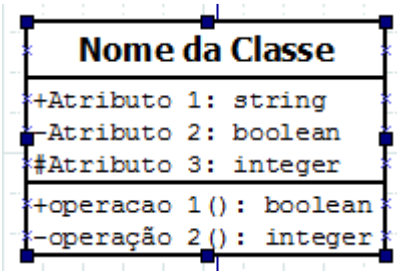

Figura 12 - Representação de uma classe em UML.

Os atributos são mostrados por seu nome, tipo e visibilidade. Sendo que a visibilidade pode ser:

- *+atributos públicos*
- *#atributo protegidos*
- *-atributos privados*

As operações ou métodos também são exibidos por nome e podem conter parâmetros e valores de retorno. Os símbolos usados para indicar a visibilidade dos métodos são os mesmos dos atributos.

De modo geral, as classes podem se relacionar das seguintes maneiras:

- Generalização/Especialização: a generalização pode ser entendida como um processo de herança. Neste tipo relacionamento tem-se uma classe base e classes derivadas. Em UML, são representadas por uma linha conectando duas classes, com uma seta no lado da classe base, conforme representado na figura 13.

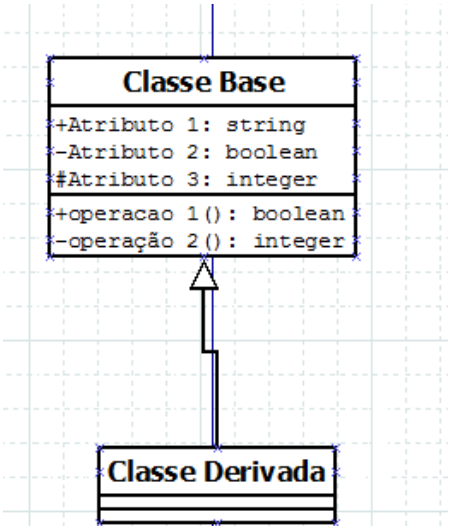

Figura 13 – Exemplo de representação de uma generalização em UML.

- Associações: seu objetivo é definir a maneira como as classes estão unidas e interagem entre si, compartilhando informações. Em UML, associações são representadas como linhas conectando as classes que participam do relacionamento, podendo representar a regra e a multiplicidade de cada um dos participantes. A figura 14 representa uma associação:

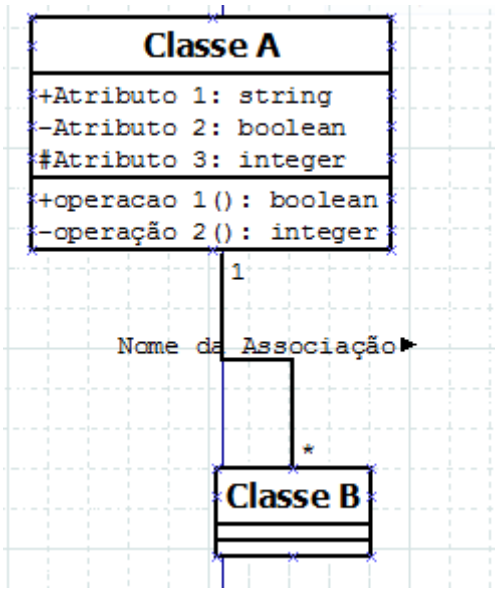

Figura 14 – Exemplo de representação de uma associação em UML.

- Agregação: considerada um tipo especial de associação, onde as classes fazem parte de um relacionamento “todo-parte”. Este tipo de associação tenta demonstrar uma relação todo/parte entre os objetos associados. Objeto-parte

não pode ser destruído por um objeto diferente do objeto-todo. O símbolo de agregação difere do de associação por conter um losângulo na extremidade da classe que contém o objeto-todo. A agregação está apresentada na figura 15:

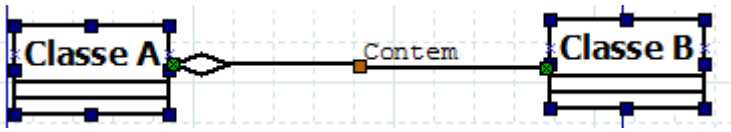

Figura 15 – Exemplo de representação de uma agregação em UML.

- Composição: são associações onde as partes não podem existir independentes. Existem somente dentro do todo. Os objetos parte não podem existir quando o objeto todo é destruído, ou seja, se o todo for destruído, suas partes obrigatoriamente também passarão a não existir. A figura 16 mostra um exemplo de representação gráfica de uma agregação em UML.

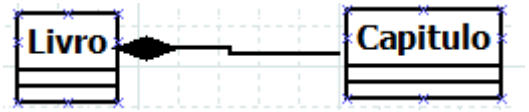

Figura 16 – Exemplo de representação de uma agregação em UML.

Os diagramas de classe podem conter outros elementos além de classes:

- Interfaces: são classes abstratas que significam instâncias que não podem ser diretamente criadas delas. Elas podem conter operações mas não podem conter atributos. Classes podem derivar de interfaces (através da realização de uma associação) e instâncias podem então ser feitas destes diagramas.
- Tipos de dados: são primitivos uma vez que são tipicamente construídos numa linguagem de programação. Exemplos comuns são inteiros e lógicos. Eles não podem ser relacionados à classes mas classes pode se relacionar com eles.
- Enumerações: é uma lista simples de valores. Um exemplo típico é uma enumeração para dias da semana. As opções de uma enumeração são chamadas Literais de Enumeração. Como tipos de dados, elas não podem ter relacionamentos para classes, mas classes podem relacionar-se com elas.

- Pacotes: representam um espaço de nomes numa linguagem de programação. Num diagrama eles são usados para representar partes de um sistema que contém mais de uma classe, talvez centenas de classes. Os pacotes podem ser representados conforme figura 17.

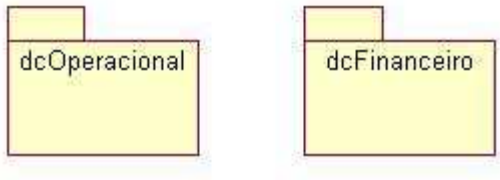

Figura 17 - Exemplo de pacote de classes em UML.

2.3.5. Diagrama de objeto

Um diagrama de objetos mostra um conjunto de objetos e seus relacionamentos. É bastante semelhante ao diagrama de classes já que os objetos são instâncias das mesmas. A figura 18 mostra um exemplo de diagrama de objeto.

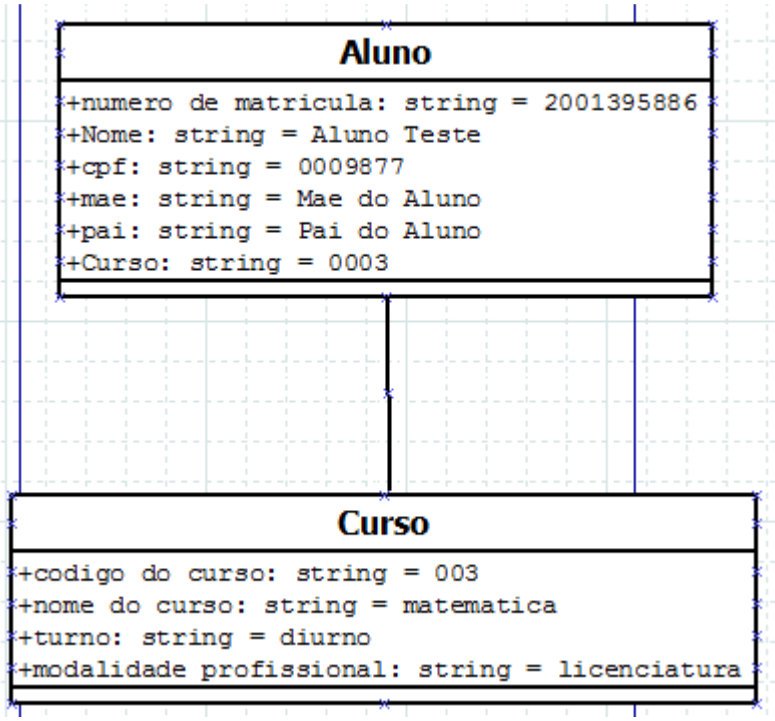

Figura 18 - Exemplo de representação diagrama de objetos em UML

2.3.6. Diagrama de estados

Um diagrama de estados apresenta o ciclo de vida de uma dada classe, os eventos que causam a transição de um estado para outro e as ações que resultam de uma alteração de um estado. O diagrama de estados deve ser definido para classes que denotem um comportamento dinâmico significativo. Estes diagramas permitem representar decisões, sincronizações, atividades, estados, transições, estados iniciais e estados finais. Abaixo, a figura 19 apresenta um exemplo de diagrama de estados.

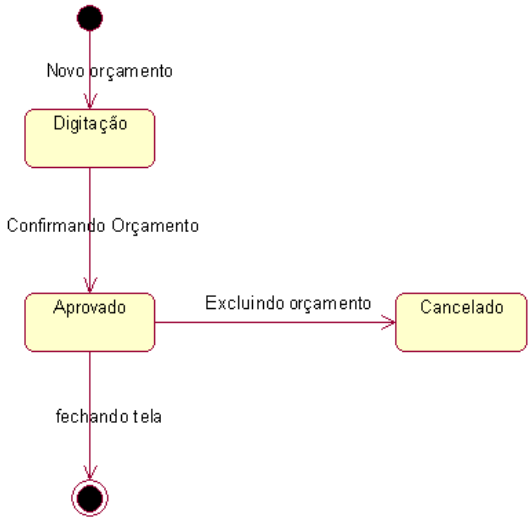

Figura 19 - Exemplo diagrama de estados em UML.

2.3.7. Diagrama de atividades

O diagrama de atividade, representado pela figura 20, mostra uma sequência de atividades. Uma atividade é interpretada como a tomada de uma ação e, por vezes, o referente estado. O diagrama permite representar condições para a tomada de decisões e permite representar comportamento paralelo de atividades.

