INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE, *CAMPUS* PASSO FUNDO
CURSO SUPERIOR EM TECNOLOGIAS EM SISTEMAS PARA INTERNET

GIUSEPPE MATHEUS BAIRROS PEREIRA

ESTUDO DA ENGENHARIA DE SOFTWARE E IMPLEMENTAÇÃO DE UM PROTÓTIPO DE APROVEITAMENTO DE DISCIPLINAS DO INSTITUTO FEDERAL SUL-RIO-GRANDENSE, *CAMPUS* PASSO FUNDO

Élder Bernardi

PASSO FUNDO, 2013

GIUSEPPE MATHEUS BAIRROS PEREIRA

ESTUDO DA ENGENHARIA DE SOFTWARE E IMPLEMENTAÇÃO DE UM PROTÓTIPO DE APROVEITAMENTO DE DISCIPLINAS DO INSTITUTO FEDERAL SUL-RIO-GRANDENSE, *CAMPUS* PASSO FUNDO

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Msc. Élder Bernardi

Passo Fundo, 2013

GIUSEPPE MATHEUS BAIRROS PEREIRA

ESTUDO DA ENGENHARIA DE SOFTWARE E IMPLEMENTAÇÃO DE UM PROTÓTIPO DE APROVEITAMENTO DE DISCIPLINAS DO INSTITUTO FEDERAL SUL-RIO-GRANDENSE, *CAMPUS* PASSO FUNDO

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Orientador Prof. Me. Élder Fonseca Bernardi

_____

Convidado Prof. Me. Evandro Miguel Kuszera

_____

Convidado Prof. Me. Rafael Marisco Bertei

_____

Prof. Me. Evandro Miguel Kuszera

Coordenação do Curso

PASSO FUNDO, 2013

Aos meus pais, Paulo e Mari, que sempre me ajudaram e instruíram.
À minha noiva Aline, pelo amor e dedicação.

AGRADECIMENTO

Agradeço a todos que me apoiaram, mas em especial um grupo de colegas e guerreiros que fizeram parte dessa jornada, Thomaz Xavier, José Alcibíades Almeida, Josiane Almeida, Luís Augusto Rodriguez, Nórton Mattiello Vanz, Samuel Silvestrin, Thiago Vargas Acunha, Vinícius Castelani Reck, Vinícius Pierdoná Lima, há estes um muito obrigado. Agradeço a todos os professores e funcionários do Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense, Campus Passo Fundo, mas em especial um muito obrigado ao meu orientador Élder Bernadi, à professora Anúbis Graciela Rosseto e a assistente em assuntos administrativos Roseli Moterle, a nossa querida Rosi.

RESUMO

Este trabalho apresenta todos os passos decorridos para o desenvolvimento de um protótipo com as funcionalidades básicas relacionadas ao aproveitamento de disciplinas no Instituto Federal Sul-Rio-Grandense, Campus Passo Fundo, de modo que possa ser retomado por colaboradores e futuramente implantado para uso no Instituto. Para isso implementou-se um sistema, na linguagem java, com os frameworks Hibernate e JSF, executável em um container Tomcat ou Glassfish com banco de dados Mysql.

Palavras-chave: Java Web, Engenharia de Requisitos, Frameworks, Modelagem de dados, Projeto, Codificação.

ABSTRACT

This work presents all steps of the development of a prototype with basic functionalities related to the use of disciplines at the Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, so that it can be taken up by employees and future deployed for use at the Institute . For this we have implemented a system in Java language, with the Hibernate frameworks and JSF executable into a container Tomcat or Glassfish with MySQL database.

Keywords: Java Web Requirements Engineering Frameworks, Data modeling, design, coding.

LISTA DE FIGURAS

Figura 1 - Modelo Cascata 15
Figura 2 - Modelo Cascata com Realimentação 17
Figura 3 - Ciclo de vida em Espiral 18
Figura 4 - Ciclo de Vida em Espiral II 19
Figura 5 - Interação do projeto de software e análise 22
Figura 6 - Cronograma Simples 28
Figura 7 – Exemplo de Diagrama de Casos de Uso 33
Figura 8 - Exemplo de Diagrama de Classes 34
Figura 9 - Diagrama de Pacotes 35
Figura 10 - Diagrama de Sequência 36
Figura 11 - Diagrama de Comunicação 37
Figura 12 - Diagrama de Máquina de Estados 38
Figura 13 - Diagrama de Atividade 39
Figura 14 - Diagrama de Implantação 40
Figura 15 - Diagrama de Estrutura Composta 41
Figura 16 – Resumo dos Diagramas 42
Figura 17 - Ciclo de Vida do Projeto 47
Figura 18 - Diagrama Casos de Uso 65
Figura 19 - Diagrama de Classes do Projeto 67
Figura 20 - Diagrama de Atividade Solicitação 68
Figura 21 - Diagrama de Sequência Solicitação 69
Figura 22 - JSF - Representação em alto nível 75
Figura 23 - Fluxo Alunos 77
Figura 24 - Fluxo Matrículas 79
Figura 25 - Fluxo Cursos 81
Figura 26 - Fluxo Disciplinas 82
Figura 27 - Fluxo Solicitações 84
Figura 28 - Menu do Usuário 86
Figura 29- Fluxo Usuários 87

LISTA DE TABELAS

Tabela 1 - Entrevistas 45
Tabela 2 – Enunciado do Trabalho – Visão Geral 48
Tabela 3 - Enunciado do trabalho – Funções 49
Tabela 4 - Enunciado do trabalho – Necessidades e benefícios 50
Tabela 5 - Enunciado do trabalho – Requisitos Funcionais 51
Tabela 6 - Descrição dos Atores 54
Tabela 7 - Casos de Uso 55
Tabela 8 - Caso de Uso 1 – Gestão de Usuários 56
Tabela 9 - Caso de Uso 2 - Gestão de Solicitações 57
Tabela 10 - Caso de Uso 3 - Emissão de Relatório 58
Tabela 11 – Caso de Uso 4 - Operação de Criação de Solicitação 59
Tabela 12 - Caso de Uso 5 Encerramento de Solicitação 61
Tabela 13 - Caso de Uso 6 - Alteração de Matricula 62
Tabela 14 - Caso de Uso 7 - Comunicação com Aluno 63
Tabela 15 - Requisitos Atendidos Aluno 76
Tabela 16 - Requisitos Atendidos Matrículas 78
Tabela 17 - Requisitos Atendidos Curso 80
Tabela 18 - Requisitos Atendidos Disciplinas 81
Tabela 19 - Requisitos Atendidos Solicitações 83
Tabela 20 - Requisitos Atendidos Usuários 84

SUMÁRIO

1 Introdução 6

Objetivos do trabalho 9

1.1 Objetivos Específicos 9

1.2 Organização do Texto 9

2 Referencial Teórico 11

2.1 Engenharia de Software 11

2.1.1 Definições de Software 11

2.1.2 Processos e ciclos de vida 13

2.1.3 Modelos ciclos de vida 14

2.1.3.1 Cascata 14

2.1.3.2 Espiral 18

2.1.3.3 Prototipagem 20

2.1.3.4 Ciclo de Vida do Projeto 21

2.1.2 Engenharia de Requisitos 21

2.1.2.1 Requisitos Funcionais 24

2.1.2.2 Requisitos Não Funcionais 25

2.1.2.3 Gestão de Requisitos 26

2.2 Projeto da análise 27

2.2.1 Cronograma do Projeto 27

2.3 Modelagem de Dados 29

2.3.1 UML 2.0 30

2.3.2 Diagramas da UML 2.0 31

2.3.2.1 Diagrama Casos de Uso 31

2.3.2.2 Diagrama de Classes 33

2.3.2.3 Diagrama de Objetos 34

2.3.2.4 Diagrama de Pacotes 35

2.3.2.5 Diagrama de Sequência 36
2.3.2.6 Diagrama de Comunicação 36
2.3.2.7 Diagrama de Máquina de Estados 37
2.3.2.8 Diagrama de Atividade 38
2.3.2.9 Diagrama de Visão Geral de Interação 38
2.3.2.10 Diagrama de Componentes 39
2.3.2.11 Diagrama de Implantação 40
2.3.2.12 Diagrama de Estrutura Composta 40
2.3.2.13 Diagrama de tempo ou de temporização 41
2.3.3 Resumo dos diagramas 42
3 Modelagem do Sistema 44
3.1 Ciclo de Vida do Projeto 47
3.1 Análise de Requisitos 47
3.1.1 Tabela Visão Geral 48
3.1.2 Tabela Funções 49
3.1.3 Tabela de Necessidades e Benefícios 50
3.1.4 Tabela de Requisitos 51
3.1.5 Tabela de Atores 54
3.2 Casos de Uso 55
3.3 Documentações do Caso de Uso 56
3.4 Diagrama de Casos de Uso 64
3.5 Diagramas do Projeto 65
3.5.1 Diagrama de Classes 66
3.5.2 Diagrama de Atividade 67
3.5.3 Diagrama de Sequência 68
4 Implementação 71
4.2 Padrões de Projeto 71

4.2.1 Singleton 71
4.3 Tecnologias 72
4.3.1 Linguagem Java 72
4.3.2 Apache Tomcat 73
4.3.3 Servidor de Dados Mysql 73
4.3.4 Hibernate 73
4.3.5 JSF Framework 74
4.4 Requisitos Atendidos 75
4.4.1 Requisitos atendidos da entidade Aluno 75
4.4.2 Requisitos atendidos da entidade Matrículas 78
4.4.3 Requisitos atendidos da entidade Cursos 80
4.4.4 Requisitos atendidos da entidade Disciplinas 81
4.4.5 Requisitos atendidos da entidade Solicitações 83
4.4.6 Requisitos atendidos da entidade Usuários 84
4.4.7 Menu do Usuário 85
5 Considerações Finais 89
6 Referências 91
7 Anexos 93

1 Introdução

Com a evolução da internet a humanidade foi se adaptando ao mundo virtual para acompanhar essa tecnologia. A Internet surgiu como uma maneira nova de comunicação.

Com o surgimento do conceito de navegação, navegar dentre as informações virtualmente, ocorreu à criação de serviços e ferramentas capazes de controlar e processar informações, exemplo disto seria contas de e-mail e cartórios online.

Um exemplo que pode ser notado no nosso cotidiano é o crescente número de usuários de contas de e-mail (Eletronic mail, Correio Eletrônico). O sistema e a estrutura física de envio de cartas não foi substituído, mas naturalmente as pessoas começaram a se comunicar não mais pelas cartas tradicionais, manuscritas e enviadas a empresa responsável, tornando-se mais prático com o surgimento dos e-mails, esse sistema entrega o e-mail (carta virtual, ou carta eletrônica) ao destinatário, quase que instantaneamente e não passa por nenhuma intervenção humana.

O importante é entender que sistemas podem auxiliar e aperfeiçoar um serviço. Considerando isso, este trabalho irá analisar o ambiente, planejar, projetar e programar um sistema que ajudará o Instituto Federal Sul-rio-grandense (IFSUL), Campus Passo Fundo, a gerenciar as solicitações de aproveitamento de disciplinas.

O Instituto Federal Sul-Rio-Grandense, Campus Passo Fundo, passa por algumas dificuldades ao lidar com as solicitações de aproveitamento de disciplinas. O maior problema pode ser resumido pela falta de tratamento dos documentos do aluno.

O aluno solicita aproveitamento de disciplina na CORAC (Coordenação de Registros Acadêmicos), setor responsável no Instituto. Nesta solicitação a documentação é encaminhada para o coordenador do curso junto com a ficha de solicitação do aluno.

Todo esse material referente à solicitação de aproveitamento é analisado pelos coordenadores dos cursos da área de informática, Curso superior em Tecnologia em Sistemas para Internet e Curso Técnico em Informática.

Com apenas esse material, o coordenador não tem informações sobre quais cadeiras o aluno quer aproveitar e nem o controle da carga horária, então ele deve analisar cada disciplina cursada e suas cargas horárias, comparando-as com as disciplinas do Instituto.

Devido ao volume de documentos e ausência de uma ferramenta ou sistema, há sobrecarga de trabalho nos coordenadores e com isso a possibilidade de falhas, pois o mesmo aluno pode reenviar essa mesma documentação várias vezes a fim de obter aproveitamento.

Como se trata de um software produto, realizaremos desde análise de requisitos até implementação parcial com uso de técnicas de levantamento de requisitos, será feita a descrição do problema a fim de oferecer uma solução, utilizando de técnicas de engenharia de software que serão definidas e estudas no projeto.

Para a codificação, estudaremos tecnologias de programação que devem estar em concordância com as técnicas de análise, pois a importância de um estudo do problema implica na eficiência e qualidade do software final. Para medir a qualidade e eficiência, serão aplicadas metodologias de testes e modificações apoiadas em um ciclo de vida.

Portanto, torna-se necessário o estudo de uma solução a fim de tornar esse sistema mais rápido e eficaz, com mais segurança e não estar sujeito a falhas, para que haja o controle dessas informações, o processo não possui nenhum tipo de controle e desenvolveremos um sistema-web protótipo para melhorá-lo intitulado SAD 1.0 (Sistema de Aproveitamento de Disciplinas).

Objetivos do trabalho

Neste trabalho, buscou-se desenvolver um protótipo de controle de aproveitamento de disciplinas do Instituto Federal Sul-Rio-Grandense, Campus Passo Fundo. Para isso estudou-se conceitos de engenharia de softwares, análise de requisitos, modelagem de sistemas e implementação.

1.1 Objetivos Específicos

- Realizar um estudo teórico da engenharia de software.
- Identificar os problemas do atual sistema.
- Listar os requisitos funcionais a serem atendidos.
- Estudar as metodologias de engenharia de requisitos.
- Desenvolver um protótipo.

1.2 Organização do Texto

Este documento está organizado da seguinte maneira: Capítulo 2 apresenta o referencial teórico, o estudo da engenharia de software, projeto, modelagem de dados e UML 2.0. No Capítulo 3 apresenta a modelagem do sistema, diagramas, casos de uso, com base nos estudos do Capítulo 2 e análise de requisitos. No Capítulo 4 apresenta as tecnologias usadas, padrões de projeto, ciclo de vida do projeto e os requisitos atendidos. Finalmente apresenta-se as considerações finais e a documentação dos casos de uso.

2 Referencial Teórico

Neste capítulo teremos como foco o estudo da engenharia de software, modelagem de dados e a UML 2.0, a fim de adquirirmos conhecimento para a implementação e documentação do sistema.

2.1 Engenharia de Software

A engenharia de software trata como devemos desenvolver um software, é uma disciplina que integra métodos, ferramentas e procedimentos para o desenvolvimento de softwares (PRESSMAN, 2005, p.17).

2.1.1 Definições de Software

A definição de software muitas vezes é bastante distorcida, costuma-se associar software com programas de computador. Não está totalmente errado afirmar isso, porém um software deve ser entendido como a união ou o conjunto de várias partes, como lógica de negócio, configurações, programas, *plug-ins*, tutoriais e *tours*, referência na web para documentação e, dependendo do software, muito mais subconjuntos. Os softwares podem ser tratados como produtos e vistos de duas maneiras diferentes, genéricos e personalizados (Pressman, 2010, p. 13).