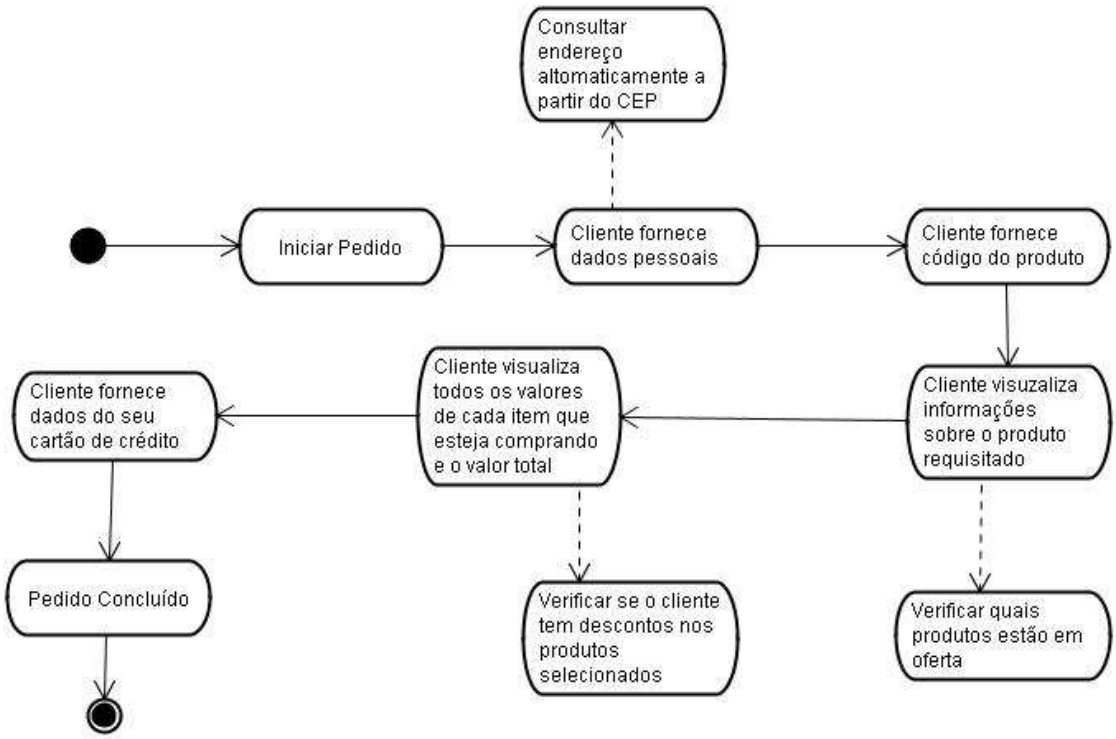

Figura 20 - Exemplo de diagrama de atividades em UML. Fonte: http://www.dimap.ufrn.br/~jair/ES/exemplos/fazerpedido/Fazerpedido5.html acessado em 07/04/2013.

2.3.8. Diagrama de componentes

O diagrama de componentes modela o sistema baseado em componentes, indicando quais são e seus relacionamentos. Por componentes neste diagrama entendem-se os aspectos físicos do sistema que está sendo modelado. Este diagrama mostra os artefatos de que os componentes são feitos, como arquivos de código fonte, bibliotecas de programação ou tabelas de bancos de dados.

Um componente é mostrado em UML como um retângulo com uma elipse e dois retângulos menores do lado esquerdo. O nome do componente é escrito abaixo ou dentro de seu símbolo. A figura 21 apresenta um exemplo de diagrama de componentes.

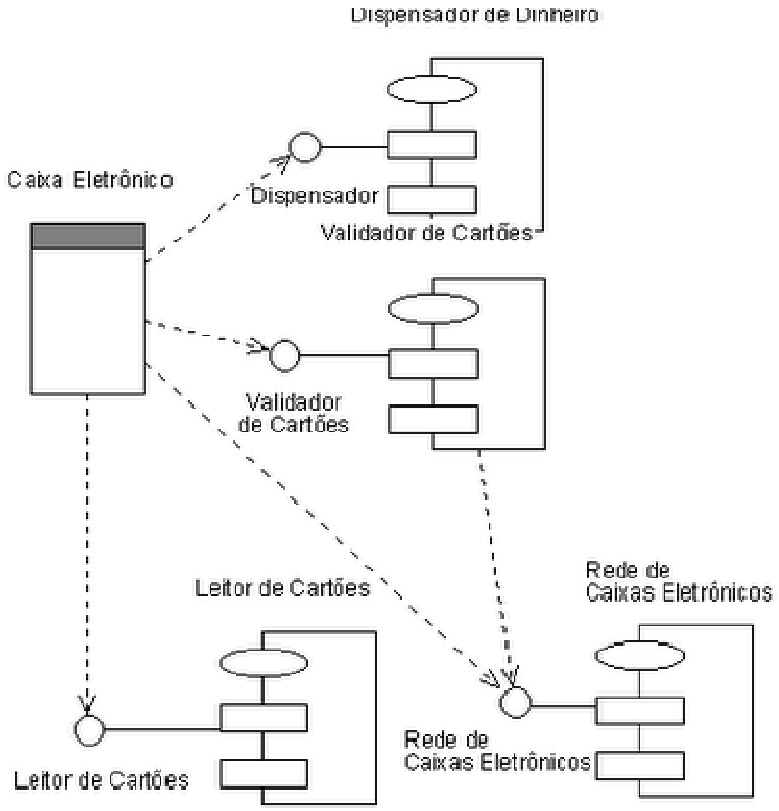

Figura 21 -- Exemplo de diagrama de componentes em UML. Fonte: http://luciano-ti.blogspot.com.br/2010/08/uml.html acessado em 07/04/2013.

2.3.9. Diagrama de implantação

Este diagrama modela o uso físico do sistema, considerando computadores, dispositivos e suas interconexões. Seu principal elemento é o nodo, que representa um recurso computacional. A figura 22 apresenta um diagrama de implementação.

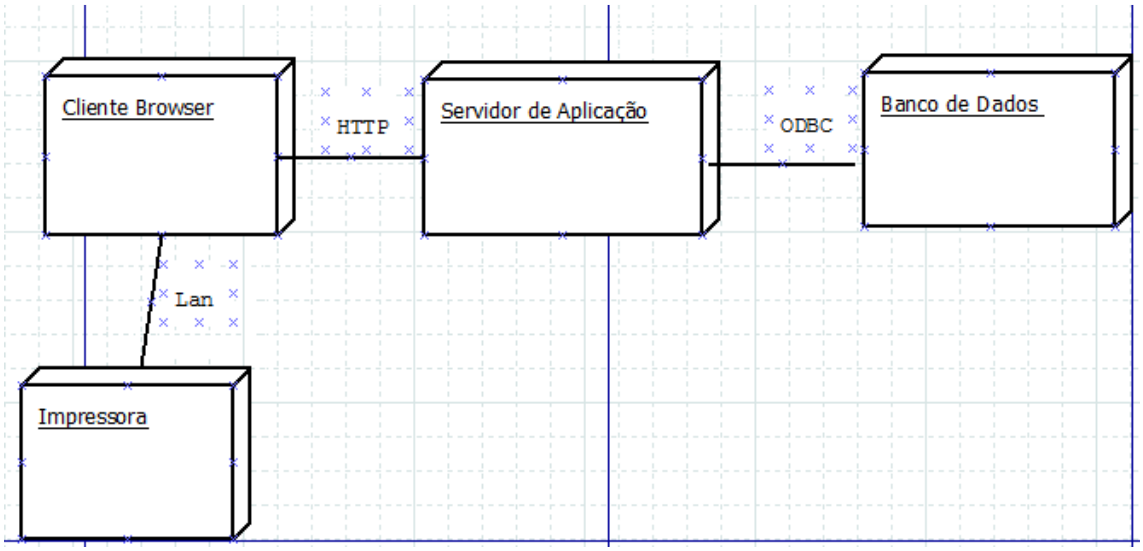

Figura 22 - Exemplo de diagrama de implementação em UML.

3. ESTUDO DE CASO: PRODUÇÕES EMBARGADAS

Este capítulo visa explanar sobre o Sistema Opus, BDTD – Biblioteca Digital de Teses e Dissertações, além de descrever a situação atual e a solução proposta para o estudo de caso deste trabalho, a implementação de um mecanismo de controle e armazenamento, no Sistema Opus, das produções embargadas.

3.1. Sistema OPUS e BDTD- Biblioteca Digital de Teses e Dissertações

O Opus é sistema de registro das produções bibliográficas da UFMG. O Opus permite o lançamento por professores e funcionários de dados sobre as produções e também o armazenamento da obra em formato eletrônico.

O Opus possui um controle de acesso utilizando perfis de usuários. Abaixo os perfis existentes no sistema:

- Administrativos:
  - BU – Biblioteca Universitária – usuários deste perfil acessam e informavam dados referentes a “Biblioteca”.
  - CPPD – Comissão Permanente de Pessoal Docente – usuários deste perfil acessam relatórios exclusivos deste departamento.
  - PRPq – Pró-Reitoria de Pesquisa - usuários deste perfil acessam relatórios exclusivos deste departamento.

- Departamento – usuários deste perfil podem revisar e corrigir produções dos departamentos a qual estão associados. Isso inclui todos os cursos do departamento.
- Curso - usuários deste podem revisar e corrigir produções somente do curso a qual estão associados.

O lançamento de autores é uma parte importante para garantir a confiabilidade da informação, já que no lançamento de produções, os dados referentes aos autores são os mesmo disponíveis na base de dados da UFMG. Porém é possível a inserção de autores externos a Instituição.

Os lançamentos de produções no Opus passam por um fluxo de controle. Estes fluxos de controle permitem que departamentos, cursos, pró-reitorias e outras instâncias possam verificar e corrigir dados, garantindo a qualidade da informação armazenada. Além disso, é possível verificar as alterações realizadas e seus autores, possibilitando rastreabilidade e segurança da informação.

Não existe fechamento de sistema, é possível lançar dados ao longo do ano, à medida que as produções forem ocorrendo.

Os dados armazenados no Opus são exportados para outros sistemas, de uso interno a UFMG, tais como sistemas de uso específicos de um setor ou departamento e para a BDTD. Este último sistema permite o acesso à obra completa ou parte da mesma. Além destes sistemas considerados de uso interno, os dados lançados no Opus são utilizados para alimentar vários outros sistemas externos. Estes sistemas externos podem ser entendidos como sistemas desenvolvidos e mantidos por organizações terceiras, que possuem algum vínculo com a UFMG. Como exemplo de sistemas externos pode-se citar: o Coleta Capes**,** BDTD e IBICT (Instituto Brasileiro de Informação em Ciência e Tecnologia), Sistema Pergamum (gerenciador de serviços de bibliotecas).

São exportadas do Opus para a BDTD todas as produções bibliográficas dos tipos Tese, Dissertação e Monografias de Especialização e que foram aprovadas pelos departamentos e revisadas pela BU. A BDTD está implementada em *Dspace-Institutional Digital Repository System* (projeto colaborativo da MIT *Libraries* e a

Hewlett-Packard Company)[4] que foi desenvolvido para possibilitar a criação de repositórios digitais com funções de captura, distribuição e preservação da produção intelectual, permitindo sua adoção por outras instituições.

O DSpace permite o gerenciamento da produção científica em qualquer tipo de material digital, dando-lhe maior visibilidade e garantindo a sua acessabilidade ao longo do tempo.

Na BDTD ficam armazenados os metadados e os anexos referentes às produções. Por metadados entende-se entendido como “dados sobre outros dados”, ou seja, são informações úteis para identificar, localizar, compreender e gerenciar os dados. A fonte de dados está no Opus e a BDTD apenas referencia estes dados através dos metadados especificados. A exportação acontece de forma automática e transparente para o usuário, através de um agente desenvolvido em Java, que está agendado no servidor e que roda de 5 em 5 minutos. Este agente é responsável por conectar no Opus e buscar as informações necessárias para o BDTD, bem como os anexos das produções. O agente então gera um XML da produção. O XML é uma linguagem de marcação[5] para a criação de documentos com dados organizados hierarquicamente.

Após a geração do XML, o agente aplica o XLST no XML gerado anteriormente, para converter o XML no formato desejado, no caso o *Dublin Core*.[6] O XSLT também é uma linguagem de marcação e sua função é definir a apresentação dos documentos XML nos browsers e outros aplicativos que a suportem.

O esquema de metadados padrão do DSpace é o *Dublin Core*. O agente executa um comando de importação para Dspace (BDTD). A importação acontece por linhas de comando, usando bibliotecas próprias e inerentes ao Dspace. A figura 23 mostra o esquema em alto nível de exportação de dados e anexos para a BDTD.

[4] Fonte: http://www.dspace.org/, acessado em 25/06/2013.
[5] Fonte: http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm, acessado em 25/06/2013.
[6] Dublin Core é um esquema de metadados que visa descrever objetos digitais, tais como, vídeos, sons, imagens, textos e sites. Fonte: http://dublincore.org/ ,acessado em 25/06/2013.

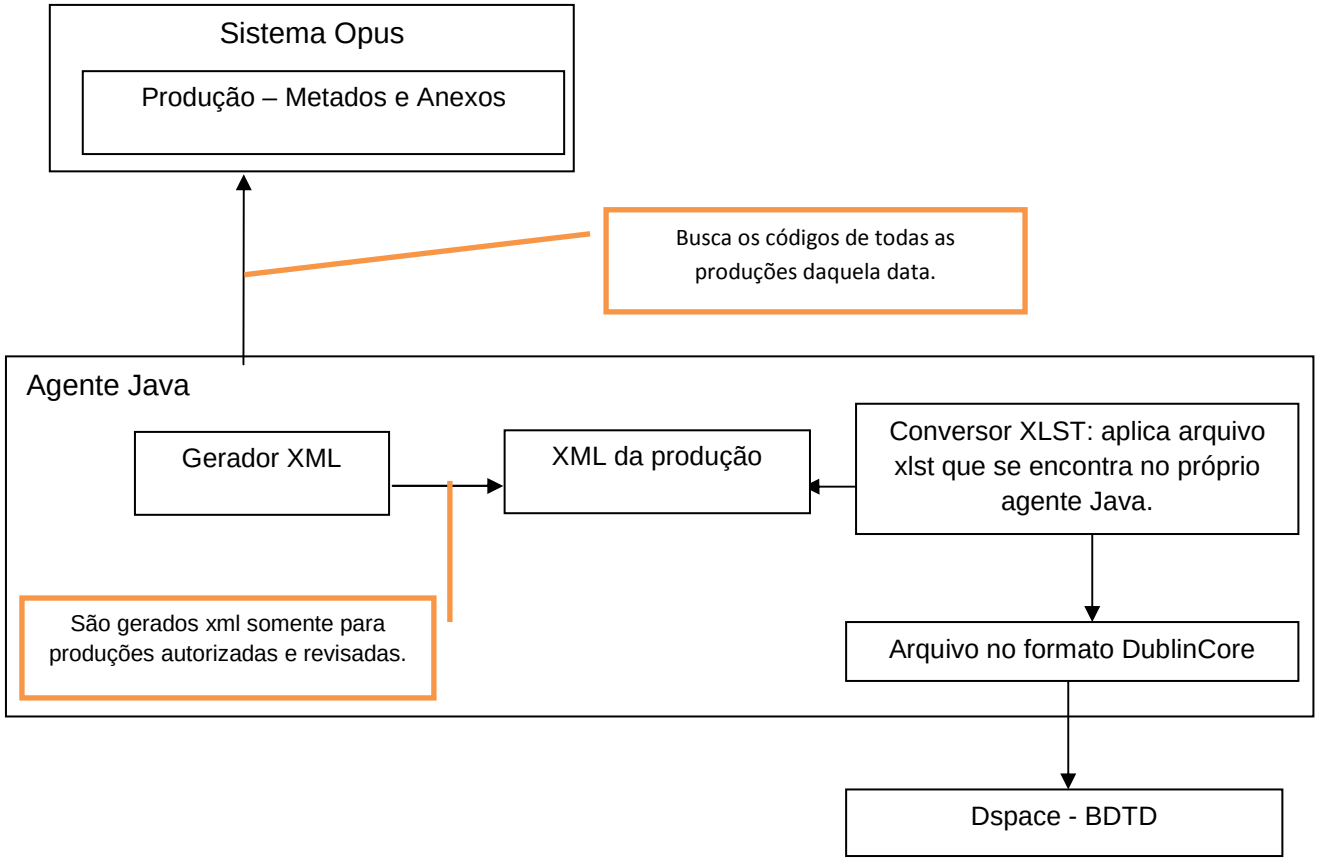

Figura 23 - Esquema de exportação Opus - Dpsace.

3.2. Situação Atual

Existe um fluxo para que uma produção seja disponibilizada na BDTD. O fluxo consiste nas seguintes etapas:

- A secretária do curso de Pós Graduação lança a tese/dissertação no Opus (informações que serão de base como metadados para a BDTD e arquivo contendo texto completo no formato pdf);
- O aluno entrega à secretária a autorização para divulgação;
- Secretária autoriza no OPUS acesso ao texto completo da tese mediante apresentação da autorização pelo aluno;
- Biblioteca verifica os dados da tese autorizada, fazendo os devidos ajustes e catalogação;
- Biblioteca autoriza e revisa a tese/dissertação;
- As informações são exportadas automaticamente para o BDTD.

A Biblioteca Universitária pode verificar e corrigir os dados das teses e dissertações e também é responsável por entrar com informações específicas de indexação das mesmas. Uma produção revisada pela BU não pode mais ser alterada pelo cadastrante ou autores, cabendo apenas ao curso ou departamento associado, fazer as alterações necessárias.

No Sistema Opus para usuários com perfil BU, na produção aparece a aba Biblioteca, com a opção de selecionar o estado de revisão pela BU: [Revisado|Não Revisado]. Nas Teses, Dissertações e Monografias de Especialização, além desta opção, aparecem os campos de: Cabeçalho de assunto, Biblioteca depositária e Número de chamada.

Conforme mencionado anteriormente, dentre as atividades de verificação de dados da produção por parte da biblioteca, consiste em revisar os dados dessas produções, inserindo e autorizando anexos e modificando o estado da produção.

As produções bibliográficas referentes a Teses e Dissertações possuem três estados para o campo Estado de revisão pela BU:

- Vazio (em branco), a produção assume este estado após a sua criação.
- Revisado, após analisar e constar que a produção está correta, a bibliotecária altera seu estado para “Revisado”.
- Não Revisado, este estado é utilizado para produções que foram analisadas pela BU e foi constado algum dado incorreto Além disso, este estado também é utilizado para retirar uma produção que foi exportada incorretamente para BDTD.

Em determinadas situações, como patentes, por exemplo, o autor solicita que o texto completo de sua produção não seja disponibilizado na BDTD. Nestas situações, o autor pode ou não liberar parte do texto. Neste momento a produção é embargada, ou seja, seu texto completo não estará disponível na BDTD por um período pré-determinado.

Atualmente, o estado de embargado não existe no Opus. Desta forma, para que uma produção que se encontra nesta situação não fique disponível na BDTD, as

bibliotecárias alteram o seu estado para Não Revisado, fazendo assim com que a mesma seja retirada da BDTD.

Como o Sistema Opus não contempla este tipo de estado, não há controle dessas produções embargadas, o que gera muitos problemas para a Biblioteca Universitária da UFMG. O controle é realizado pelas bibliotecárias de forma manual, o que faz com que o processo esteja suscetível e propenso a erros e falhas.

É importante salientar que a situação de embargo aplica-se somente as produções de teses de doutorado e dissertações mestrado. O fluxo de exportação para as monografias de especialização não será alterado.

3.3. Metodologia de trabalho

A primeira atividade para início dos trabalhos de levantamento de requisitos visando atender a demanda da BU foi a escolha de funcionário(s) denominado(s) “usuário(s) chave”. Estes usuários são selecionados por possuírem conhecimento consolidado dos processos e por serem pessoas que se destacam nas suas áreas. Eles devem fazer parte da equipe de levantamento e testes do sistema, fornecendo informações sobre os processos de trabalho, bem como avaliar como o sistema se integra nas atividades e atende os requisitos solicitados. Tais usuários-chave recomendam, ainda, a melhor forma de adequação do sistema às necessidades da Instituição e propõem melhorias. Além disso, participam dos testes e homologam a implantação do sistema.