Os softwares genéricos são produzidos por alguma empresa de desenvolvimento de software e vendidos no mercado de forma que vários clientes comprem e utilizem por conta própria. Um exemplo seria os antivírus que são disponibilizados inicialmente como estratégia de mercado com licença gratuita para uso domiciliar a fim da divulgação e com versões pagas para uso empresarial (Pressman, 2010, p. 13).

Os personalizados também são produzidos por alguma empresa de desenvolvimento de software, mas criados tendo como base o interesse do cliente.

O cliente procura a empresa e solicita a criação de um sistema específico para solução de um problema específico descrito por ele. Esse sistema só irá funcionar corretamente no ambiente daquele cliente, não é genérico, portanto. Como exemplo, pode-se citar os softwares de mercados, farmácias e instituições (Pressman, 2010, p. 14).

De acordo com Pressman (2010, p.13),

> O software tornou-se o elemento-chave na evolução de sistemas e produtos baseados em computador, e uma das tecnologias mais importantes em todo o mundo. Ao longo dos últimos 50 anos, o software evoluiu de um ferramental especializado em solução de problemas e análise de informações para um produto de indústria. Mas ainda temos problemas na construção de software de alta qualidade de prazo e dentro do orçamento. O software – programas, dados e documentos – dirigidos a uma ampla variedade e tecnologias e aplicações, continua a obedecer a uma série de leis que permanecem as mesmas há cerca de 30 anos. O intuito de engenharia de software é fornecer uma estrutura para a construção de software com alta qualidade.

Com a evolução dos softwares e o crescimento do mercado, serviços e ambientes virtuais, necessitou-se uma evolução visando a melhoria da qualidade e do desempenho dos softwares. A engenharia de software surgiu a fim de auxiliar no planejamento e desenvolvimento de softwares priorizando sua eficiência, segurança, qualidade e estabilidade. Nesse sentido, dada a exigência de qualidade final do produto do mercado atual, a aplicação de técnicas de engenharia de software torna-se indispensável para obtenção de bons resultados na criação de softwares.

Bauer (*apud* PRESSMAN, 2010, p.17) afirma que Engenharia de Software é “a criação e utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais.”

Sommerville (2007, p. 5) entende que “A engenharia de software é uma disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios inicias de especificação do sistema até sua manutenção.”

Com estudo de autores como Pressman e Sommerville, é possível entender que a engenharia de software abrange todo o projeto de maneira estratégica, apontando passos a serem seguidos e métodos a serem adotados para atingir um resultado final de maior qualidade. Cada etapa deve ser executada com devida competência. Cada um dos autores defende uma maneira de organizar e executar esses passos. Seguindo orientação das obras escritas por Pressman, Sommerville e Pádua, será possível definir os processos e o clico de vida do sistema.

2.1.2 Processos e ciclos de vida

O conceito de processo dentro da engenharia de software é muito abstrato, pode ser imaginado como uma sequência de passos previsíveis a serem seguidos para obter um resultado já esperado e definido. Pode ser definido como um conjunto de ações e atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços. “Um conceito diferente seria que o processo seria uma espécie de receita, como a de um bolo, em que a empresa, com conceitos abstratos, o cozinheiro, deve seguir e confeccionar o bolo, o sistema.” (PÁDUA, 2011, p. 88).

O processo pode satisfazer ou não o ciclo de vida de um sistema, depende de suas características. Esse processo pode satisfazer uma etapa de um aplicativo Desktop, mas pode se tornar inútil no desenvolvimento de um site (PÁDUA, 2011, p. 88).

Um exemplo de processo dentro de uma fábrica de software seria a etapa de testes do sistema, é uma sequência de atividades executadas que produzem um resultado concreto e sólido que é analisado e então definido o futuro do sistema, desde modificações, entrega e até mesmo descontinuação. Isso varia muito de projeto para projeto (PÁDUA, 2011, p. 89).

Um ciclo de vida define como serão as sequências e a ordem exata com que os processos do projeto serão executados. Existem muitos tipos de ciclos de vida e cada um pode ser usado e aplicado em diferentes situações e casos. Nesse caso,

estudar-se-á os mais comuns e usados, alguns deles são evoluções de anteriores. Ciclos de vida ajudam na organização, manutenção e execução do projeto (PÁDUA, 2011, p. 89).

Consultando conceitos de ciclos de vida e processos, Pádua (2011, p. 90) afirma que:

> Em engenharia de software, processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contração de software. Podem-se também definir subprocessos para cada um desses; por exemplo, um processo de desenvolvimento abrange subprocessos de determinação dos requisitos, analise, desenho, implementação e testes. Em um processo de desenvolvimento de software, o ponto de partida para a arquitetura de um processo é a escolha de um modelo de ciclo de vida.
>
> O ciclo de vida mais caótico é aquele que pode ser chamado de “codifica-remenda”. Partindo apenas de uma especificação (ou nem isso), os desenvolvedores começam imediatamente a codificar, remendando à medida que os erros vão sendo descobertos. Nenhum processo é definido e infelizmente, é provavelmente o ciclo de vida mais usado.

2.1.3 Modelos ciclos de vida

Como a variedade de sistemas, softwares e serviços é muito grande, não existe um ciclo de vida padrão para uso. Com isso surgiu os modelos de ciclo de vida, alguns mais detalhistas, outros com foco nos testes e análise, outros com foco na segurança e manutenção. A seguir, são mencionados os mais usados e aprovados pelo mercado. Eles tratam a maneira com que os processos adotados pelo projeto serão executados, avaliados e descritos.

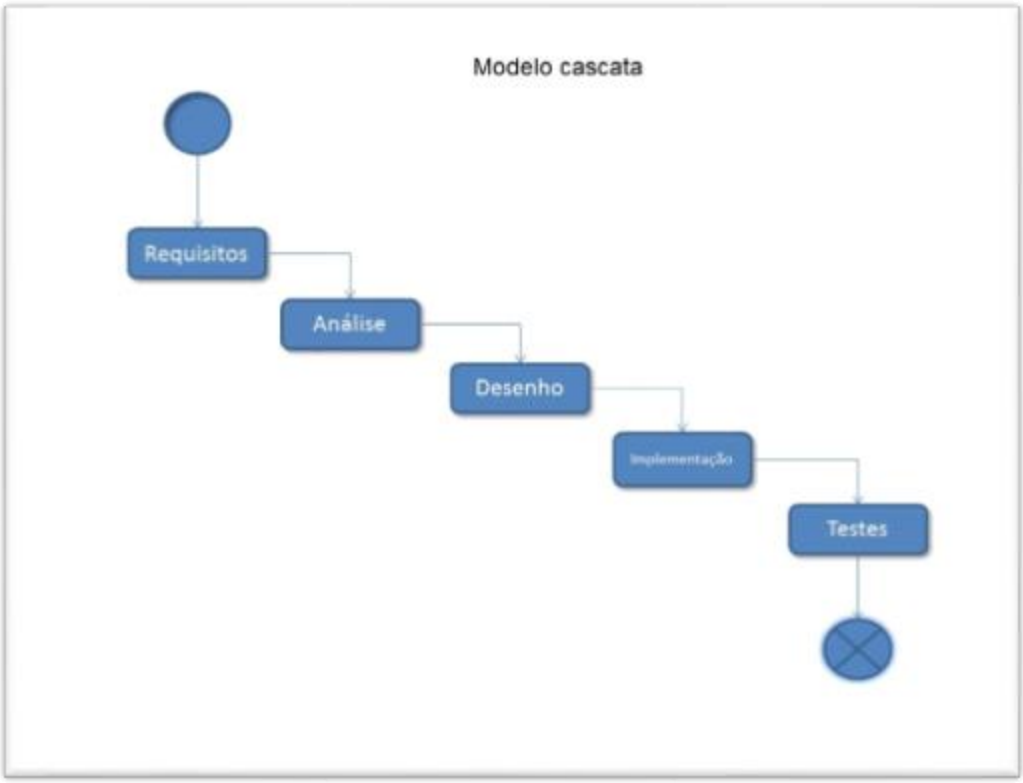
2.1.3.1 Cascata

Esse modelo também é conhecido como ciclo de vida clássico, organiza os processos de maneira sequencial e sistemática. Pode-se entender que se trata de um sistema burocrático onde as etapas e processos devem ser muito bem

entendidos e executados, pois problemas em etapas anteriores não são previstos e tratados.

A Figura 1 (Pádua, 2011 pag. 92) ilustra o modelo cascata e as setas indicam o fluxo das informações. Cada etapa deve ser executada antes da próxima ser iniciada, seguindo assim até a entrega do resultado final.

Figura 1 - Modelo Cascata



Fonte: Pádua, 2011, p. 92.

O Modelo de vida cascata possui alguns problemas, como a falta de tratamento de etapas anteriores e pouca interação com o cliente, que só recebe o produto final do projeto.

Pressman (2010, p. 39) fala sobre alguns problemas desse modelo e faz criticas sobre sua eficácia:

> O modelo em cascata é o paradigma mais antigo da engenharia de software. No entanto, nas duas últimas décadas, a crítica a esse modelo de

> processo tem provocado, mesmo em seus mais ardentes adeptos, questionamentos sobre sua eficácia [HAN95]. Entre os problemas que são algumas vezes encontrados quando o modelo em cascata é aplicado estão:
>
> Projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Apesar de o modelo linear poder acomodar a iteração, ele o faz indiretamente. Como resultados, as modificações, podem causar confusão à medida que a equipe de projeto prossegue.
>
> Em geral, é difícil para o cliente estabelecer todos os requisitos explicitamente. O modelo em cascata exige isso e tem dificuldade de acomodar a incerteza natural que existe no começo de muitos projetos.
>
> O cliente precisa ter paciência. Uma versão executável dos(s) programa(s) não vai ficar disponível até o período final do intervalo de tempo do projeto. Um erro grosseiro pode se tornar desastrosos se não for detectado até que o programa executável seja revisto.

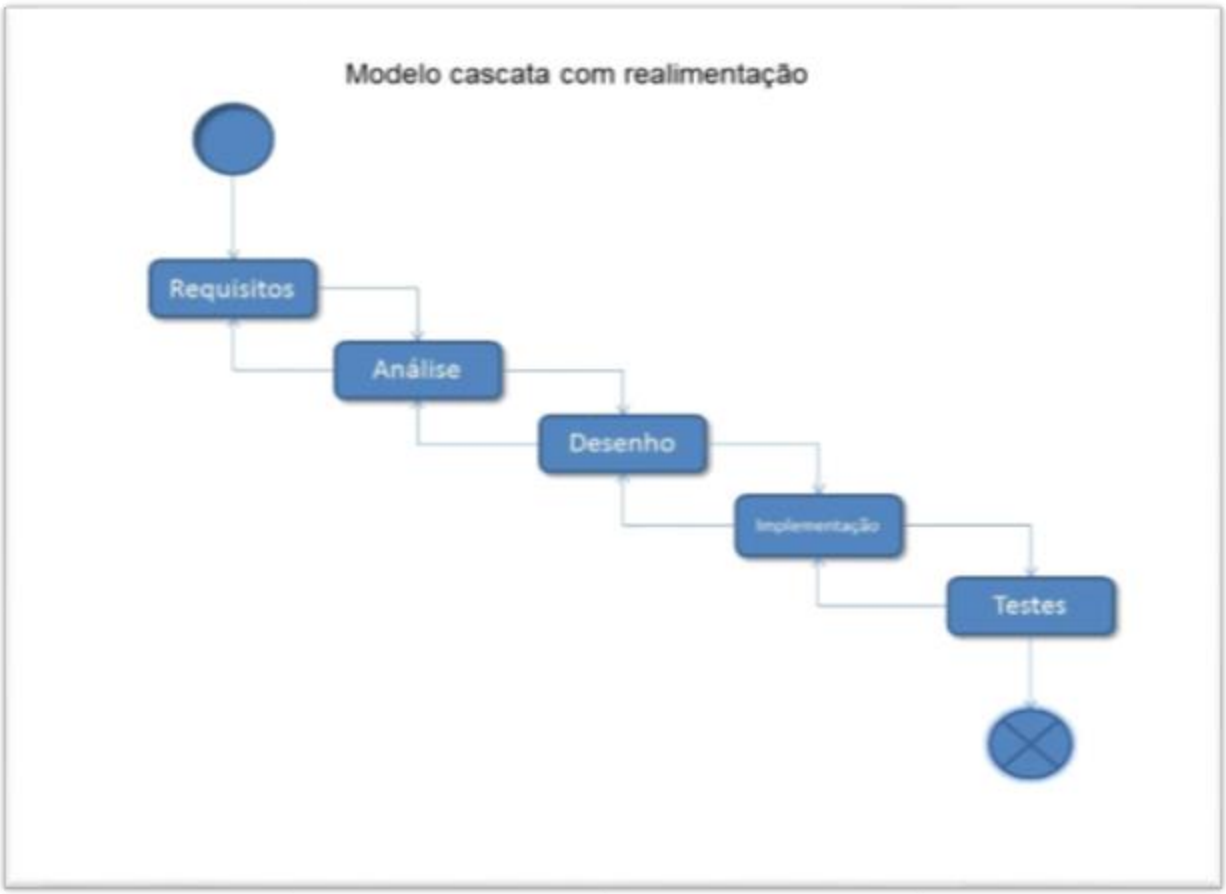
Com o estudo da aplicação prática desse modelo foi notado que é preciso realizar alterações, revisões e correções de etapas anteriores, então surgiu uma variante de do modelo cascata, chamada de Modelo cascata com realimentação (Figura 2). Essa modificação permite retomar processos e etapas anteriores caso necessite alteração, mas consequentemente torna difícil a gestão do projeto devido a esses fluxos de informações.

Segundo Pádua (2011, p. 93), o modelo cascata é adequado a projetos de pequena duração:

> Na prática, é sempre necessário permitir que, em fases posteriores, haja revisão e alteração de resultados das fases anteriores. Por exemplo, os modelos de documentos de desenho podem ser alterados durante à implementação, à medida que os problemas vem sendo descobertos. Uma variante que permite superposição entre fases e realimentação de correções é um modelo mais realista, com realimentação entre fases, chamado de Cascata com realimentação. A realimentação entre fases torna difícil gerenciar projetos baseados nesse modelo de ciclo de vida.
>
> O ciclo de vida em cascata é adequado a projetos de pequena duração, tais como os projetos do PSP- Personal Software Process [Humphrey95]. É conveniente também para miniprocessos, que são subprocessos bem-

delimitados executados dentro de um processo maior; em capítulos posteriores, atividades como resolução de problemas, resolução de defeitos, alteração de requisitos, aquisições, manutenção e inovação são tratadas como miniprocessos com ciclo de vida em cascata. Finalmente, o modelo em cascata serve de referencia mesmo em projetos em que não é aplicado de forma pura; por isso, é um dos modelos suportados pela ferramenta de estimativas COCOMO II[Boehm+00].

Figura 2 - Modelo Cascata com Realimentação



Fonte: Pádua, 2011, p. 93.

O modelo cascata ainda é o mais usado em muitas organizações, “Em artigos relativamente recentes ([Neill+03], [Laplante+04]), Ph. Laplante e C.Neill relatam que o modelo em cascata ainda é o ciclo de vida mais usado, segundo levantamento feito pelos autores.” (PÁDUA 2011, p. 93).