Em reuniões realizadas com a diretoria da Biblioteca Universitária, ficou definido que a Bibliotecária Chefe da FAFICH/UFMG, seria responsável por auxiliar os analistas do LCC neste processo de levantamento das necessidades da Biblioteca Universitária. Já nas etapas posteriores, tais como fase de testes e homologação, outros usuários seriam envolvidos. Estes usuários ainda serão definidos em momento oportuno.

A partir da definição acima, iniciou-se os trabalhos de levantamento de requisitos. A metodologia escolhida para tal atividade foi a realização de reuniões, utilizando como técnica de levantamento de requisitos o uso entrevistas com a usuária. Esta técnica foi escolhida por ser simples e produzir bons resultados na fase inicial de obtenção de dados.

As reuniões eram planejadas de forma a terem objetivos pré-definidos a serem alcançados, além de definir hora, lugar e duração, levando-se em conta a agenda do entrevistado. A reunião era conduzida pela gerente ou analista responsável do LCC, com interação direta com a usuária.

Ao término da entrevista, atas de reuniões eram registradas, revendo e confirmando o que foi registrado com o entrevistado, a fim de evitar confusões no entendimento das informações. Deixamos sempre aberta a possibilidade de voltar a fazer contato e esclarecer dúvidas que surjam durante a análise das informações levantadas.

Na etapa posterior os analistas reavaliaram as atas de reuniões, a fim de reorganizar as idéias, estudá-las e verificar possíveis incoerências com reuniões anteriores ou outras fontes de informação. Neste momento, já se inicia a elaboração do documento de Plano de Projeto contendo essas primeiras informações. Este documento vai se aprimorando a cada reunião realizada. A cada versão gerada deste documento, é enviado ao usuário para que o mesmo possa verificar a coerência das informações levantadas, bem como sugerir acertos e melhorias. Após várias reuniões o escopo do projeto é fechado. Importante salientar, que durante este processo, definiu-se que o estado de embargado aplica-se somente a teses e dissertações. O fluxo de monografias de especialização não será alterado.

Esta etapa do trabalho foi baseada nas atividades propostas de levantamento, análise e negociação de requisitos de um Processo de Engenharia de Requisitos. Tendo em vista que, nesta situação, a negociação de requisitos não engloba a inclusão de novos requisitos em um projeto na fase de desenvolvimento (codificação), mas sim na definição de escopo acordado entre as partes.

3.4. Solução Proposta

Com base no escopo do projeto e diante do levantamento inicial das necessidades da Biblioteca Universitária, foram identificadas alterações necessárias no ambiente OPUS e na interface (exportação) entre este sistema e BDTD, de modo a atenderem a situação de embargo de produção.

Um mecanismo de controle deve ser implementado para produções embargadas, de modo a possibilitar a entrada de dados específicos para esta

situação tais como: datas de solicitação, início e fim do embargo, bem como permitir o armazenamento de partes de texto, que serão liberadas, e o texto completo que será embargado. O documento de armazenamento do anexo do arquivo deve conter, além do código da produção, o anexo a que se refere (o Sistema Opus já faz isso atualmente) e o estado de Revisão BU na qual se encontra aquela produção. Dessa forma o agente Java exportador para o Dspace – BDTD entenderá qual anexo deve exportar.

Além disso, o sistema deve controlar vencimentos dos embargos e liberações dos textos. Todas as regras de negócios descritas neste trabalho foram definidas e informadas ao LCC pela Biblioteca Universitária.

Conforme mencionado anteriormente existe somente dois estados válidos para o campo Estado de revisão pela BU: Revisado e Não Revisado, já que o "Vazio" é o valor inicial para qualquer produção inserida e que ainda não passou pela etapa do fluxo de Revisão da BU. Para o projeto de Produções Embargadas um novo estado deve ser acrescentado: "Embargado". A tela para Estado Revisão BU deverá ficar conforme a figura 24:

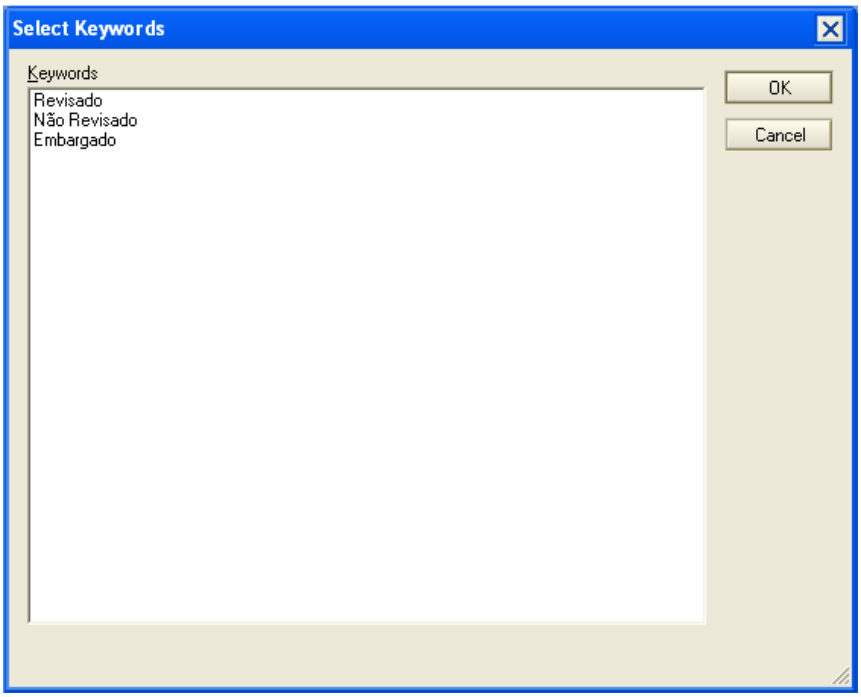

Figura 24 – Proposta de tela para estado Revisão BU contendo o estado de Embargo.

Ao selecionar a opção de do estado de Embargado, o usuário dever informar o motivo da retenção através do campo “Retido por motivo de:” Os motivos do embargo poderão ser caracterizados como:

- Patentes: Projetos em que é necessário sigilo porque irão gerar patentes. Nestes casos, o prazo de embargo é de 5 anos.
- Publicação: Situações em que o autor irá publicar seu trabalho ou projeto e necessita de um tempo de sigilo. Um exemplo é a publicação de um livro. O autor pode liberar somente um capítulo ou o resumo do mesmo. Para este tipo de embargo, o prazo de um ano, podendo ser renovado por mais um ano.

Abaixo uma proposta de tela para o motivo do embargo (figura25):

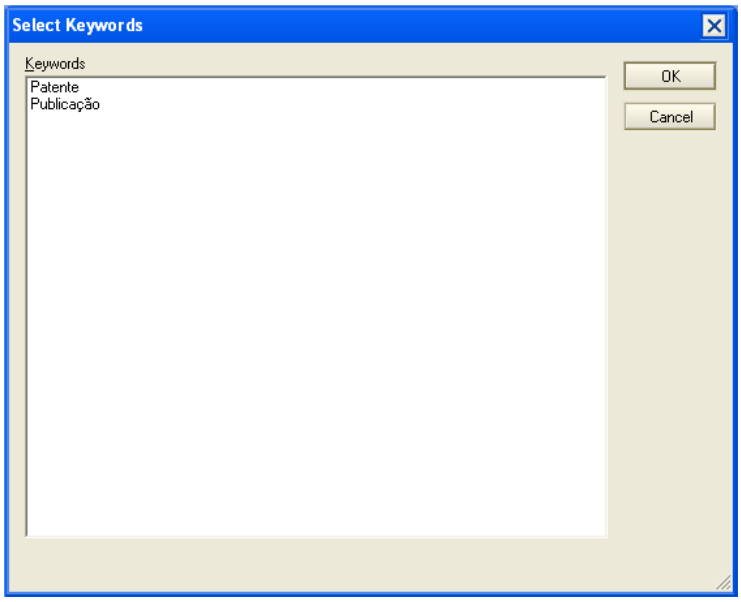

Figura 25 - Proposta de tela para motivo de Embargo

As produções embargadas possuem data de início e de término do embargo. A partir da data inicial informada para o embargo, será calculada automaticamente a data de término (cinco anos para o motivo “Patente” e um ano para o motivo “Publicação”). Entretanto, a edição deste campo estará habilitada e a data de término do embargo poderá ser alterada a qualquer momento a critério da BU.

A data de solicitação do embargo refere-se à data em que o autor solicitou o embargo. Esta data pode coincidir ou não com a data de início do embargo (figura 26).

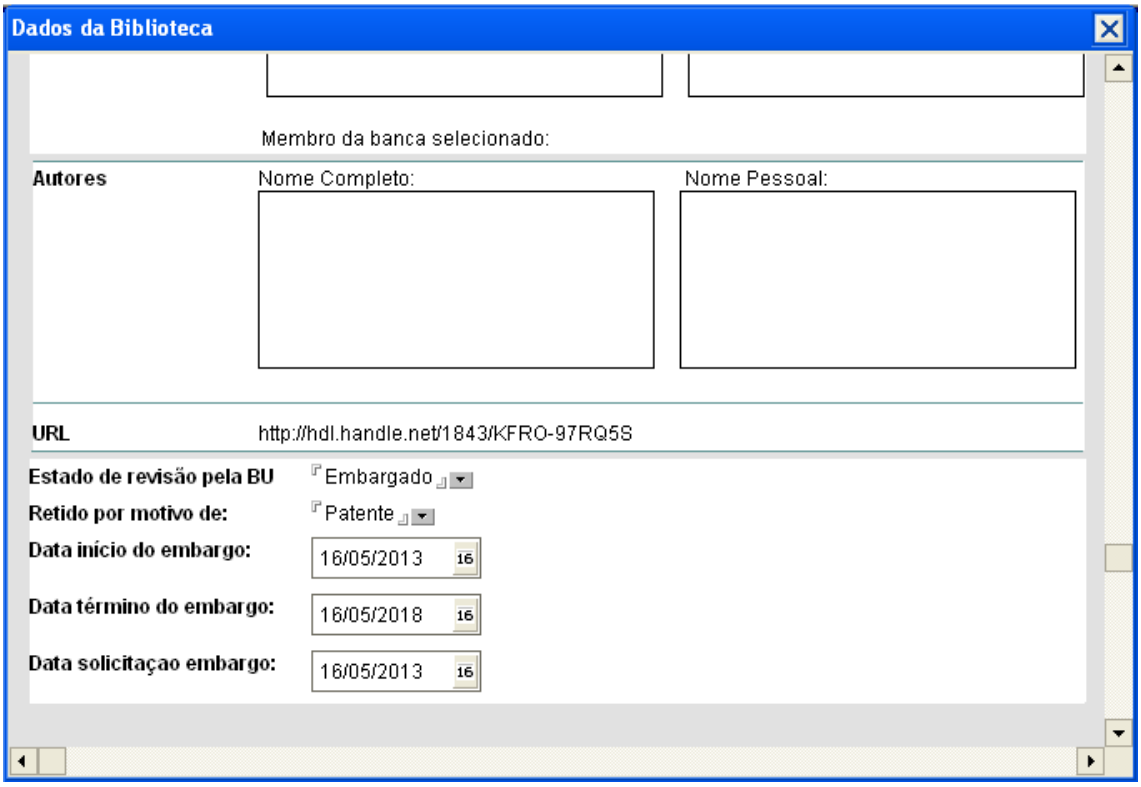

Figura 26 - Proposta de tela Dados Biblioteca com estado embargado, motivo, data de solicitação, início e término do embargo.

Com relação às datas, algumas validações devem ser implementadas tais como:

- A data de solicitação não pode ser maior que a data atual.
- A data de início do embargo não pode ser menor que a data atual.

Para controlar os arquivos (anexos) que serão disponibilizados e os que serão embargados, o Opus contemplará, na aba “Resumo/Texto Completo”, a opção de armazenar o arquivo de texto completo e/ou o(s) arquivo(s) referentes a parte(s) do texto. Porém a parte que se destina a inclusão de anexos de partes de texto somente estará disponível para o usuário se o mesmo informar que a produção será embargada. Se o estado estiver embargado o arquivo referente ao Texto Completo será armazenado somente no Opus, não estará disponível na BDTD. Já os arquivos

que se encontrarem armazenados em Partes do Texto estarão disponíveis na BDTD e serão deletados da base Opus somente quando a produção tiver seu estado de Revisão BU alterado para “Revisado”. Alterações de estado Revisão BU de “Embargado” para “Não Revisado” não comtemplará a deleção do arquivo constante de parte de texto da base Opus. É importante salientar que é de responsabilidade da BU garantir que o conteúdo dos arquivos anexados realmente são textos completos ou partes do mesmo. A figura 27 mostra como será o armazenamento destes anexos de textos.

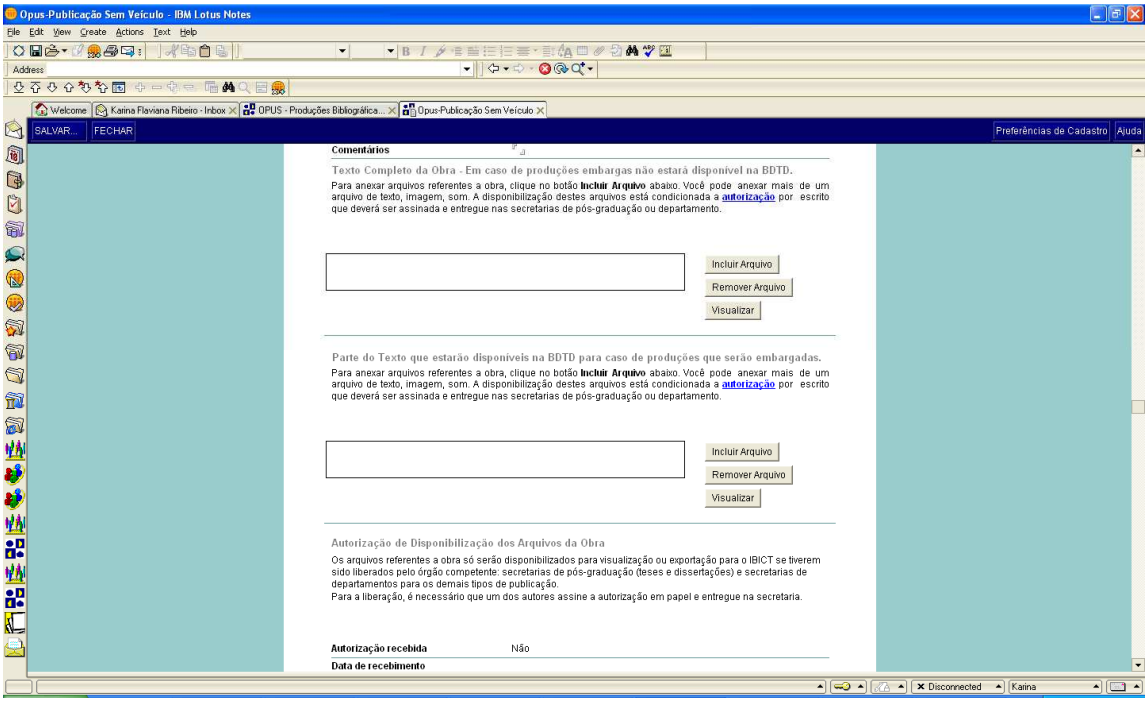

Figura 27 - Tela proposta para armazenamento de arquivos de texto completo e arquivos de parte de texto.

Para controle das produções embargadas, serão implementadas no Opus e disponibilizadas para BU interfaces para visualização referentes somente as produções embargadas. Inicialmente serão desenvolvidas três interfaces:

- Uma interface para visualização de todas as produções com status embargados, independente de datas especificadas.
- Uma interface para visualização de todas as produções com data de embargo vencida.
- Uma interface para visualização de todas as produções com data de embargo a vencer.

Para implementação considera-se data de embargo vencidas as produções que tenham data final de embargo menor que a data atual (data do dia). E considera-se data de embargo a vencer as produções que tenham data final de embargo igual ou maior que a data atual (data do dia). Estas produções serão ordenadas por data, ou seja, aquelas com vencimento mais próximo serão as primeiras a serem exibidas.

As produções embargadas que tiverem com data de embargo final vencidas não terão datas alteradas automaticamente pelo Sistema Opus. É de responsabilidade da Biblioteca Universitária o controle das produções embargadas.

O filtro para estas visões já fica definido dentro do código do Sistema Opus, não existe uma tela para o usuário realizar filtros mais específicos ou diferenciados dos mencionados anteriormente.

Conforme dito anteriormente, atualmente são exportadas do Opus para a BDTD todas as produções bibliográficas dos tipos Tese, Dissertação e Monografias de Especialização e que foram aprovadas pelos departamentos e revisadas pela BU. O agente Java responsável pela exportação do anexo da produção, não faz nenhuma conferência se o mesmo se refere a um texto completo ou parte de um texto.

Para a situação de embargo o agente deverá ser alterado para verificar o Estado Revisão BU, verificando se o mesmo se encontra como “Embargado”. Nestes casos o agente deverá exportar apenas os anexos que constem em “Parte do Texto que estarão disponíveis na BDTD para caso de produções que serão embargadas”. Esta informação estará no documento de anexo com a inclusão de um campo de controle “Estado de Revisão BU” na qual se encontra aquela produção. Esta alteração foi mencionada anteriormente.

Alterações no Estado Revisão BU devem ser contempladas. Uma produção embargada que tiver seu estado alterado para “Não Revisado”, o agente Java de exportação deverá retirar a produção do BDTD. Este fluxo não precisará ser implementado, já que atualmente quando uma produção está com estado “Não Revisado” o agente Java já retira automaticamente da BDTD, independente do estado anterior da produção. Caso o Estado de Revisão BU se altere para

“Revisado”, o agente deverá retirar, da BDTD, o anexo que se refere à parte do texto e exportar o anexo que se refere a texto completo. O agente Java está implementado da seguinte forma: ao perceber que houve uma alteração em uma produção, o agente retira a produção e seu anexo, e insere novamente a “nova” produção (entende-se aqui o termo nova como a produção alterada) e seu respectivo anexo. O agente já contempla alterações de “Revisado” para “Não Revisado”.

3.5. Modelagem dos requisitos utilizando UML

Os diagramas nos ajudam a visualizar melhor o panorama e relacionamentos entre os elementos da aplicação, além disso, a modelagem nos ajuda a delimitar o problema, restringindo o foco a um aspecto por vez. Os modelos podem ser expressos em diferentes níveis de precisão. Neste trabalho os diagramas estão com uma visão macro, focando na funcionalidade a ser implementada, já que os sistemas analisados já estão implantados e em uso.

Devido o trabalho ter como foco o levantamento de requisitos de uma nova funcionalidade solicitada pela Biblioteca Universitária da UFMG, foram desenvolvidos os diagramas de caso de uso, diagramas de sequência e diagramas de implantação que serão apresentados a seguir.

A importância do diagrama de caso de uso para este trabalho é sua relevância em mostrar o contexto, a visão do sistema. Os casos de uso são importantes para visualizar, especificar e documentar o sistema.

Já o diagrama de sequência foi escolhido por apresentar uma sequência de eventos que determina o comportamento de um caso de uso. Uma forma de visualizar a sequência de eventos que serão realizados durante o uso do sistema para a situação estudada.

Neste trabalho também é apresentado o diagrama de implantação. Apesar de não ser um dos diagramas mais importantes para a fase de levantamento de requisitos, optou-se por apresentá-lo para melhor entendimento dos aspectos físicos, representando hardware e ambientes de execução de software.

3.5.1. Diagrama de Caso de Uso

Segundo Booch et al. (1998) os diagramas de caso de uso são aplicados para captar o comportamento pretendido do sistema, sem ser necessário especificar como este comportamento será implementado.