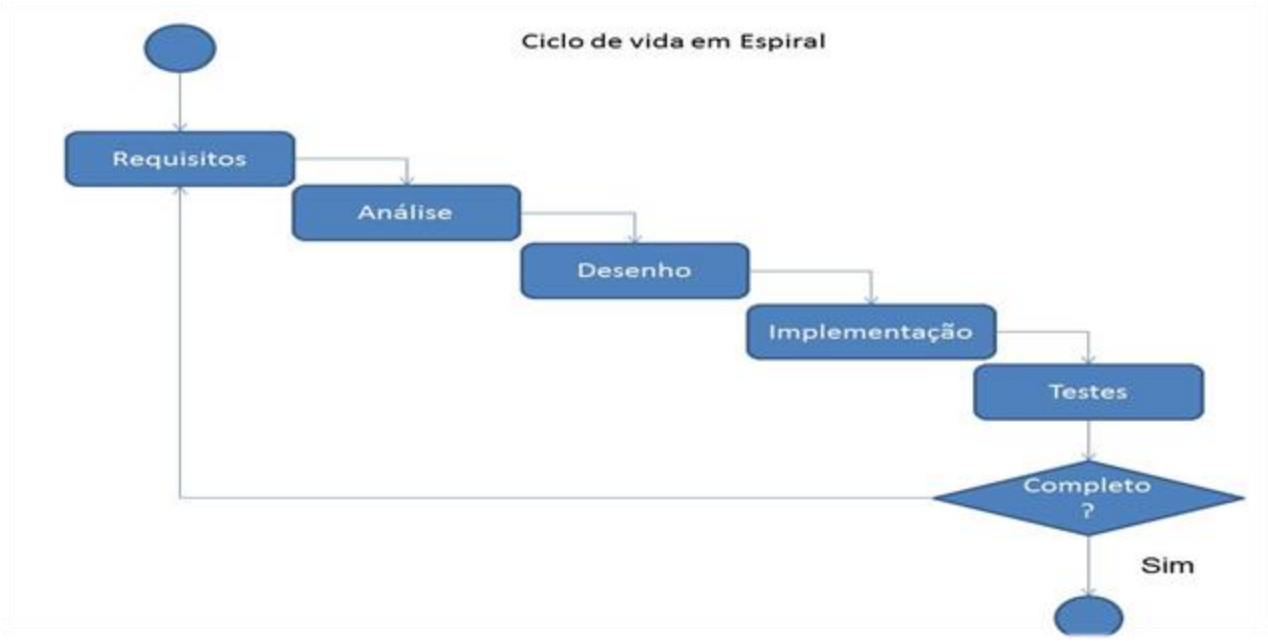
A partir do modelo cascata surgiram outros modelos de ciclos de vida, na ideia de suprir as necessidades do projeto e acabar com tradicionais problemas desse modelo, mas novos problemas foram surgindo e os modelos criticados, estudados e alterados.

2.1.3.2 Espiral

Esse modelo é totalmente diferente do modelo cascata visto anteriormente, suas etapas e processos seguem uma sequência repetitiva e fecham ciclos, como uma espiral imaginaria. A cada final de ciclo, pode-se dizer que há uma versão do projeto e esta versão ser levada ao início do próximo ciclo. Essa maneira de abordagem de desenvolvimento permite alta-flexibilidade e visibilidade.

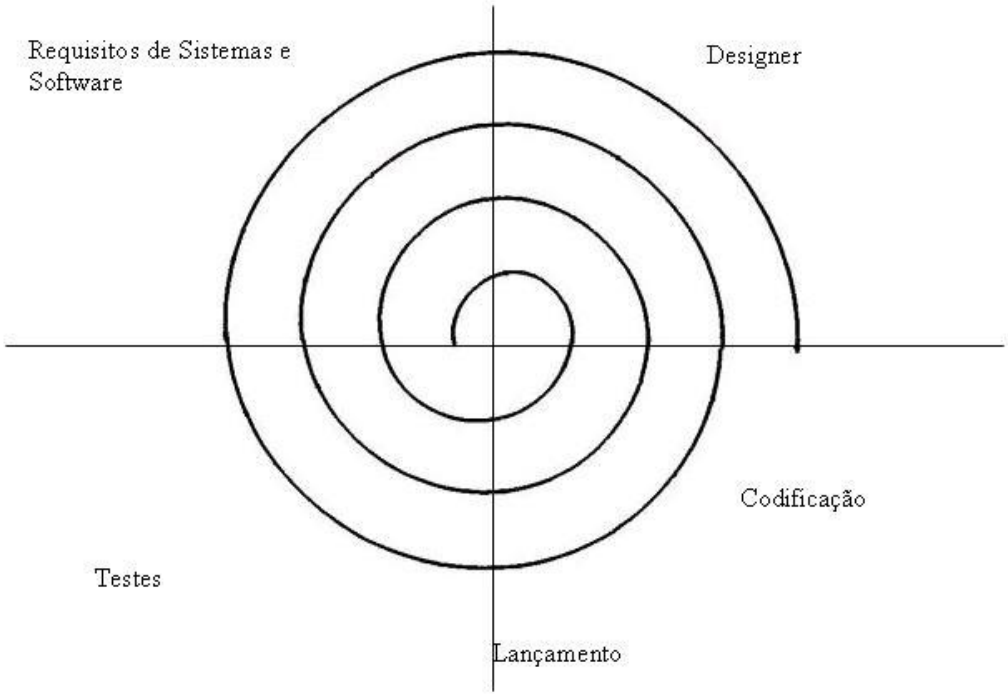
Essa flexibilidade implica no alto nível de conhecimento tácito e disciplina da equipe envolvida, a gestão sofre dificuldades no controle das informações, pois nesse modelo, cada ciclo faz com que a estrutura e essência do produto se distorçam. Na Figura 3 observa-se um modelo desse tipo de ciclo.

Figura 3 - Ciclo de vida em Espiral



Fonte: Pádua, 2011, p. 94.

Figura 4 - Ciclo de Vida em Espiral II



Fonte: Pádua, 2011, p. 94.

Pádua (2011, p. 94) complementa alguns conceitos sobre esse modelo:

> O produto é desenvolvido em uma série de iterações. Cada nova iteração corresponde a uma nova volta na espiral. Por liberação (release) entende-se um estagio parcialmente operacional de um produto, que é submetido à avaliação de usuários em determinado marco de um projeto. Embora a influência do processo MBASE tenha sido provavelmente muito importante para a difusão atual desse modelo, a origem da ideia é bem mais antiga, como é mostrado em estudo de Craig Larman e Victor Basili [Larman+03].
>
> Liberações escolhidas podem ser designadas como versões oficiais do produto e colocas em uso. Isso permite construir produtos em prazos curtos, como novas características e recursos que são agregados à medida que a experiência descobre sua necessidade. As atividades de manutenção são usadas para identificar problemas; seus registros fornecem dados para definir os requisitos das próximas liberações. :

Usando o modelo espiral, o software é desenvolvido numa série de versões evolucionarias. Durante as primeiras iterações, as versões podem ser um modelo de papel ou protótipo. Durante as últimas iterações, são produzidas versões cada vez mais completas do sistema submetido à engenharia. Um modelo espiral é dividido em um conjunto de atividades de arcabouço definidas pela equipe de engenharia de software; O modelo espiral é uma abordagem realista do desenvolvimento de sistemas de softwares de grande porte. Como o software evolui à medida que o processo avança, o desenvolvedor e o cliente entendem melhor e reagem aos riscos de cada nível evolucionário. Pressman (2010, p. 45).

É possível ter-se modelos mistos, na prática o modelo espiral puro é raramente usado, pois algumas etapas e processos não são repetidos nos próximos ciclos como a etapa de iniciação.

Uma variante do modelo espiral é o modelo Prototipagem evolutiva. Nesse modelo, a espiral é usada não para desenvolver o produto completo, mas para construir uma série de versões provisórias chamadas de protótipos. (PÁDUA, 2011, p. 95).

2.1.3.3 Prototipagem

Nas etapas de criação de um software, a análise de requisitos é crucial para o sucesso e eficiência do software. Muitas vezes o cliente não sabe descrever os seus problemas e requisitos, deixando o desenvolvedor inseguro e prejudicando a criação de uma solução de qualidade.

Esse modelo é considerado uma técnica de desenvolvimento e implementado junto com outros ciclos de vida citados nesse trabalho, pois mantem o conceito de entregas parciais do software para avaliação do cliente.

Essa interação com o cliente permite identificar e tratar as falhas, erros e necessidades do cliente, o protótipo serve como um mecanismo para identificação de requisitos do software. Se um protótipo executável é elaborado, o desenvolvedor tenta usar partes dos programas existentes e ferramentas de geração de código. (PRESSMAN, 2010, p. 42).

A prototipagem auxilia na identificação dos requisitos, mas acaba complicando a modelagem e codificação do sistema. Pressman (2010, p. 43), comenta alguns problemas comuns na prototipagem:

> O cliente vê o que parece ser uma versão executável do software, ignorando que o protótipo apenas consiga funcionar precariamente (“ele é mantido em pé com goma de mascar e arame”), sem saber que, na pressa de fazê-lo rodar, ninguém considerou a sua qualidade global ou manutenibilidade no longo prazo. Quando informado.

2.1.3.4 Ciclo de Vida do Projeto

A criação de um ciclo de vida do projeto, encontra-se no Capítulo 3, baseado em prototipagem devido às reuniões com orientador e a codificação semanalmente revisada e se necessária a remodelagem do sistema assemelhando-se aos processos da prototipagem.

2.1.2 Engenharia de Requisitos

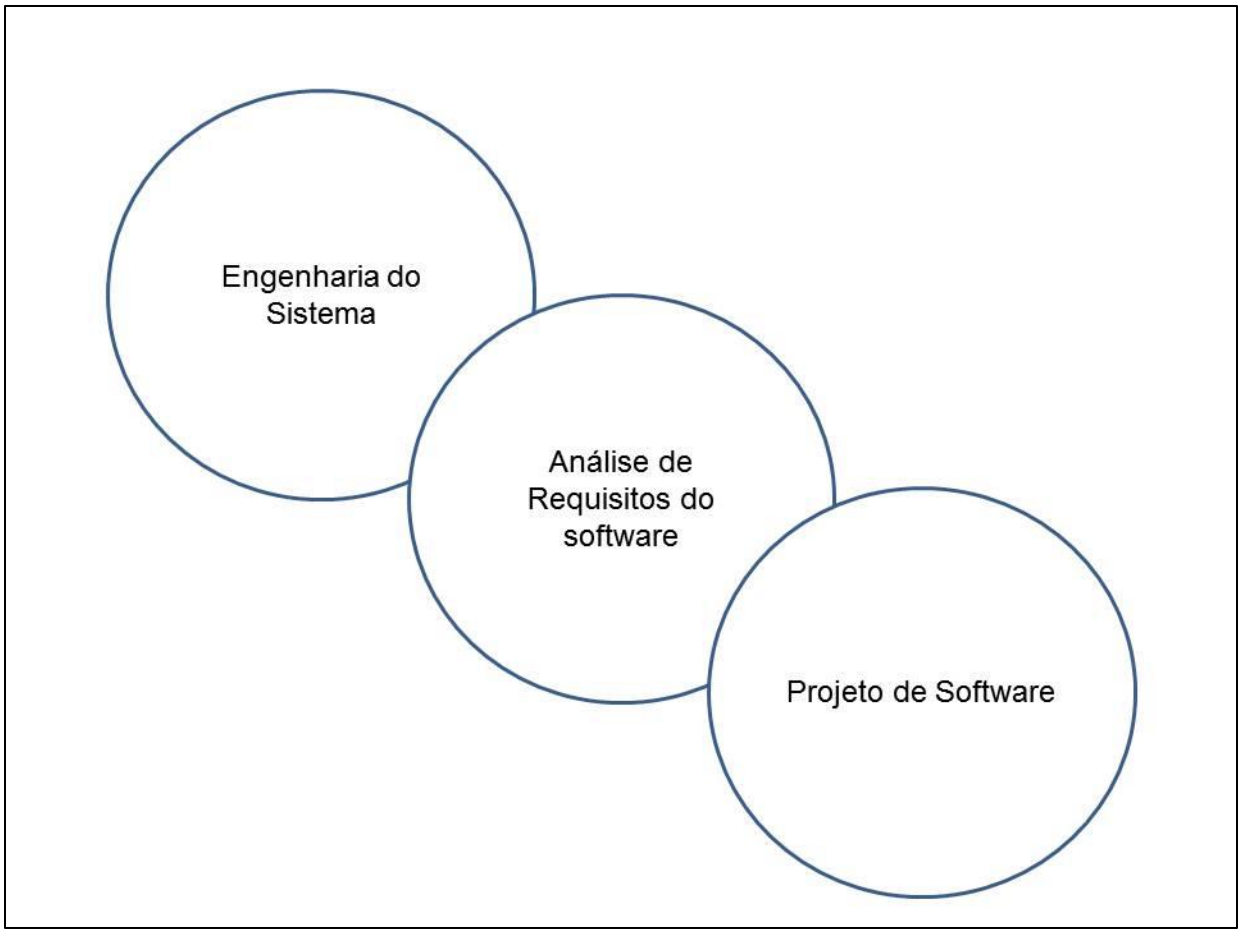
Uma das maiores dificuldades na criação de um software é simplesmente não saber definir os requisitos que ele necessita atender, dependendo do sistema deve se descrever suas características funcionais a estéticas, com que maneira irá se comportar nas diferentes alterações do ambiente.

Um texto escrito por Pressman (2010, p.116) deixa clara essa ideia:

> É seu pior pesadelo. Um cliente entra no seu escritório, senta-se, olha você direto nos olhos e diz: “Eu sei que você pensa que entende o que eu disse, mas o que você não entende é que, o que eu disse, não é o que eu queria dizer”. Invariavelmente, isso acontece no final de um projeto, depois que os compromissos de prazo de entrega foram feitos, que as reputações estão envolvidas e que dinheiro sério está em jogo.

A engenharia de requisitos, análise de requisitos ou levantamento de requisitos é responsável por identificar os requisitos mínimos do software para que ele ofereça uma solução ao problema do cliente, ela liga o projeto de software com a engenharia do sistema. (Figura 5)

Figura 5 - Interação do projeto de software e análise



Fonte: Pressman, 1995, p. 233.

Pressman (2010, p.116) relata em um parágrafo um pouco da sua experiência como analista:

> Todos nós que temos trabalhado no negócio de sistemas e softwares há alguns anos, vivemos esse pesadelo e, no entanto, poucos de nós aprenderam a se livrar dele. Lutamos quando tentamos levantar requisitos de nossos clientes. Temos dificuldade de entender as informações que conseguimos. Frequentemente registramos os requisitos de maneira desorganizada e gastamos pouco tempo verificando o que de fato registramos. Permitimos que as modificações nos controlem, em vez de estabelecer mecanismos para controlar as modificações. Em resumo,

> falhamos em estabelecer uma fundamentação sólida para o sistema ou software. Cada um desses problemas é um desafio. Quando eles se combinam, a perspectiva é danosa mesmo para os gerentes e profissionais mais experientes. Mas soluções existem de fato.

Esse conceito também é claramente expresso por Sommerville (2007, p. 79):

> Os Requisitos de um sistema de descrições de serviços fornecidos pelo sistema e as suas restrições operacionais. Esses requisitos refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido ou encontrar uma informação.