A figura 28 demostra, através de um diagrama de caso de uso, a inclusão da funcionalidade de embargo de produções no Sistema Opus.

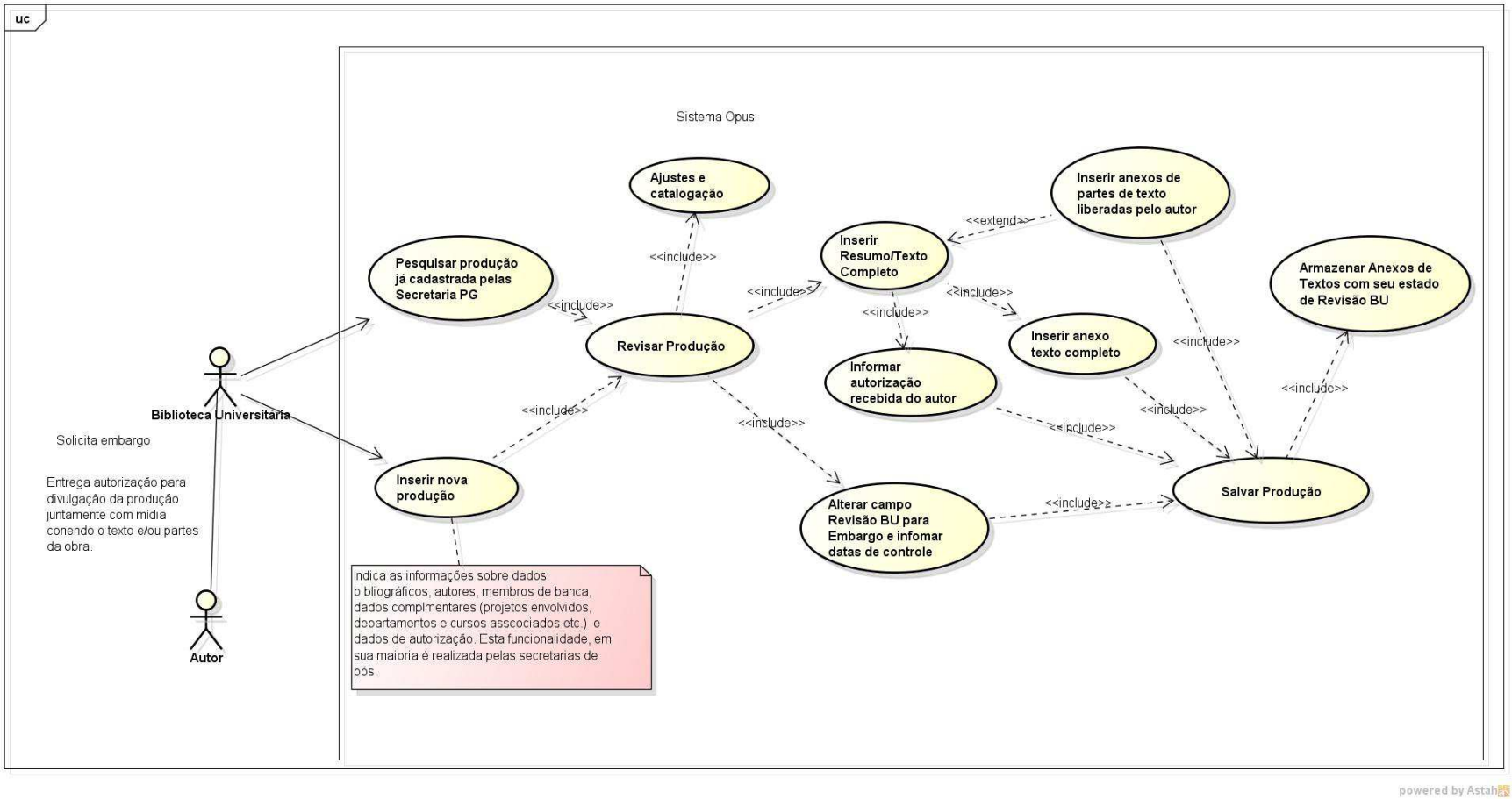

Figura 28 - Diagrama de caso de uso, a inclusão da funcionalidade de embargo de produções no Sistema Opus.

O diagrama de caso de uso demostrando a exportação Opus para o BDTD para as produções embargadas está representando na figura 29.

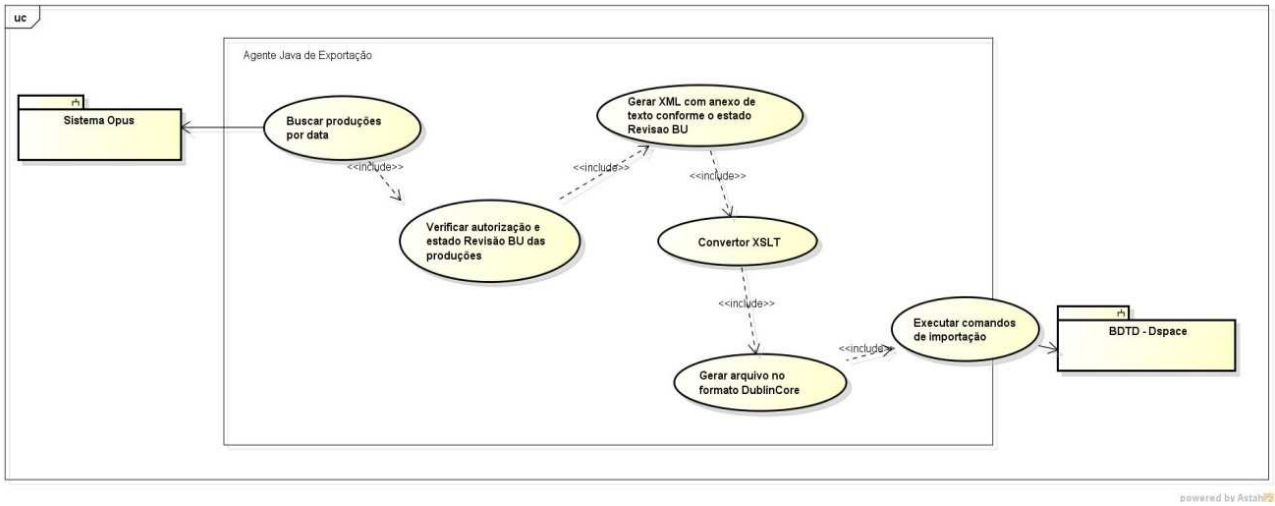

Figura 29 -Diagrama de caso de uso demostrando a exportação Opus para o BDTD

A figura 30 mostra como será a retirada do embargo das produções que se encontrem neste estado no Sistema Opus.

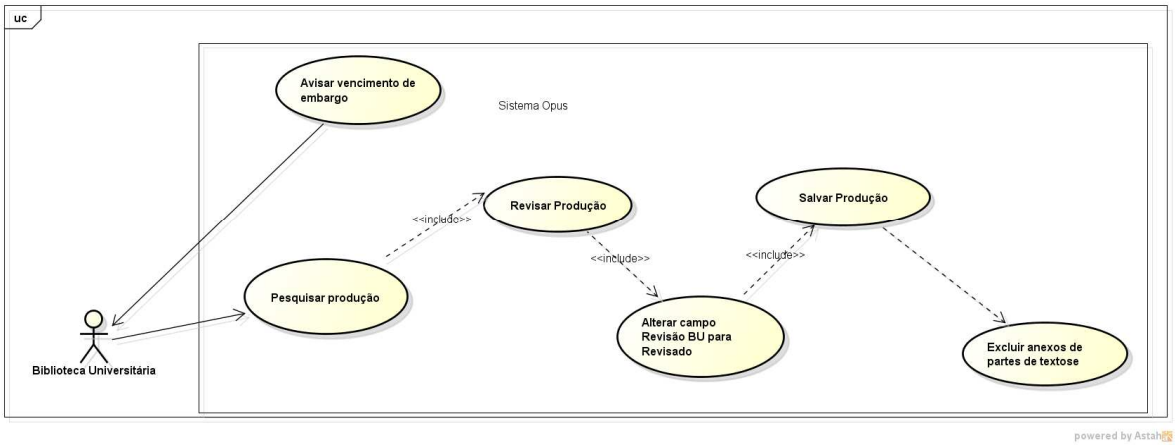

Figura 30 - Retirada do estado de Embargo de produções no Sistema Opus.

O diagrama de sequência enfatiza o tempo de sequência e mostra objetos participando em interações e as mensagens que trocam entre si.

Para cada evento recebido o sistema irá executar uma operação em resposta.

As figuras 31 e 32 apresentam respectivamente, os diagramas de sequência das produções embargadas no Opus e de exportação do Opus para o BDTD

(Dspace).