A análise de requisitos, é considerada por muitos autores e também por Pressman a etapa mais problemática de todas. Nenhuma das outras etapas da engenharia de software é tão danosa quanto essa, caso for mal feita ou pouco tratada. O grande problema que acontece na maioria dos casos é que nem o cliente sabe descrever o problema claramente. Um jeito para resolver isso foi à criação dos métodos de análise de requisitos.

Alguns problemas nessa etapa são a ambiguidade e dilemas entre analistas e clientes, Pressman (1995, p. 231) afirma:

> Tanto o desenvolvedor como o cliente desempenham um papel ativo na análise e especificação de requisitos. O cliente tenta reformular um conceito de função e desempenho e software, às vezes nebuloso, em detalhes concretos. O desenvolvedor age como indagador, consultor e solucionador de problemas.
>
> A análise e especificação de requisitos pode parecer uma tarefa relativamente simples, mas as aparências enganam. O conteúdo de comunicação é muito elevado. Abundam as chances de interpretações errôneas e informações falsas. A ambiguidade é provável. O Dilema com o qual se defronta um engenheiro de software pode ser mais bem entendido repetindo-se a declaração de um cliente anônimo: “Sei que você acredita que entendeu o que acha que eu disse, mas não estou certo de que percebe que aquilo que ouviu não é o que eu pretendia dizer...”

No mercado atual os programadores muitas vezes são ansiosos, compulsivos e extrapolam o escopo do projeto e suas especificações, codificam e constroem o

software compreendendo os requisitos à medida que codificam, pensando assim a engenharia de requisitos se torna inútil.

Quando isso acontece não existe nenhuma gestão, controle dos requisitos e suas iterações, isso é errado e pode levar a um projeto falho de software. (PRESSMAN, 1995, p. 117).

Fred Brooks (*apud* PRESSMAN, 1995, p.117) afirma que “A parte individual mais difícil da construção de um sistema é decidir o que construir. Nenhuma parte do trabalho danifica tanto o sistema resultante se for feita errado. Nenhuma outra parte é mais difícil de consertar depois.”.

Os requisitos podem ser classificados, segundo a Engenharia de Software, em duas categorias: requisitos funcionais e requisitos não funcionais.

2.1.2.1 Requisitos Funcionais

Os requisitos funcionais são aqueles que descrevem as funções do sistema, o que ele vai realmente fazer, o que ele vai processar, as funções que ele oferecerá. Esses requisitos variam de cada sistema, pois cada sistema atende e processa algo diferente, logo suas funções são diferentes. Neste projeto, está sendo realizado o estudo de um software produto considerado por Pressman como personalizado (Ver definição de software no item 5.1). Esses são os requisitos que na maioria dos casos são identificados pelos analistas por meio dos métodos de levantamento de requisitos.

O Q-Acadêmico do Instituto Federal Sul-Rio-Grandense pode ser citado como um exemplo de software que contempla um requisito funcional, pois ele permite a consulta e exibição do boletim do aluno e a opção de impressão seria um exemplo de requisito funcional. A organização dos menus e as configurações também foram requisitos descritos no escopo do Q-Acadêmico.

É natural nessa etapa cometer erros, devido aos métodos aplicados, eles ajudam a esclarecer um requisito, mas nem sempre o alvo do método é o correto,

isso também varia conforme a competência e o conhecimento tácito do analista ou do grupo de analistas.

2.1.2.2 Requisitos Não Funcionais

São requisitos de fácil definição, como nome diz, eles não estão diretamente ligados às funcionalidades do sistema, seriam como descrição de características visuais, físicas e de hardware também.

São relacionados às propriedades emergentes do projeto, como confiabilidade, segurança, espaço de armazenamento e tempo de resposta. Com esse pensamento é fácil identificar que eles são responsáveis pelas restrições e limitações do sistema, capacidades, disponibilidade.

Isso tudo não quer dizer que sejam menos importantes que os funcionais, pois todos os requisitos devem ser tratados com a mesma prioridade, uma vez que dependendo do requisito, funcional ou não funcional, pode ser a causa da invalidação total do sistema. Um exemplo citado por Sommerville (2007, p. 82) e que

> Os usuários do sistema em geral encontram meios de contornar uma função do sistema que não atenda às suas necessidades. Contudo, uma falha no atendimento de um requisito não funcional pode significar que todo sistema é inútil. Por exemplo, se uma aeronave não atende os requisitos de confiabilidade, ela não será certificada como segura para operação; se um sistema de controle de tempo real falhar em atender os requisitos de desempenho, as funções de controle não operarão corretamente.

Sommerville deixa claro que todos os requisitos devem estar em plena implementação e funcionamento, a falta de atendimento de algum pode acarretar desde pequenos a grandes problemas. Um exemplo seria a venda de um software personalizado que atende todos os requisitos funcionais de uma empresa imaginária. Supondo que nessa empresa as máquinas tem baixo processamento, se o requisito não funcional de processamento não for atendido, poucas máquinas ou

nenhuma nessa empresa serão capazes de utilizar esse software, logo tornando o produto final inútil naquele ambiente.

2.1.2.3 Gestão de Requisitos

Todos os requisitos coletados devem ser registrados e arquivados de maneira organizada, ajudando na alteração dos mesmos, na gestão do projeto e na manutenção do sistema futuramente. Cadastrando esses requisitos, é possível uma análise mais ampla das necessidades do cliente e o surgimento de requisitos derivados.

O cadastro desses requisitos não é uma tarefa fácil e deve-se usar métodos, processos ou ferramentas como um artefato de partida para um planejamento inicial. (PÁDUA, 2011, p.507).

O conceito artefato de partida é explicado por Pádua (2011, p. 507):

> No desenvolvimento de software, o escopo de um produto é definido por seus requisitos. Portanto, a definição final do escopo de um produto, no processo Praxis 3.0, é o Modelo do Problema, ou a Especificação dos requisitos, dele derivada. Como a própria produção desse artefato requer a execução de um conjunto de atividades que podem ser complexas, é preciso dispor de um artefato de partida, pelo menos para o planejamento inicial.
>
> Existem muitos tipos possíveis de artefatos de partida, dependendo das práticas da organização, do cliente e da natureza do produto. Podem-se encontrar na Web muitos formatos diferentes para esse artefato; são usuais as denominações documento de visão, definição do produto, conceito de operação e enunciado do trabalho. O processo Praxis 3.0 usa uma forma simples de Enunciado do trabalho que contém uma seção de visão geral resumindo os dados mais importantes do projeto proposto; uma lista das funções previstas; e uma avaliação das necessidades que geram as funções previstas, assim como os benefícios delas esperados.

Essas tabelas e métodos permitem uma visualização das funções, um conjunto inicial dos requisitos do projeto e as tarefas a serem entregues.

2.2 Projeto da análise

O projeto deve descrever em forma de documento as tarefas do sistema todo. As informações e dados obtidos na análise de requisitos são necessários para sua criação. É vital o estudo dos resultados da análise de requisitos para criação de um projeto de qualidade.

Existem sistemas que são difíceis de serem descritos com qualidade e de forma clara todas suas informações e detalhes, por isso, são desmembrados em subsistemas interligados entre si e tratados separadamente.

O planejamento do projeto é muito importante e comentado por Pressman (1995, p. 129):

> O tempo é o mais valioso bem disponível a um engenheiro de software. Se houver suficiente tempo disponível tempo, um problema pode ser adequadamente analisado, uma solução pode ser compreensivamente projetada, o código-fonte, cuidadosamente implementado e o programa, minunciosamente testado. Mas parece que nunca há tempo o bastante.

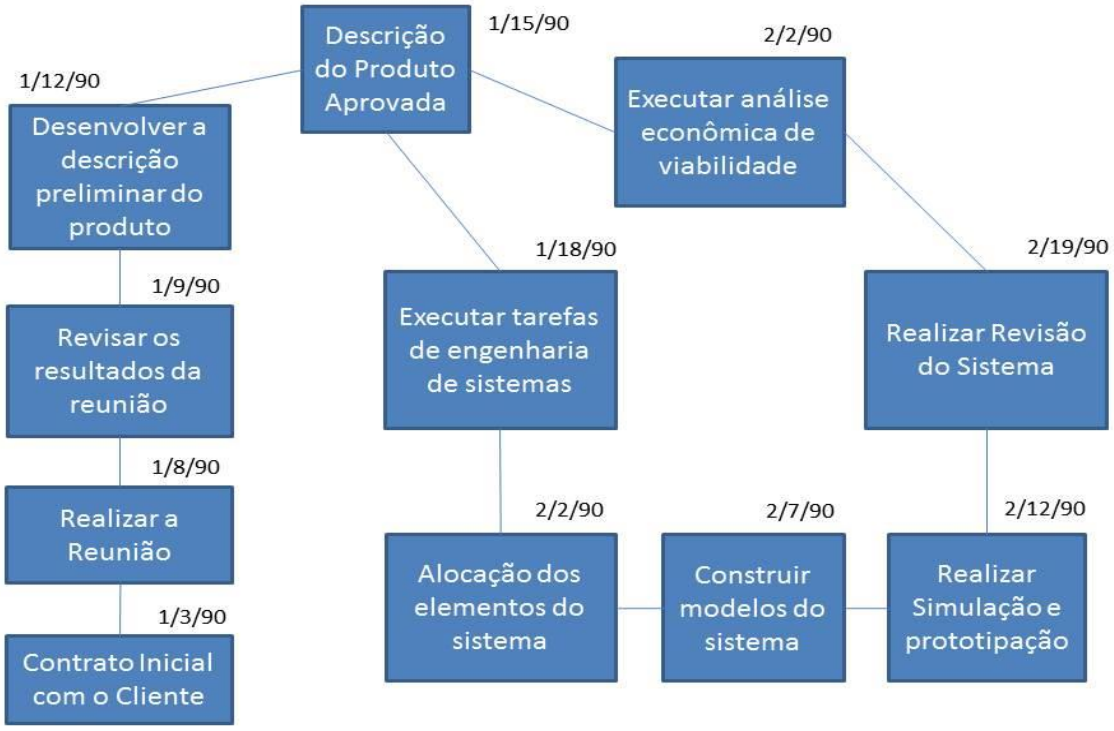
Todo engenheiro de software aprendeu a trabalhar sobre pressão de tempo privado do cumprimento de prazos finais arbitrários, que são estabelecidos por quem não tem de construir o produto. O projeto é planejado e tem seu cronograma criado casualmente com base em estimativas. Os riscos não são considerados até que se tornem um completo problema. (PRESSMAN, 1995, p. 129).

2.2.1 Cronograma do Projeto

A determinação de um cronograma para projetos de desenvolvimento de software pode ser vista a partir de duas perspectivas bem diferentes. A primeira

seria estimulada uma data final de entrega. A organização se vê comprometida em dividir as tarefas e esforço dentro do espaço de tempo previsto. A outra perspectiva estipula que os limites cronológicos aproximados tenham sido discutidos, mas que a data final seja estabelecida pela equipe de engenharia de software. As tarefas distribuídas de tal maneira a fim de tirar o maior proveito dos recursos e uma data final é definida após análise. (PRESSMAN, 1995, p.141). Na figura 6, observa-se um modelo de cronograma.

Figura 6 - Cronograma Simples



Pressman, 1995, p. 149

Para organizar essas informações, é necessário especificar o gênero do projeto, pode-se estruturá-lo de maneiras diferentes consequentemente alterando a organização do sistema. Serão estudadas duas arquiteturas de maior interesse neste sistema: Projeto de software em tempo real e Projeto orientado a objetos.

2.3 Modelagem de Dados

Com a análise de requisitos, estudo, investigação dos envolvidos e exploração dos recursos, obter-se-á uma grande quantidade de informações e será preciso organizar, filtrar, controlar e modelar tudo isso a fim de incorporar ao projeto com máximo de clareza e qualidade.

Toda essa dedicação é feita para que os programadores entendam claramente os requisitos e codifiquem corretamente a iteração. Tudo isso é documentado, registrado e arquivado.

A importância de se modelar o software ou modelar os dados é explicada por Guedes (2011, p. 20):

> Qual a real necessidade de se modelar um software? Muitos “profissionais” podem afirmar que conseguem determinar todas as necessidades de um sistema de informações de cabeça, e que sempre trabalharam assim. Qual a real necessidade de se projetar uma casa? Um pedreiro experiente não é capaz de construí-la sem um projeto? Isso pode ser verdade, mas a questão é muito mais ampla, envolvendo fatores extremamente complexos, como levantamento e análise de requisitos, prototipação, tamanho do projeto, complexidade, prazos, custos, documentação, manutenção e reusabilidade, entre outros.
>
> Existe uma diferença gigante entre construir uma pequena casa e construir um prédio de vários andares. Obviamente, para se construir um edifício é necessário, em primeiro lugar, desenvolver um projeto muito bem-elaborado, cujos cálculos têm de estar extremamente corretos e precisos. Além disso, é preciso fornecer uma estimativa de custos, determinar em quanto tempo a construção estará concluída, avaliar a quantidade de profissionais necessária à execução da obra, especificar a quantidade de material a ser adquirida para construção, ser modelados de cabeça, nem mesmo a maioria dos pequenos projetos pode sê-lo, exceto, talvez, aqueles extremamente simples.

Diversos métodos são usados para realizar essas tarefas, como o projeto deste sistema possui estrutura de um projeto Orientado a objetos com linguagem

orientada a objetos, será feito um estudo sobre o desenvolvimento de software utilizando UML 2.0, Casos de Uso, Diagramas.

2.3.1 UML 2.0

A UML 2.0 (*Unified Modeling Language* – Linguagem de Modelagem Unificada) é a linguagem-padrão de modelagem na engenharia de software. Seus conceitos e diagramas não são obrigatoriamente ligados a alguma etapa do processo de desenvolvimento do software assim como nem todos seus diagramas são utilizados, pois cada diagrama tem sua função especifica e em certos projetos somente alguns são necessários (GUEDES, 2011, p.19).

A UML Surgiu na fusão de três métodos de modelagem: método Boochm, método OMT (*Object Modeling Technique*) de Jacobson e método OOSE (*Object-Oriented Software Engineering*) de Rumbaugh. A junção desses métodos teve como ordem de início a união do método Booch ao OMT de Jacobson e o resultado fundiu-se com o método OOSE de Rumbaugh (GUEDES, 2011, p.20).

Não convêm descrevê-los no desenvolvimento deste trabalho, pois o resultado dessa fusão foi à primeira versão da UML.

Guedes, (2011, p. 20):

> O trabalho de Booch, Jacobson e Rumbaugh, conhecidos popularmente como “Os Três Amigos”, resultou no lançamento, em 1996, da primeira versão da UML propriamente dita.

O surgimento da UML 2.0 foi consequência da aceitação da UML no mercado, todas as empresas atuantes na engenharia de software adotaram a UML e passaram a contribuir para o projeto, fornecendo informações, sugestões e

melhorias. Em 1997 a UML foi adotada pela OMG (*Object Management Group*) como linguagem-padrão de modelagem.

Guedes, (2011 p.20):

> A versão 2.0 da linguagem foi oficialmente lançada em julho de 2005, encontrando-se esta atualmente na versão 2.3 beta. A documentação oficial da UML pode ser consultada no site da OMG em www.omg.org ou mais exatamente em www.uml.org.

A UML 2.0 possui inúmeros diagramas, iremos estudá-los e, com isso, tendo a análise de requisitos feita modelaremos o sistema com uso dos diagramas relevantes.