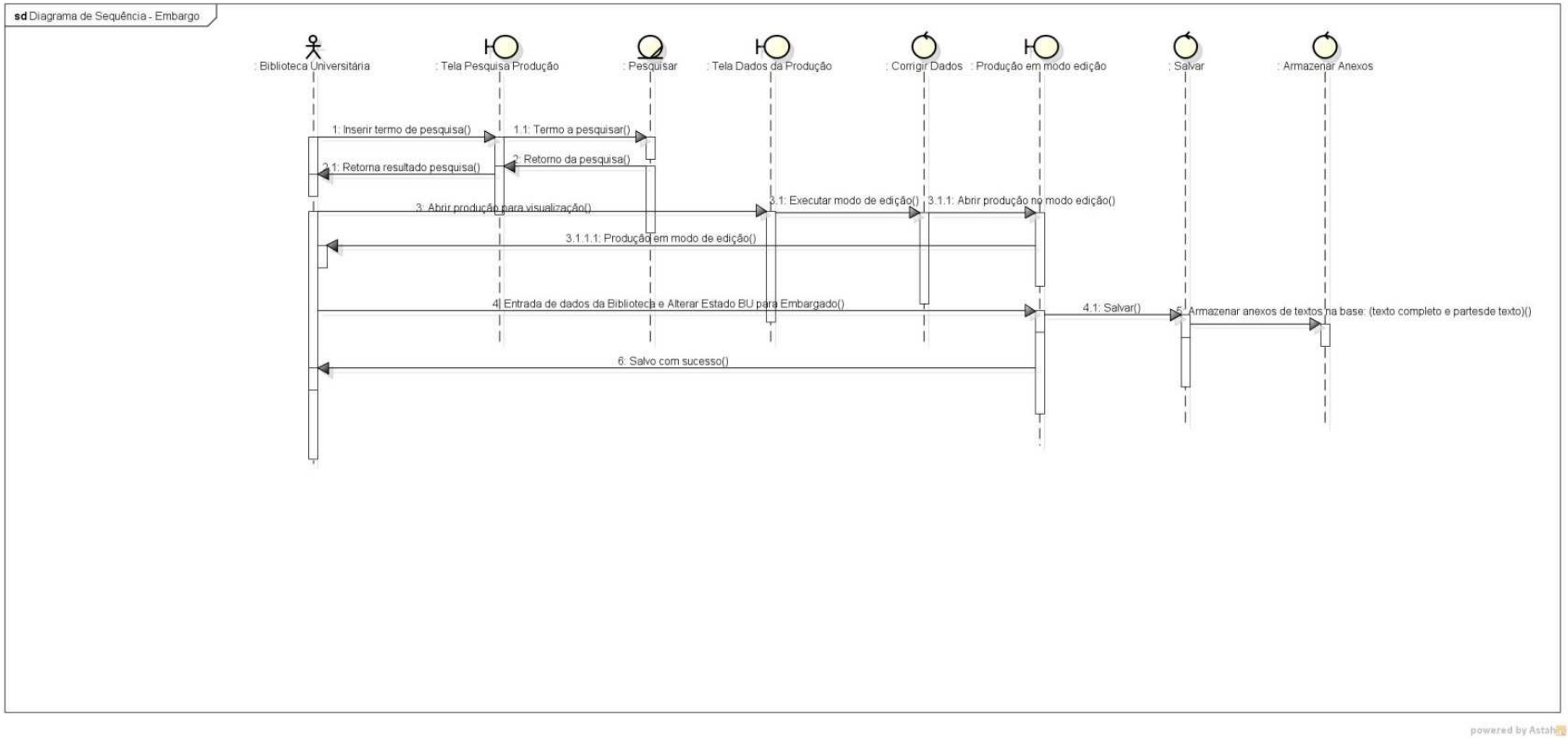

Figura 31 - Diagrama Caso de Uso: Retirada Embargo no Sistema Opus

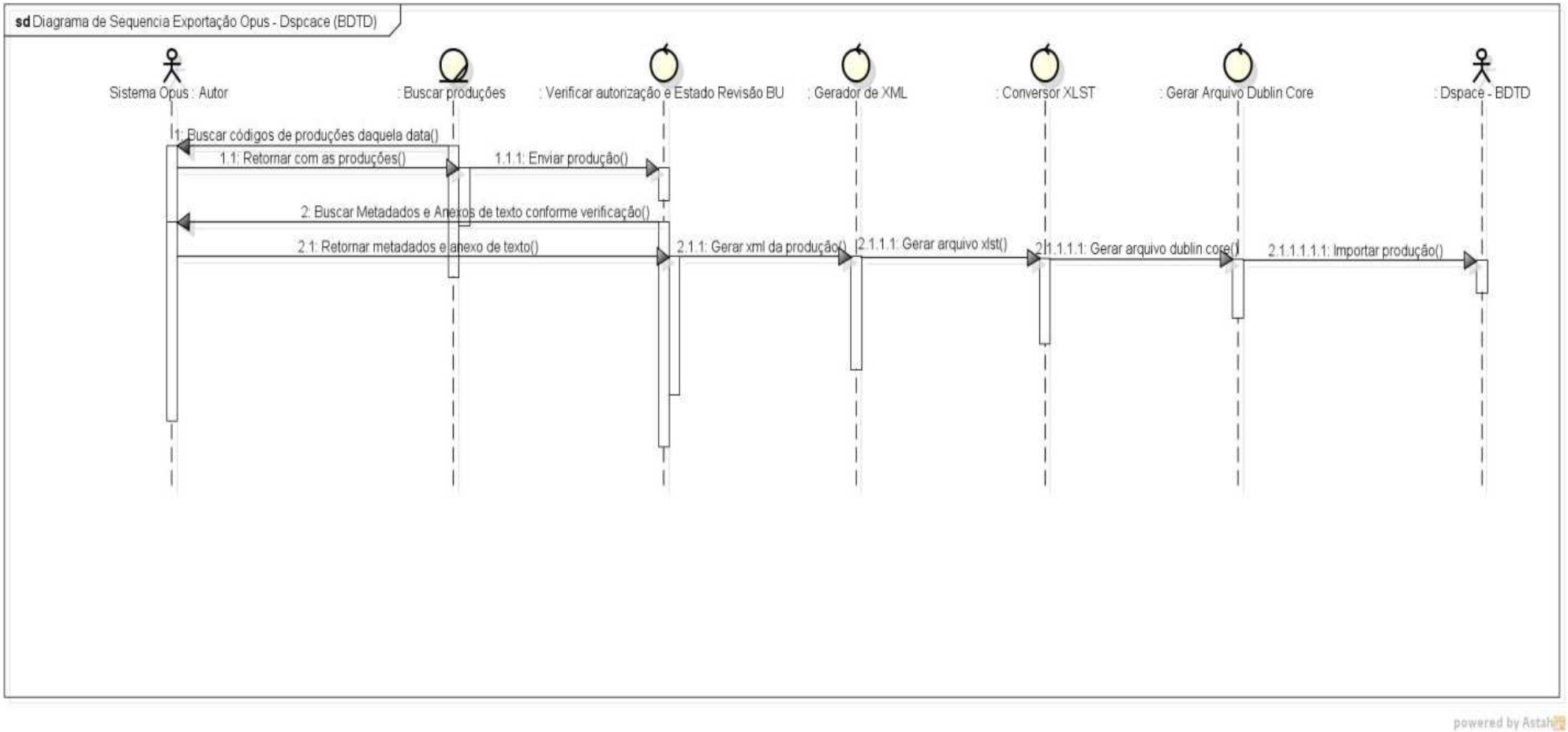

Figura 32 - Diagrama de sequência – exportação de produções embargadas do Sistema Opus para DSpace (BDTD).

A retirada do estado de embargo no Sistema Opus é mostrada pelo diagrama de sequência representado na figura 33.

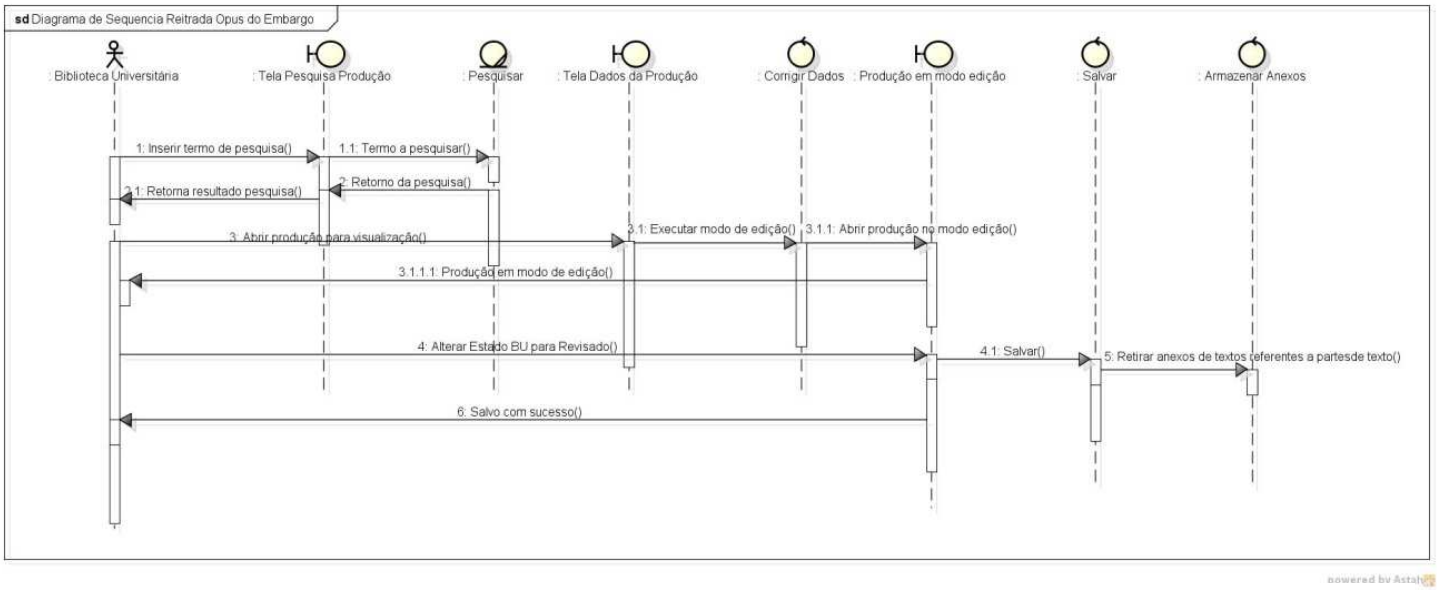

Figura 33 - Diagrama de sequência de retirada do estado de estado de embargo de produções no Sistema Opus.

Os diagramas de casos de uso e de sequência demostrando a exportação de Opus para BDTD, para a situação de retirada do embargo das produções não foram modelados, pois seguirão o mesmo fluxo hoje implementado para alterações entre “Revisado” e “Não Revisado”, contendo apenas a alteração de verificação do campo Estado Revisão BU no documento de anexo, alteração mencionada anteriormente. “Com a alteração do Estado Revisão BU para os valores de “Embargado” para “Revisado” ou” Não Revisado o agente Java de exportação do Opus para a BDTD devem realizar as mesmas regras de negócio já definidas, não é necessário a modelagem pois haverá uma única alteração em seu fluxo, onde comtemplará todas as mudanças de estado da produção.

1.1.1. Diagramas de Implantação

Os diagramas de implantação permitem definir a arquitetura de execução dos sistemas, a arquitetura física, hardware, software e conexões.

Para a implantação do estudo de caso deste trabalho não será necessária alterações em hardware, softwares ou conexões. Mas é interessante mostrar como

hoje está desenhada esta arquitetura. A figura 34 mostra o diagrama de implantação Opus – BDTD.

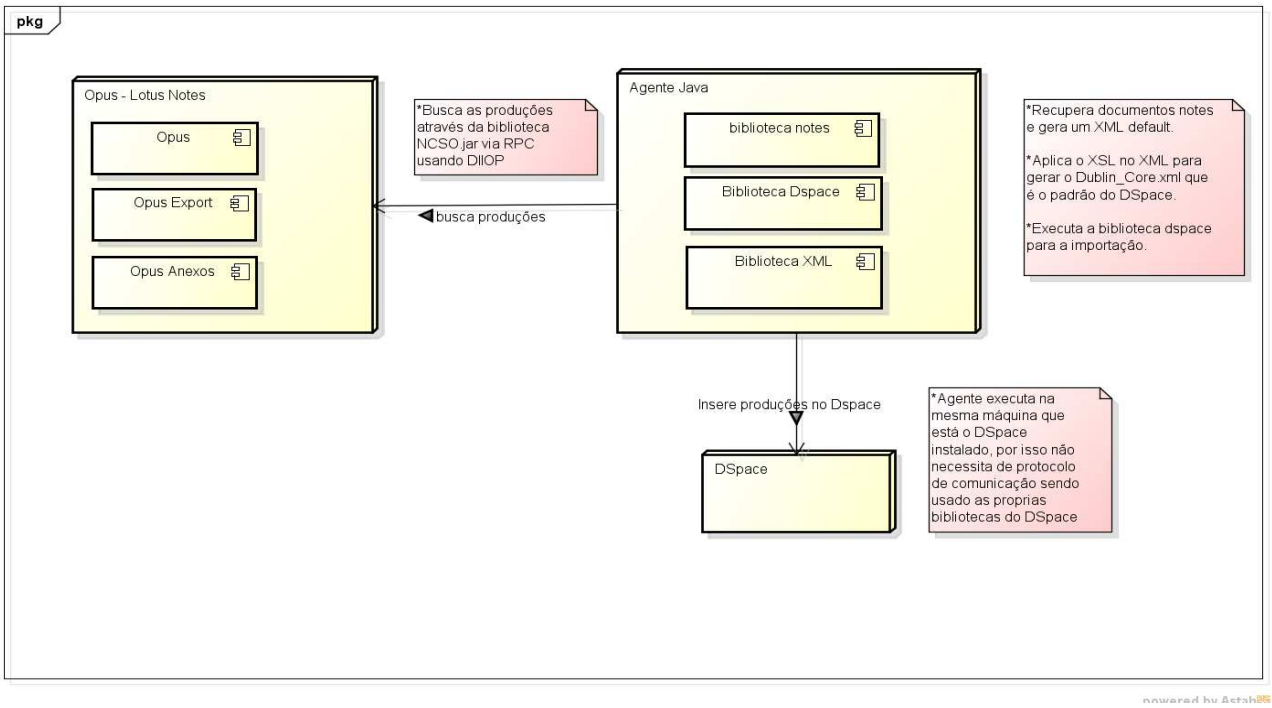

Figura 34 - Diagrama de implantação do Sistema Opus – BDTD.

2. Infra Estrutura

2.1. Orçamento Físico-Financeiro

Neste estudo não se aplicam aquisição recursos físicos ou financeiros, pois não será necessária a compra de máquinas, softwares, hardwares ou licenças para a introdução da atividade de modelagem nas tarefas de levantamento e desenvolvimento das aplicações do LCC.

É importante salientar que, mesmos os analistas da equipe terem embasamento teórico da linguagem de modelagem UML, por trabalharem com a mesma em estudos acadêmicos, propõe-se um treinamento para enriquecimento deste conhecimento, visando uma melhor utilização prática da técnica em suas atividades.

2.2. Equipe

A etapa de levantamento e modelagem de requisitos para implantação da funcionalidade de embargo de produções no Sistema Opus e BDTD contou com os seguintes profissionais do setor de Desenvolvimento do LCC:

- Karina Flaviana Ribeiro - Analista de Requisitos e responsável pelo acompanhamento e revisão da documentação técnica gerada durante todo o processo de análise e desenvolvimento, bem como acompanhamento da codificação no Sistema Opus e apoio nas alterações referentes ao Dspace (BDTD).

A implementação (codificação) dos requisitos levantados será realizada pelos profissionais citados abaixo:

- Bruno Oliveira de Carvalho - Analista de Sistemas responsável pelo Dspace, onde está implementada a BDTD. Todo atual processo de importação/exportação (agente Java) é de sua responsabilidade.
- Rogério Rosendo Lopes - Analista de Sistemas responsável pela codificação no Opus dos requisitos levantados.
- Walison Dias da Silva Analista de Sistemas responsável pela codificação no Opus dos requisitos levantados.

3. Conclusão

Atualmente os setores de desenvolvimento de softwares das Instituições Públicas passaram a ter uma maior percepção da importância da implantação de processos de Engenharia de Requisitos, uma vez que os benefícios trazidos pela mesma são percebidos através da redução de custos e prazos. Além da melhoria da qualidade em termos gerais e satisfação do usuário, resultando em ganhos para toda a comunidade acadêmica.

Pode-se concluir neste trabalho que a utilização da UML como uma linguagem de modelagem foi importante para melhor visualização do projeto, proporcionando uma comunicação mais clara concisa entre analistas e usuários. Isso porque ao apresentar diagramas, simplificamos a realidade, o que permite a diminuição de ambigüidades, entendimentos incorretos e até contraditórios.

O objetivo inicial foi a utilização da UML como forma de visualizar e especificar os requisitos. Atualmente a funcionalidade usada como caso de estudo deste trabalho encontra-se em desenvolvimento, e os diagramas da UML desenvolvidos neste trabalho estão sendo usados na fase de codificação da mesma. O documento de Especificação Técnica não consta como anexo deste trabalho porque o mesmo ainda não foi finalizado. O documento encontra-se em fase de construção, pois conforme dito anteriormente, durante toda a codificação este documento é passível de alterações. Ao final, os diagramas desenvolvidos contarão neste documento de Especificação Técnica.

Vale ressaltar a que a introdução de novos métodos e técnicas em uma organização é um processo que requer a participação e engajamento da equipe.

Não existe um processo ideal capaz de ser utilizado em qualquer projeto. Sempre é necessário realizar modificações, em função de características de sistemas, pessoas e tecnologia, dentre outros.

Como tema para trabalhos futuros propõe-se a continuação da formulação dos demais diagramas da UML para este mesma funcionalidade, bem como uma apresentação para a equipe, para análise geral, por parte da mesma, dos ganhos da introdução da linguagem de modelagem nas suas atividades de desenvolvimento.

Referências

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **The unified modeling language user guide**. 1. ed. California: Addison-Wesley, 1998, 512 p.

ANÁLISE orientada a objetos. Goianésia, 2011. 81 p. Disponível em http://pt.scribd.com/doc/98608144/Apostila-UML, acessado em 17 de março de 2013.

BROOKS JÚNIOR, Frederick P. No silver bullet: essence and accidents of software engineering. *In:* **`INFORMATION processing 86'.** North Holland: Elsevier Science, 1996, 1069-1076 p.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software:** fundamentos métodos e padrões. 2. ed. Rio de Janeiro: Editora: LTC, 2003, 602 p.

BORGES JÚNIOR, Gerson Rodrigues. **Engenharia de software:** metodologias de desenvolvimento de sistemas. Brasília; 2010. [apostila do Curso Sistemas de Informação - Centro Universitário do Centro Oeste – UNIDESC]. Disponível em: http://www.professorgersonborges.com.br/site/pdf/apostila/Engenharia/Intru_Engenharia_Software.pdf , acessado em 17 de março de 20013.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering:** processes and techniques. Chichester, UK : John Wiley & Sons, 1998. 282 p.

LARMAN, Graig. Utilizando UML e Padrões: **Uma introdução a analise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. Tradução Rosana T. Vaccare Braga, Paulo Cesar Masiero, Rosângela Ap. Delosso Penteado, Fernão Stella R. Germano. 3. ed. Porto Alegre: Bookman, 2007,696 p.

LEITE, Cláudia; Júlio César Sampaio. Indicadores para a Gerência de Requisitos. In ENTRADA **Anais do WER2003 - Workshop em Engenharia de Requisitos**, 6, São Paulo; 27,28 de Novembro de 2003.

LEITE, Jair C. **Análise e especificação de requisitos:** engenharia de software**. 2000.** Notas de Aula. Disponível em:<http://www2.dem.inpe.br/ijar/EngSofAnalEspc.html>, acessado em 17 de março de 20013.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.0:** do conceitual a implementação. 2.ed. Rio de Janeiro: Brasport. 2004, 284p.

MELLO, João Alexandre Bonin de. Uma metodologia para Engenharia de Requisitos para pequenas equipes de desenvolvimento de software**. Revista de Ciências Empresariais da UNIPAR**, Paraná, v. 6, n. 1, p. 97-111, jan./jun. 2005. Disponível em: <http://revistas.unipar.br/empresarial/article/view/302/273>, acessado em 17 de março de 20013.

PETERS, James F.; PEDRICZ, Witold. **Engenharia de software: teoria e prática**, Rio de Janeiro: Campus, 2001, 602 p.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo, Brasil: Makron Books do Brasil Editora Ltda, 1995, 1056 p.

SOMMERVILLE, Ian. **Engenharia de software**; Tradução Selma Shin Shimizu, Reginaldo Arakaki, Edílson de Andrade Barbosa; revisão técnica Kechi Kirama. 8. ed. São Paulo, Brasil: Pearson Addison Wesley, 2007, 552 p.

THAYER, Richard H.; DORFMAN Merlin**. Software Requirements Engineering**. 2. Ed., Alamitos, California Wiley -IEEE Computer Society,1997. 549 p.

VARGAS, Thânia Clair de Souza. **A história de UML e seus diagramas**. Disponível em <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/artigo.tcc.pdf> acessado em 17 de março de 20013.