2.3.2 Diagramas da UML 2.0

Os diagramas da UML 2.0 possuem diferentes características e funções, mas todos possuem em comum a ideia de oferecer uma visão diferente do sistema ou parte dele, sendo possível a interação entre eles (GUEDES, 2011, p.29).

O motivo de tantos diagramas segundo (Guedes, p.30):

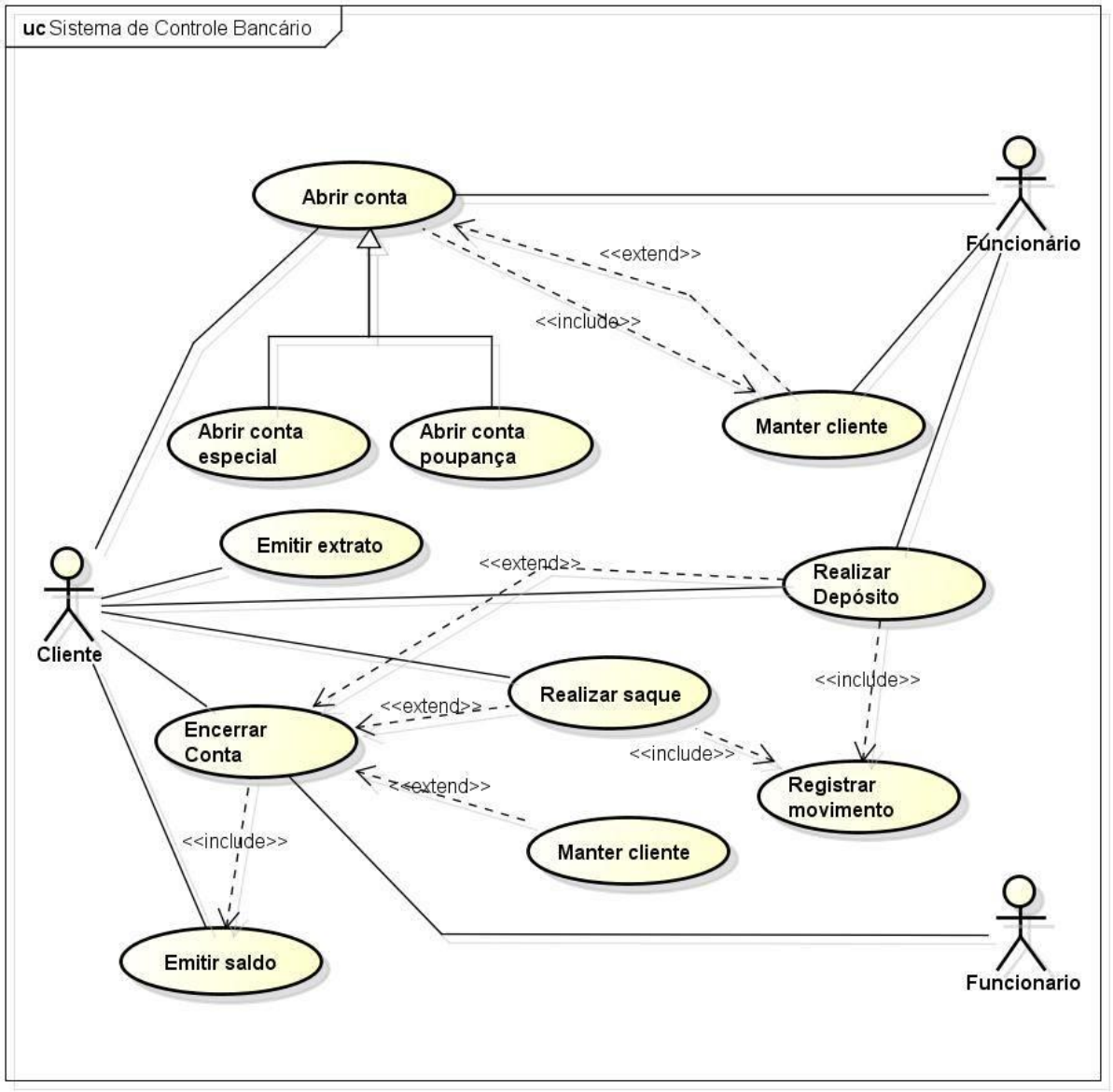
> Por que a UML é composta por tantos diagramas? O objetivo disso é fornecer múltiplas visões do sistema a ser modelado, analisando-o e modelando-o sob diversos aspectos, procurando-se, assim, atingir a completitude da modelagem, permitindo que cada diagrama complemente os outros.

Com a execução e criação de todos os diagramas é possível ter um entendimento total do sistema a ser modelado, como irá funcionar, quem irá utilizar suas funções, hardware entre muitas outras informações. Iremos explicar e ilustrar os mais importantes para o nosso projeto.

2.3.2.1 Diagrama Casos de Uso

Diagrama de casos de uso é uma ideia geral do sistema e com o fluxo das suas informações e também é o mais informal da UML. Usa uma linguagem simples, geralmente criado na etapa de levantamento de requisitos e pode ser auxiliar na criação de alguns dos outros diagramas. (GUEDES, 2011, p.30). A Figura 7 ilustra um exemplo deste diagrama.

Figura 7 – Exemplo de Diagrama de Casos de Uso



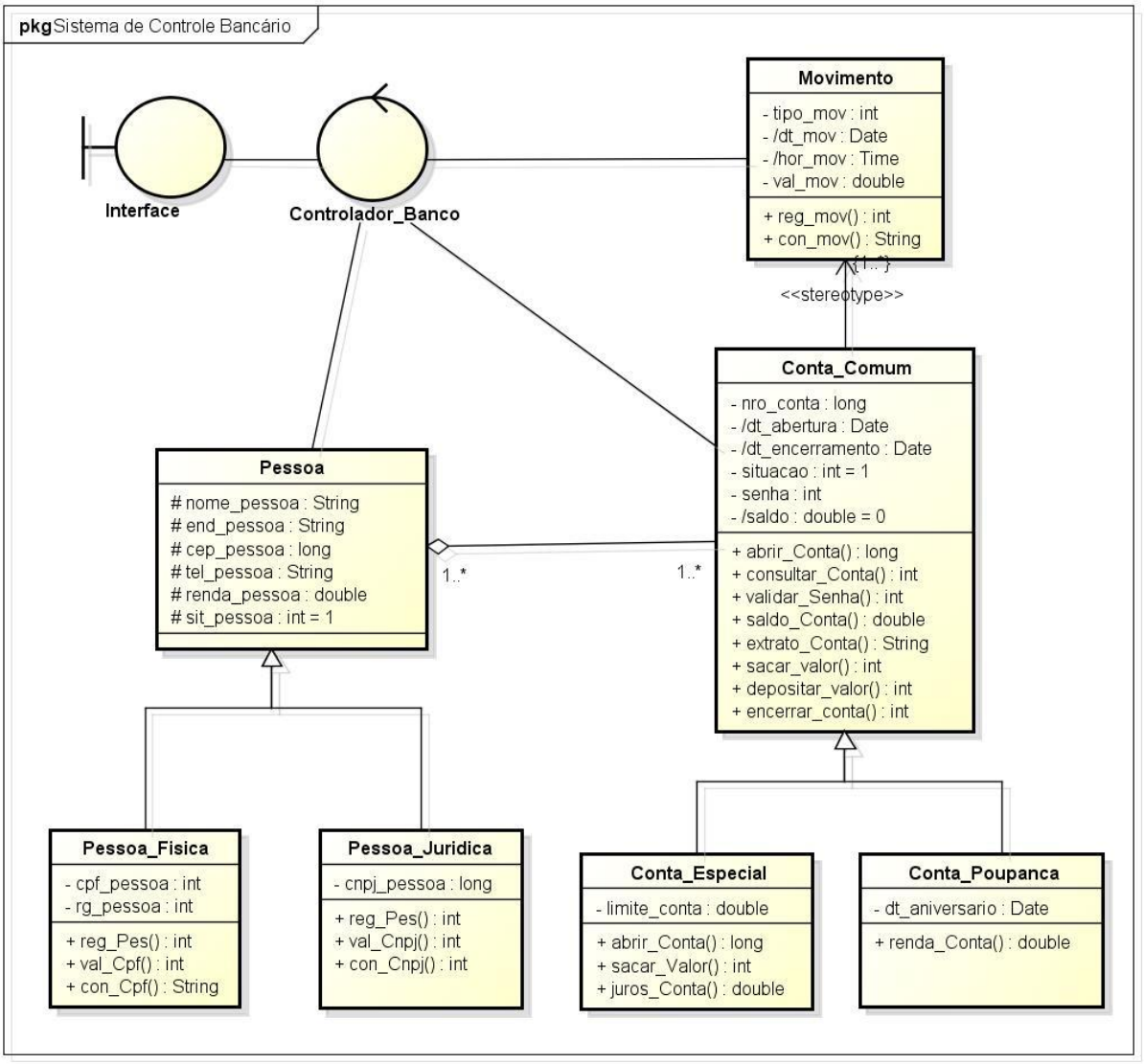
Fonte: Guedes, 2011, p. 31

2.3.2.2 Diagrama de Classes

Diagrama de classes segundo muitos profissionais é o mais importante e consequentemente o mais utilizado, auxilia praticamente todos os outros diagramas. Contêm informações das classes do sistema, seus atributos, métodos e como se

relacionam e interagem. (GUEDES, 2011, p.31). A Figura 8 ilustra um exemplo deste diagrama.

Figura 8 - Exemplo de Diagrama de Classes



Fonte: Guedes, 2011, p. 32

2.3.2.3 Diagrama de Objetos

Diagrama de objetos depende e complementa o diagrama de classes. O diagrama fornece uma visão dos valores dos objetos de um diagrama de classe em determinado momento de execução de um processo. (GUEDES, 2011, p.32).

2.3.2.4 Diagrama de Pacotes

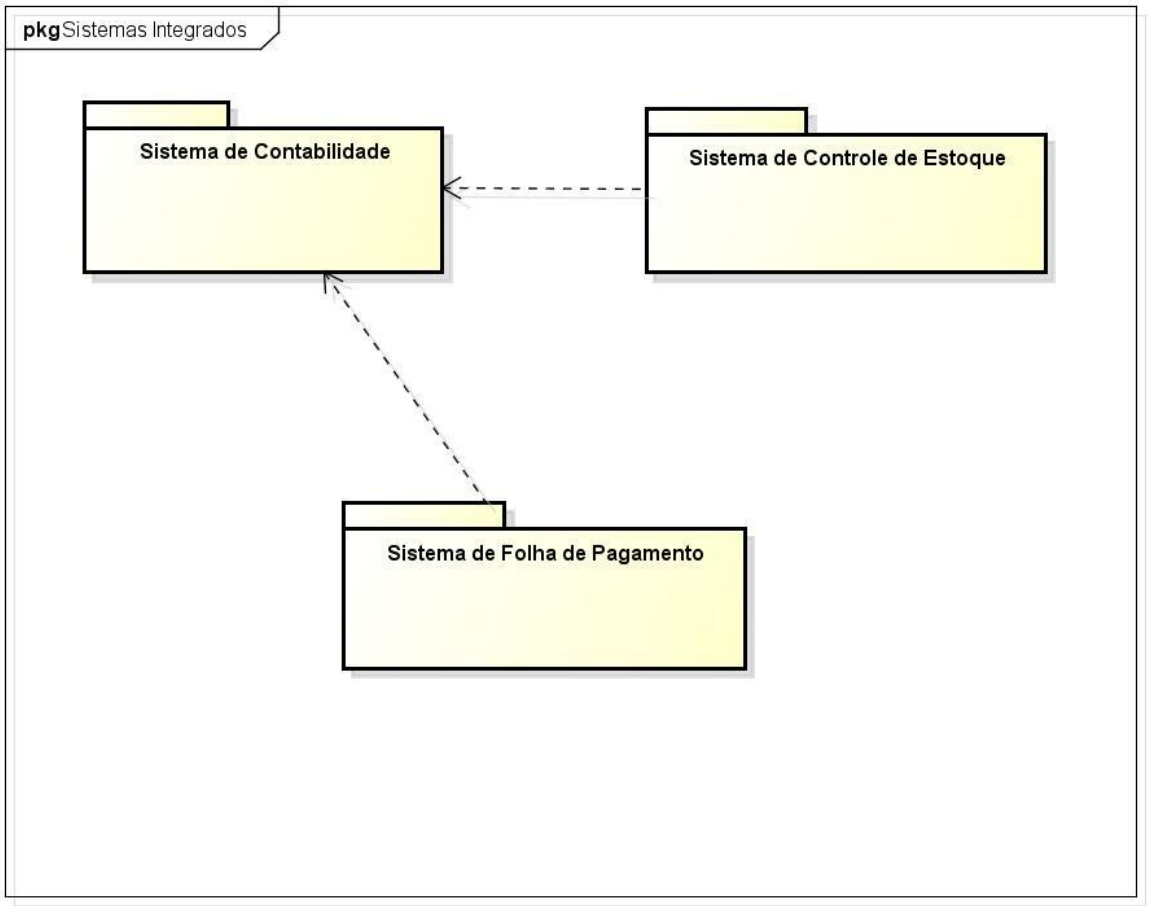
Diagrama de pacotes tem objetivo de demonstrar o relacionamento dos pacotes ou subsistemas do sistema a fim de identificar as partes que o constroem.

Segundo, (Guedes, p.33):

> O diagrama de pacotes é um diagrama estrutural que tem por objetivo representar os subsistemas ou submódulos englobados por um sistema de forma a determinar as partes que o compõem. Pode ser utilizado de maneira independente ou associado a outros diagramas. Este diagrama pode ser utilizado também para auxiliar a demonstrar a arquitetura de uma linguagem, como ocorre com a própria UML ou ainda para definir as camadas de um software ou de um processo de desenvolvimento.

A Figura 9 ilustra um exemplo deste diagrama.

Figura 9 - Diagrama de Pacotes

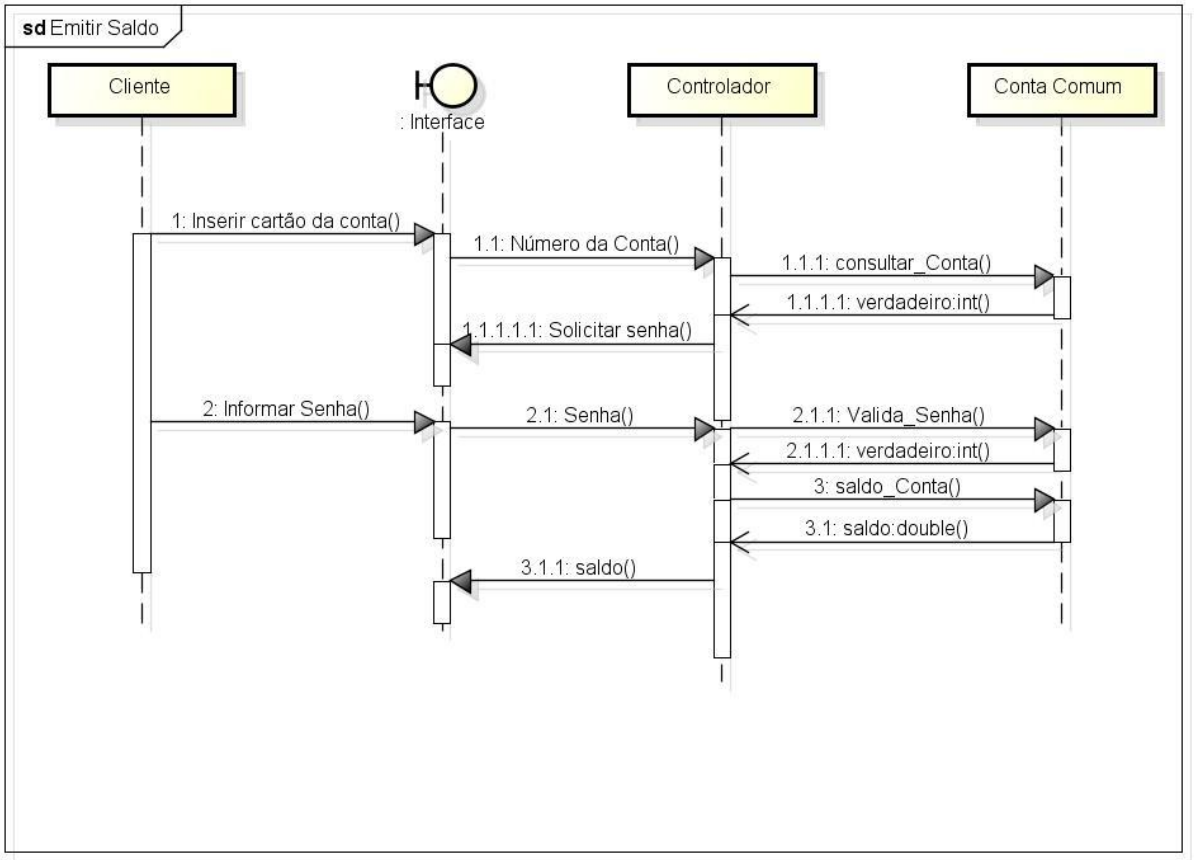


Fonte: Guedes, 2011, p. 33

2.3.2.5 Diagrama de Sequência

Diagrama de sequência é uma derivação do diagrama de casos de uso em conjunto com diagrama de classe, tem como objetivo principal demonstrar a ordem comportamental e o fluxo temporal da troca de informações entre os objetos de um processo. (GUEDES, 2011, p.33). A Figura 10 ilustra um exemplo deste diagrama.

Figura 10 - Diagrama de Sequência



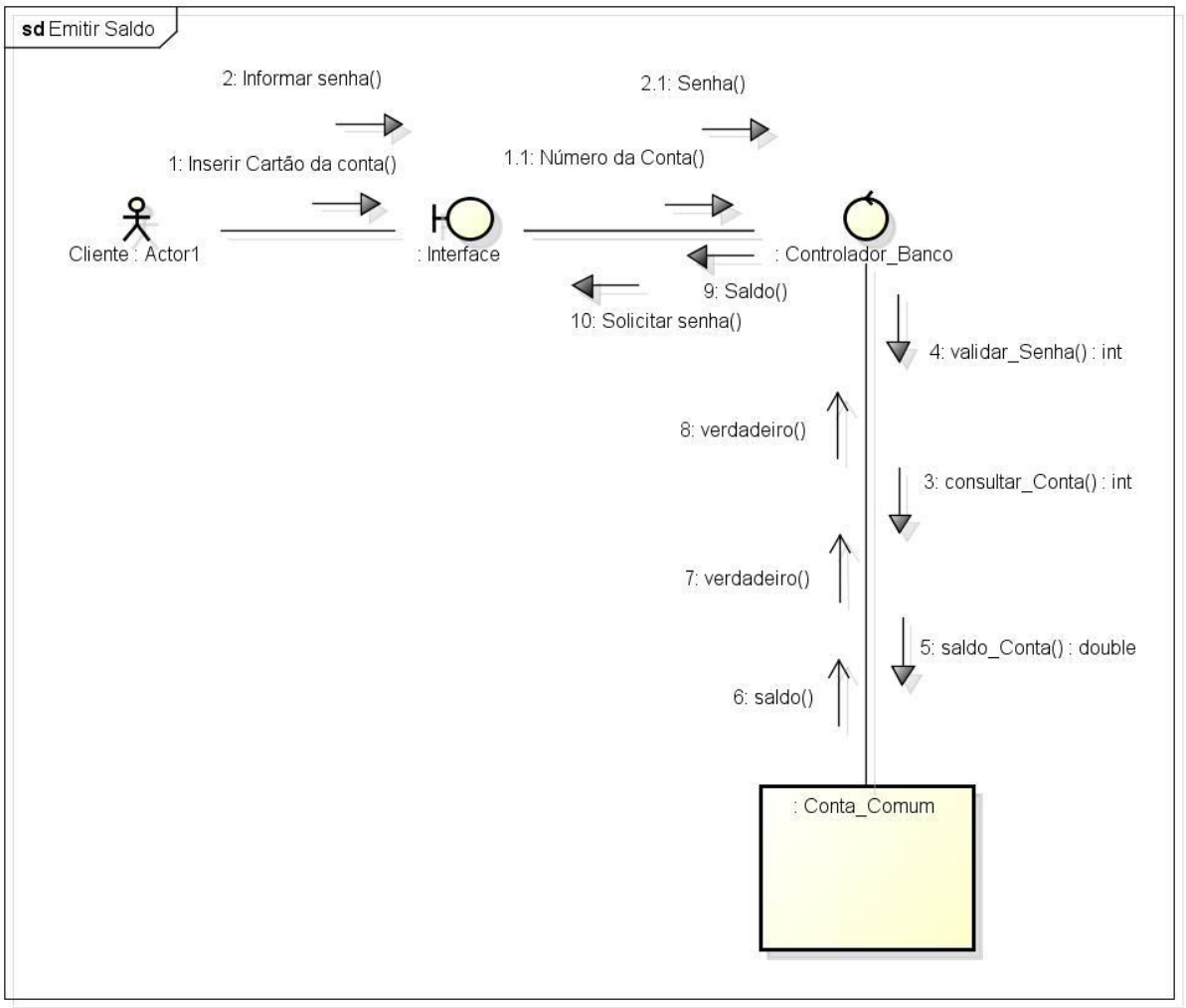
Fonte: Guedes, 2011, p. 34

2.3.2.6 Diagrama de Comunicação

Diagrama de comunicação é um complemento do diagrama de sequência, mas perde o foco temporal e preocupa-se com quais são as informações

trocadas entre as classes durante o processo. (GUEDES, 2011, p.35). A Figura 11 ilustra um exemplo deste diagrama.

Figura 11 - Diagrama de Comunicação

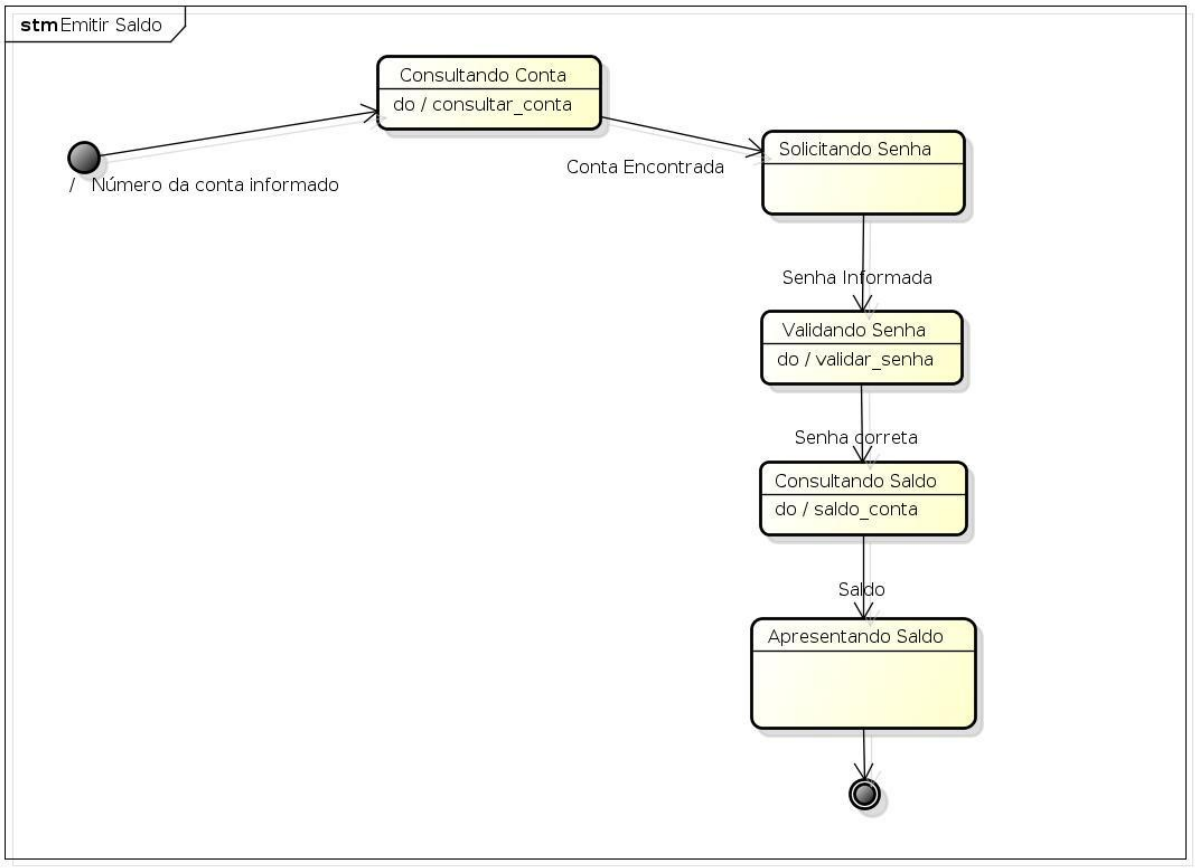


Fonte: Guedes, 2011, p. 35

2.3.2.7 Diagrama de Máquina de Estados

Diagrama de máquina de estados preocupa-se com o estado de um objeto, suas alterações, comunicações e transições. Pode ser usado para acompanhar um diagrama de casos de uso ou de outros elementos como uma instancia de uma classe. (GUEDES, 2011, p.35). A Figura 12 ilustra um exemplo deste diagrama.

Figura 12 - Diagrama de Máquina de Estados



Fonte: Guedes, 2011, p. 35

2.3.2.8 Diagrama de Atividade

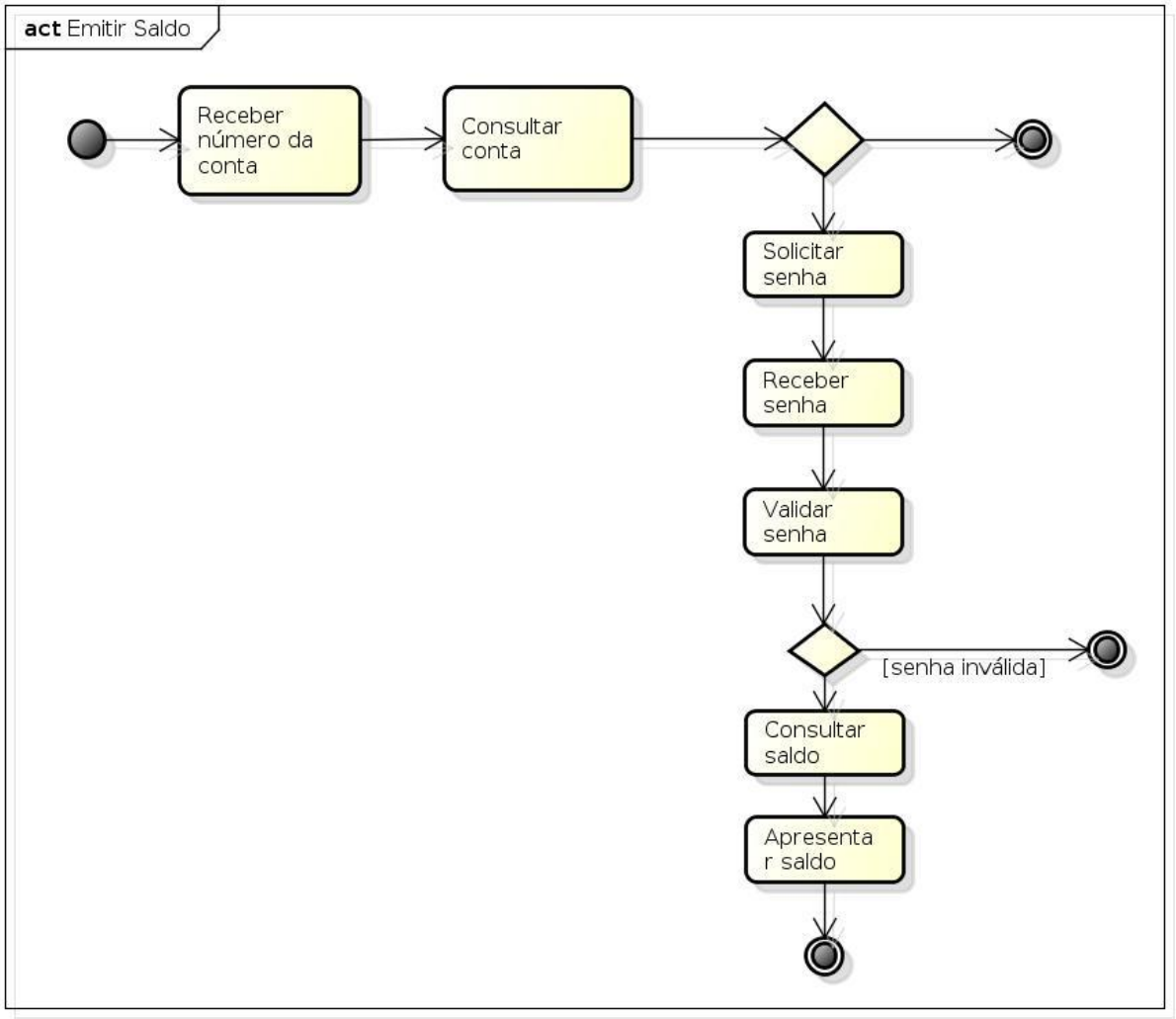
Diagrama de atividade é semelhante ao de sequência, mas preocupa-se em descrever os passos a serem efetuados ou simplesmente o fluxo de informações para conclusão de uma atividade. Esta atividade pode ser um método, um algoritmo ou até mesmo o processo completo. (GUEDES, 2011, p.36). A Figura 13 ilustra um exemplo deste diagrama.

2.3.2.9 Diagrama de Visão Geral de Interação

Diagrama de visão geral de interação é uma variação do diagrama de sequência (GUEDES, p.37):

O diagrama de visão geral de interação é uma variação do diagrama de atividade que fornece uma visão geral dentro de um sistema ou processo de negócio. Esse diagrama passou a existir apenas a partir da UML 2.

Figura 13 - Diagrama de Atividade



Fonte: Guedes, 2011, p. 36

2.3.2.10 Diagrama de Componentes

Diagrama de componentes demonstra os componentes do software quando o mesmo for implementado em questão de código-fonte, bibliotecas,

formulários, tutoriais, módulos, ferramentas, etc. Demonstrar a relação destes componentes para que tudo funcione perfeitamente. (GUEDES, 2011, p.39).

2.3.2.11 Diagrama de Implantação

Diagrama de implantação representa fielmente as definições de hardware, servidores, protocolos de comunicação, rede e qualquer outro aparato físico sobre o qual o sistema deverá ser executado. (GUEDES, 2011, p.39). A Figura 14 ilustra um exemplo deste diagrama.

Figura 14 - Diagrama de Implantação

Fonte: Guedes, 2011, p. 39

2.3.2.12 Diagrama de Estrutura Composta

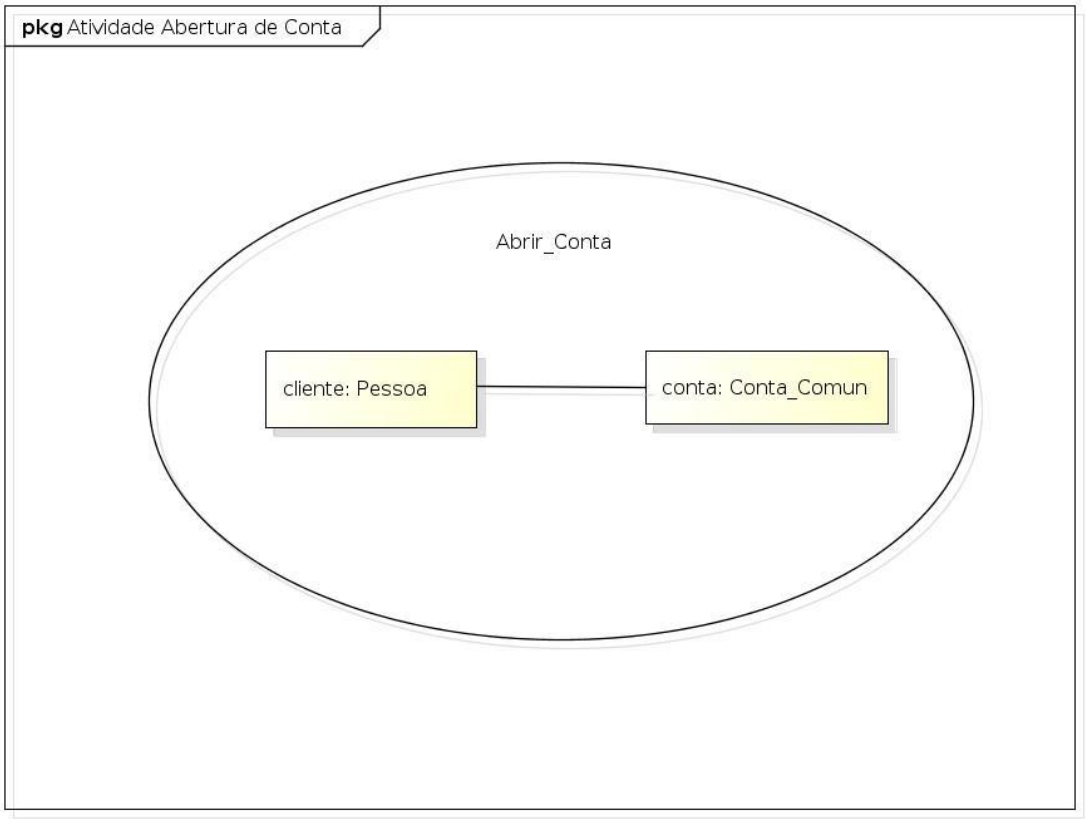
O diagrama de estrutura composta descreve a colaboração entre elementos para cumprirem uma tarefa (Guedes, p.40):

> O diagrama de estrutura composta descreve a estrutura interna de um classificador, como uma classe ou componente, detalhando as partes internas que o compõem, como estas se comunicam e colaboram entre si.

Também é utilizado para descrever uma colaboração em que um conjunto de instâncias cooperam entre si para realizar uma tarefa.

A Figura 15 ilustra um exemplo deste diagrama.

Figura 15 - Diagrama de Estrutura Composta



Fonte: Guedes, 2011, p. 40

2.3.2.13 Diagrama de tempo ou de temporização

O diagrama de tempo ou temporização preocupa-se com uma instância em determinado período e em descrever suas mudanças de estados e condições.(GUEDES, 2011, p.40).

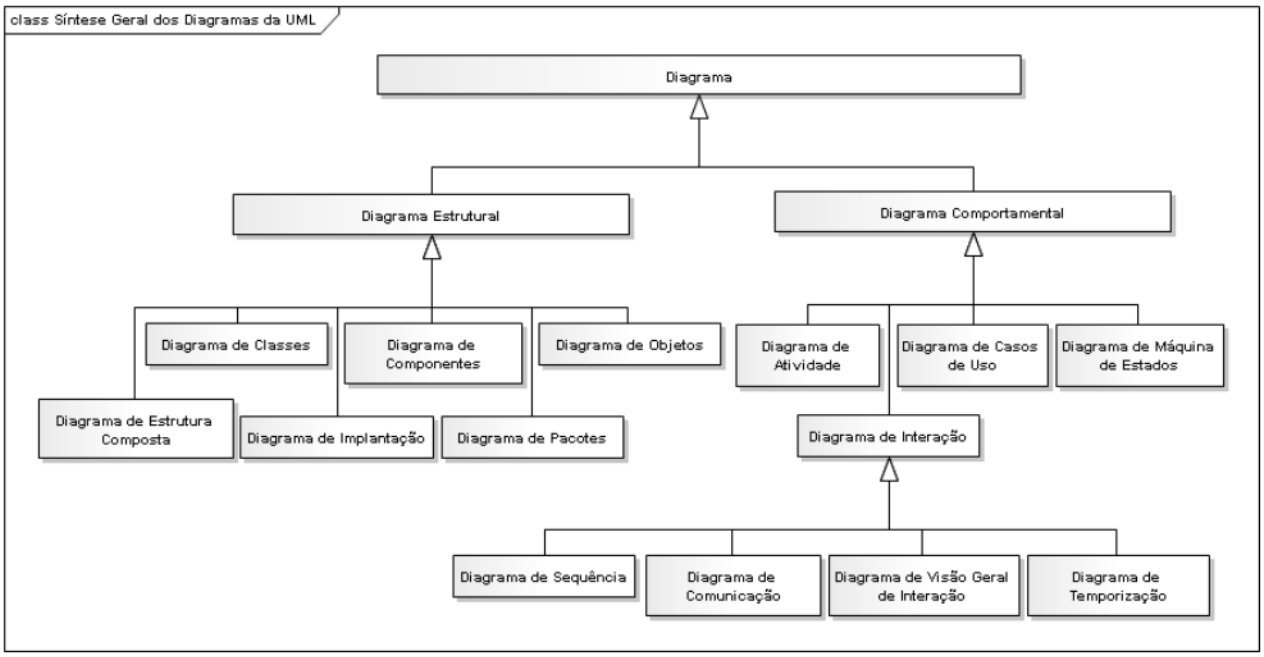
2.3.3 Resumo dos diagramas

Os diagramas dividem-se em diagramas estruturais e diagramas comportamentais e também de interação. Para melhor entendimento da relação entre todos eles é importante destacar a explicação de (Guedes, p.41):

> Os diagramas dividem-se em diagramas estruturais e diagramas comportamentais, sendo que os últimos ainda uma subdivisão representada pelos diagramas de interação.

Conforme a Figura 16 demonstra:

Figura 16 – Resumo dos Diagramas



Fonte: Guedes, 2011, p. 42

A partir desse estudo de todos os diagramas da UML 2.0, no próximo capitulo criou-se os diagramas correspondentes a nossa necessidade.

3 Modelagem do Sistema

O motivo de modelarmos um software é a ideia de que todo software não importa seu tamanho deve ser modelado antes do seu desenvolvimento, da sua entrega ou implementação, pois eles possuem a natural tendência a modicar-se e crescer em tamanho e complexidade, além disso, não podemos afirmar que um sistema está completamente finalizado, pois está em constante modificação. (GUEDES, 2011, p.21).

Segundo Guedes (2011, p. 21) Tais modificações ocorrem por diversos fatores, como, por exemplo:

- Os clientes desejam constantemente modificações ou melhorias do sistema.
- O mercado está sempre mudando, o que força a adoção de novas estratégias por parte das empresas e, consequentemente, de seus sistemas.
- O governo seguidamente promulga novas leis e cria novos impostos e alíquotas ou, ainda, modifica as leis, os impostos e alíquotas já existentes, o que acarreta a manutenção no software.

Com as informações geradas dessa apresentação o sistema sofre as modificações, alterações e adições necessárias.

O Instituto Federal Sul-Rio-Grandense, Campus Passo, passa por algumas dificuldades ao lidar com as solicitações de aproveitamento de disciplinas. O maior problema pode ser resumido pela falta de controle da documentação que sobrecarregam as etapas de análise dos coordenadores.

A descrição da situação atual do cliente foi construída a partir de entrevistas, questionários e outros métodos investigativos da engenharia de requisitos. Foram efetuadas entrevistas com as partes envolvidas no sistema e cadastrados seus problemas e futuros requisitos, os quais estão expressos na Tabela 1:

Tabela 1 - Entrevistas

Nome	Ator	Data da Entrevista
Roseli Moterle	Assistente em Administração	11/05/12
Adenilson Tonen	Assistente em Administração	11/05/12
Evandro Kuzsera	Coordenador do Curso TPSI	14/05/12
Cibele Borêa	Técnico em Assuntos Educacionais	13/05/12

Fonte: Do autor

Para entendimento do processo de criação de uma solicitação enumerou-se suas etapas:

1. O aluno solicita aproveitamento de matéria na CORAC (Coordenação de Registros Acadêmicos) com um Assistente em Administração. Esse profissional exige uma documentação para criação da solicitação que se constitui de:

- Histórico Escolar da antiga Instituição.
- Matriz Curricular da antiga Instituição.
- Programas da antiga Instituição.
- Ementas da Antiga Instituição.
- Conteúdos programáticos da antiga Instituição.

2. Com essa documentação em mãos, o Assistente em Administração destaca de um bloco de solicitações (Bloco de Solicitações Gerais), uma ficha e a preenche com as informações do aluno referentes à solicitação de aproveitamento e sua assinatura. Um modelo da ficha de solicitações gerais consta no ANEXO A e um exemplo de Ficha preenchida no ANEXO B.

3. Analisando a ficha, ela é espelhada e destacada no meio e fim, a primeira parte antes do picote central fica arquivada na pasta do aluno que fica na CORAC, a parte de baixo espelhada assinada pelo aluno é anexada à documentação fornecida por ele e encaminhada para o coordenador do curso, a terceira e última parte fica com o aluno para futuras consultas de informações correspondentes a essa solicitação.

4. O Coordenador do Curso foi identificado como o ator mais prejudicado por esse sistema, quando ele recebe a documentação do aluno encaminhada pelo Assistente Administração não possui nenhuma orientação ou controle. O Coordenador deve analisar esses documentos comparando todas as cadeiras do curso oferecido pelo Instituto com os que constam nos documentos fornecidos pelo aluno a fim de encontrar alguma compatibilidade.

5. A quantidade de material que chega ao coordenador é muito grande e dificulta o controle. Ele conta com a ajuda dos docentes para essa análise, pois cadeiras com nomes diferentes e mesmo conteúdo e carga horária compatível podem ser aproveitadas. Quando o coordenador chega a um resultado, ele preenche um documento chamado Solicitação de Aproveitamento de Estudos (ANEXO B) e o encaminha de volta à CORAC.
6. Quando há uma resposta do Coordenador do Curso, o Assistente em Administração fica responsável por informar o aluno e encaminhar essa Solicitação de Aproveitamento Disciplinas para um Técnico em Assuntos Educacionais.

7. O Técnico em Assuntos Educacionais por sua vez realiza a alteração na matricula do aluno marcando a cadeira como Aproveitada, baseado nas informações da Solicitação de Aproveitamento de Estudos.

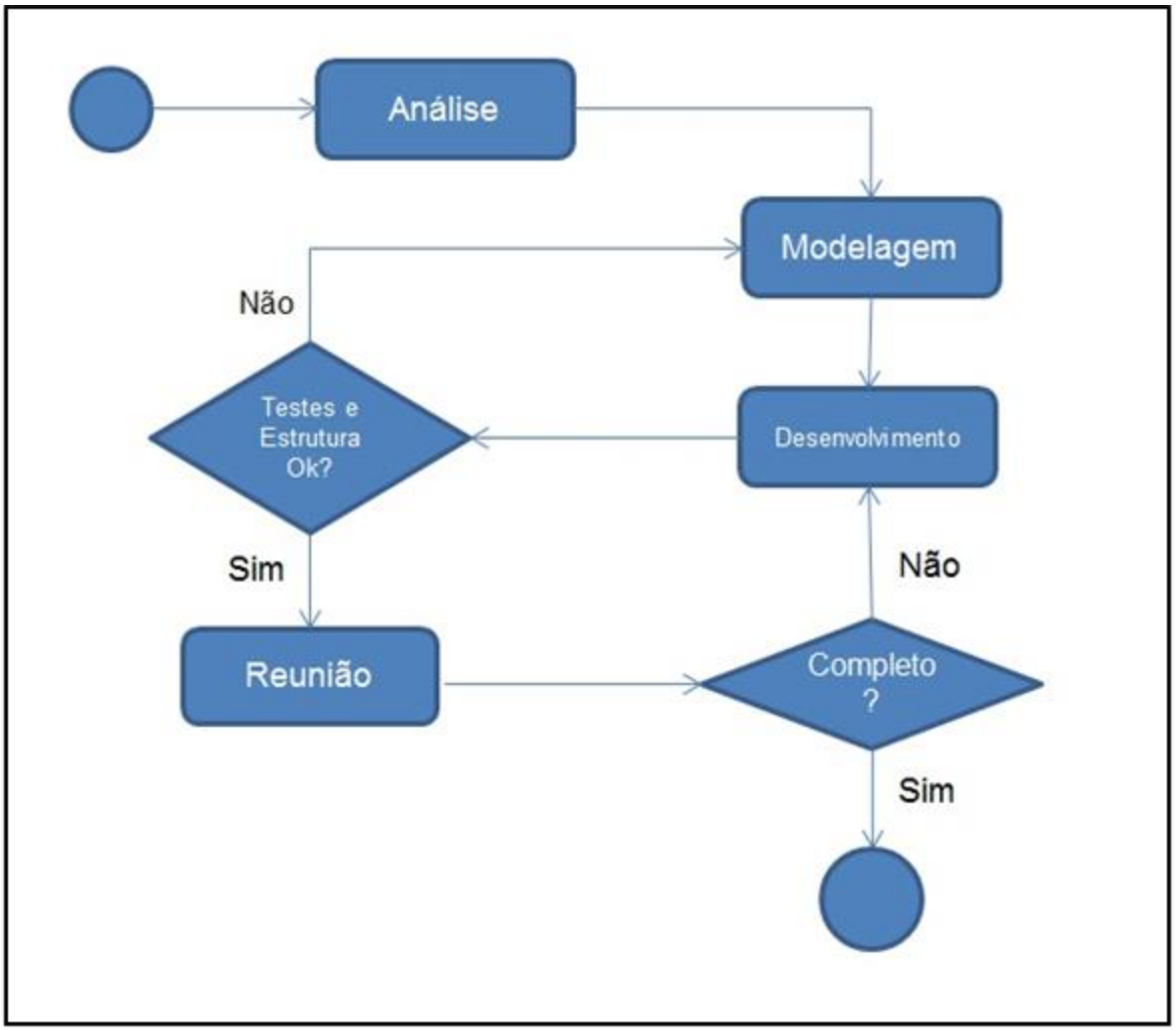
A partir desta descrição do processo atual, das etapas, das funções de cada ator e o fluxo das informações, nos próximos capítulos, usou-se tabelas para

demonstrar e descrever esses dados com mais clareza na etapa de Análise de Requisitos.

3.1 Ciclo de Vida do Projeto

Com os estudos do Capítulo de engenharia de software com base nos autores, SOMMERVILE, PRESSMAN, PÁDUA, criou-se um ciclo de vida com processos que atendem e satisfazem o nosso projeto. O ciclo de vida é representando pela figura 21.

Figura 17 - Ciclo de Vida do Projeto



Fonte: Do autor

3.1 Análise de Requisitos

Elaborou-se um Enunciado do Trabalho que contém um conjunto de tabelas que descrevem o projeto.

Elas têm objetivo de informar de forma clara o leitor, devido a quantidade excessiva de dados na etapa de análise.

3.1.1 Tabela Visão Geral

A Tabela 2 contém uma visão geral resumindo as informações mais importantes do projeto proposto. É uma tabela genérica, com informações simples, com missão do produto, cliente, proponente, prazos e etc.

Tabela 2 – Enunciado do Trabalho – Visão Geral

Produto	SAD 1.0
Missão do Produto	Oferecer apoio e controle dos aproveitamentos de disciplinas dos cursos do Instituto Federal Sul-Rio-Grandense (IFSUL).
Cliente	Instituto Federal Sul-Rio-Grandense (IFSUL).
Proponente	Aluno Giuseppe Matheus, Orientador Élder Benardi
Contato no Cliente	Élder Benardi
Contato no Proponente	Giuseppe Matheus
Caracterização do produto	O Sistema deve ser baseado em um banco de dados livre com interface web, executável em computadores de mesa, conectados por uma rede local.
Meta de Custo	Não estimada
Meta de Prazo	3 meses
Outros Aspectos	Serão utilizadas tecnologias cursadas no

	curso de Informática do Instituto.
Contrato	Compromisso de entrega parcial para aprovação na disciplina PC2.

Fonte: Do autor

3.1.2 Tabela Funções

A Tabela 3 que contém uma visão resumida das funções mais importantes e ações do projeto proposto. Esta tabela possui dados de quais as funcionalidades que o protótipo terá, ou que deve atender para tornar possível a execução da tarefa de criação de solicitação.

Tabela 3 - Enunciado do trabalho – Funções

Número	Função	Descrição
1	Gestão de Usuários	Controle de usuários que terão acesso ao SAD. Provê criação, atualização e exclusão.
2	Gestão dos Alunos	Controle dos alunos cadastrados no sistema. Provê criação, atualização, alteração, exclusão e controle.
4	Gestão de Disciplinas e Cursos	Controle das disciplinas e cursos no sistema, provê criação, atualização, alteração, exclusão e controle.
5	Gestão de aproveitamentos	Processamento das solicitações na CORAC, criação, atualização e exclusão de uma solicitação.
6	Emissão de Relatórios	Emissão do relatório com informações da solicitação.
7	Operação de Alteração de matricula	Operação de alteração de informações feitas pelo Técnico em Assuntos Administrativos.
8	Controle de Informações	Processamento e Validação na criação de novas solicitações geradas no sistema.
9	Gestão de Dados	Controle dos dados inseridos, como matrículas, datas e solicitações processadas.
10	Operação de	Operação de contato com aluno e ao coordenador

	contatos	via e-mail sobre alteração a uma solicitação.
11	Interface do Coordenador	Interface que auxilia o coordenador do curso a concluir a análise da solicitação.
12	Histórico	Provê o armazenamento de solicitações já analisadas para futuras validações e manutenções.

Fonte: Do autor

3.1.3 Tabela de Necessidades e Benefícios

Uma avaliação das necessidades e que geram as funções previstas e os benefícios dela esperados Tabela 4. Essa tabela contém quais benefícios serão alcançados com o atendimento das deficiências ou necessidades do atual processo.

Tabela 4 - Enunciado do trabalho – Necessidades e benefícios

Número	Função	Necessidades	Benefícios
1	Gestão de Usuários	Classificar os usuários	Privilégios de acesso diferenciados a cada função.
2	Gestão Alunos	Controle do Cadastro de alunos	Auxilia o controle de histórico e futuras solicitações.
3	Gestão de aproveitamentos	Registro das solicitações de aproveitamento	Complementa a gestão dos alunos e validação das solicitações
4	Emissão de Relatórios	Informações sobre o caso	Acompanhamento das informações
5	Solicitação de Alteração de matricula	Registro de alteração na matricula do aluno	Gerencia da comunicação entre CORAC e coordenação
6	Controle de	Classificar as	Apoio na avaliação

	Informações	solicitações entre as disciplinas.	dos documentos da antigo Instituto
7	Gestão de Dados	Controle das Informações do aluno	Apoio na classificação das disciplinas.
8	Operação de contatos	Comunicação entre o Sistema e o aluno	Acompanhamento da solicitação por parte do aluno
9	Interface do Coordenador	Interface de funções especificas do coordenador	Eliminação parcial do problema e redução no volume de trabalho.
10	Histórico	Armazena informações para futuras consultas	Automação do tratamento das solicitações

Fonte: Do autor

3.1.4 Tabela de Requisitos

A Tabela 5 contém uma lista dos requisitos coletados nas entrevistas e análises. Esta tabela contém de forma enumerada os requisitos que considerou-se vitais para o funcionamento do sistema em com suas funções mais básicas, para atender seu diagrama de classe.

Tabela 5 - Enunciado do trabalho – Requisitos Funcionais

ID	Nome	Prioridade	Estado	Categoria
R1	Remover Aluno	Essencial	Original	Interação
R2	Inserir Aluno	Essencial	Original	Interação
R3	Atualizar Aluno	Essencial	Original	Interação
R4	Listar Alunos	Essencial	Original	Interface de usuário
R5	Tela Controle de Alunos	Essencial	Original	Interface de usuário

R6	Mostrar as Solicitações do Aluno	Essencial	Derivado	Interface de usuário
R7	Imprimir relatório das informações do Aluno	Opcional	Derivado	Interação
R8	Enviar Email ao aluno	Opcional	Derivado	Interação
R9	Mostrar Todas Informações do Aluno	Essencial	Derivado	Interface de usuário
R10	Remover Matricula	Essencial	Original	Interação
R11	Inserir Matricula	Essencial	Original	Interação
R12	Atualizar Matricula	Essencial	Original	Interação
R13	Listar Matricula	Essencial	Original	Interface de usuário
R14	Tela Controle de Matricula	Essencial	Original	Interface de usuário
R15	Mostrar as Solicitações da Matricula	Essencial	Derivado	Interface de usuário
R16	Listar Cursos das Matricula	Essencial	Derivado	Interface de usuário
R17	Remover Curso	Essencial	Original	Interação
R18	Inserir Curso	Essencial	Original	Interação
R19	Atualizar Curso	Essencial	Original	Interação
R20	Listar Cursos	Essencial	Original	Interface de usuário
R21	Tela Controle de Cursos	Essencial	Original	Interface de usuário
R22	Mostras Disciplinas do Curso	Essencial	Derivado	Interface de usuário
R23	Listar Alunos do Curso	Essencial	Derivado	Interface de usuário
R24	Inscrever Aluno no	Essencial	Original	Interação

	Curso			
R25	Remover Aluno do Curso	Essencial	Original	Interação
R26	Remover Disciplina	Essencial	Original	Interação
R27	Inserir Disciplina	Essencial	Original	Interação
R28	Atualizar Disciplina	Essencial	Original	Interação
R29	Listar Disciplina	Essencial	Original	Interface de usuário
R30	Tela Controle de Disciplina	Essencial	Original	Interface de usuário
R31	Buscar Disciplinas	Essencial	Derivado	Interface de usuário
R32	Remover Solicitação	Essencial	Original	Interação
R33	Inserir Solicitação	Essencial	Original	Interação
R34	Atualizar Solicitação	Essencial	Original	Interação
R35	Listar Solicitação	Essencial	Original	Interface de usuário
R36	Tela Controle de Solicitações	Essencial	Original	Interface de usuário
R37	Buscar Solicitação	Essencial	Derivado	Interface de usuário
R38	Responder Solicitação	Essencial	Derivado	Interação
R39	Encerrar Solicitação	Essencial	Derivado	Interação
R40	Responsar Solicitação mediante avaliação	Essencial	Derivado	Interação
R41	Remover Usuário	Essencial	Original	Interação
R42	Inserir Usuário	Essencial	Original	Interação
R43	Atualizar Usuário	Essencial	Original	Interação
R44	Listar Usuário	Essencial	Original	Interface de usuário
R45	Tela Controle de Usuário	Essencial	Original	Interface de usuário

R46	Buscar Usuário	Essencial	Derivado	Interface de usuário
R47	Alterar Nível	Essencial	Derivado	Interação
R48	Histórico do Usuário	Essencial	Derivado	Interação
R49	Mostrar Solicitações criadas pelo Usuário	Essencial	Derivado	Interação

Fonte: Do autor

3.1.5 Tabela de Atores

Para o estudo de caso identificou-se os atores, representados na Tabela 6.

Tabela 6 - Descrição dos Atores

Numero	Ator	Descrição
1	Aluno	Principal Ator que dispara o processo de solicitação de aproveitamento.
2	Assistente em Administração	Funcionário responsável pela criação da solicitação, encaminhamento dos documentos e auxílio ao aluno.
3	Coordenador	Funcionário responsável pela análise da solicitação do aluno e encaminhamento do resultado para o Técnico em Assuntos Administrativos.
4	Técnico em Assuntos Educacionais	Funcionário responsável pela alteração dos dados da matricula do aluno.

Fonte: Do autor

3.2 Casos de Uso

O caso de uso pode ser usado para descrever o fluxo das informações no sistema, nele descreveu-se as ações mais importantes e usou-se como ferramenta para o desenho do diagrama. A Tabela 7 contém os casos de usos considerados principais no projeto.

Tabela 7 - Casos de Uso

ID	Caso de Uso	Descrição
1	Gestão de Usuários	Controle de Usuários que terão acesso ao SAD. Provê criação, exclusão e alteração.
2	Gestão de Solicitações	Controle de Solicitações geradas. Provê criação, exclusão e alteração.
3	Emissão de Relatório	Emissão de relatórios das bases de dados do SAD, para informações sobre o andamento do processo.
4	Operação de Criação de Solicitação	Operação de criação de uma solicitação feita pelo Assistente em Administração. Durante a operação é possível criar uma solicitação. Ao término da operação, um relatório é emitido com os dados e a base de dados atualizada.
5	Encerramento de Caso	Operação de encerramento de solicitação feita pelo Coordenador. Durante o processo é possível alterar dados da solicitação e atribuir um resultado final ao processo, Indeferido ou Deferido.
6	Alteração de Matricula	Operação final do processo feita pelo Técnico em Assuntos Educacionais, alterando os dados da matrícula do aluno de acordo com o resultado do Coordenador.
7	Comunicação com Aluno	Comunicação feita pelo Assistente em Administração ao Aluno sobre o status da sua solicitação. Ao término do processo a emissão de um relatório com resultados finais e registro da solicitação.

Fonte: Do autor

3.3 Documentações do Caso de Uso

No estudo dos casos de uso, documentou-se cada caso e suas funções descritas, interações e atores envolvidos.

Caso de uso gestão de usuário Tabela 8, onde o técnico em Assuntos Administrativos efetua a gerenciamento dos usuários do sistema;

Tabela 8 - Caso de Uso 1 – Gestão de Usuários

1	
Nome do Caso de Uso	Gestão de Usuários
Caso de Uso Geral	
Ator Principal	Técnico em Assuntos Educacionais
Atores Secundários	Coordenador, Assistente em Administração.
Resumo	Controle de Usuários que terão acesso ao SAD. Provê criação, exclusão e alteração de usuários. Provê criação de um perfil, logon e senha.
Pré-condições	Deve-se ocupar cargo de Assistente em Administração ou Coordenador.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Seleciona a criação de um novo usuário ou alteração dos dados de um usuário existente.	
	2. Apresentar Formulário correspondente.

3. Submissão do formulário completo.	
	4. Comitar o cadastro atualizando a base de dados.
Restrições	Dados Inválidos no Formulário devem ser tratados.
Validações	

Fonte: Do autor

Caso de uso Gestão de Solicitações Tabela 9, caso de uso onde o Assistente em Administração efetua a gerenciamento das solicitações de aproveitamento do sistema.

Tabela 9 - Caso de Uso 2 - Gestão de Solicitações

2	
Nome do Caso de Uso	Gestão de Solicitações
Caso de Uso Geral	
Ator Principal	Assistente em Administração
Atores Secundários	Coordenador, Aluno
Resumo	Controle de solicitações geradas pelo SAD. Provê criação e exclusão de solicitações e alterações de dados.
Pré-condições	
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Seleciona a interface ou tela de gestão de solicitações	
	2. Apresenta a lista de solicitações com as funções: Criar, Excluir, Editar.
3. Escolhe a função desejada.	

	4. Apresenta Formulário correspondente à função.
5. Submete o formulário das novas informações.	
	6. Comita o cadastro atualizando a base de dados e retorna a tela de gestão de solicitações.
Restrições	Dados Inválidos no Formulário devem ser tratados. Documentos exigidos para a composição de uma solicitação devem ser analisados pelo Assistente em Administração.
Validações	

Fonte: Do autor

Caso de uso Emissão de Relatório Tabela 10, caso de uso onde o Assistente em Administração efetua a impressão de documentos e formulários do sistema;

Tabela 10 - Caso de Uso 3 - Emissão de Relatório

3	
Nome do Caso de Uso	Emissão de Relatório
Caso de Uso Geral	
Ator Principal	Assistente em Administração
Atores Secundários	Aluno
Resumo	Funcionalidade de impressão de um documento contendo os dados da solicitação mediante a solicitação do Aluno.
Pré-condições	Deve-se haver a solicitação de acompanhamento do Aluno.
Pós-condições	Nenhuma
Fluxo Principal	

Ações do Ator	Ações do Sistema
1. Seleciona lista de solicitações	
	2. Apresentar a lista com as funções de gestão de solicitação e Impressão.
3. Submete a opção Impressão	
	4. Executa a tarefa de estruturação das informações do documento e em seguida envia-o a impressora local.
Restrições	Dever-se ter uma impressora instalada.
Validações	

Fonte: Do autor

Caso de uso Operação de Criação de Solicitação Tabela 11, caso de uso onde o Assistente em Administração e o Aluno efetuam a criação de uma nova solicitação de aproveitamento do sistema;

Tabela 11 – Caso de Uso 4 - Operação de Criação de Solicitação

4	
Nome do Caso de Uso	Operação de Criação de Solicitação
Caso de Uso Geral	
Ator Principal	Aluno
Atores Secundários	Assistente em Administração.
Resumo	O Aluno dirige-se a CORAC, com os documentos necessários e faz a solicitação para um Assistente, que irá criar no sistema essa solicitação.
Pré-condições	Assistente em Administração deve analisar os documentos e pesquisar por

	antigas ocorrências.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Solicita a criação de uma solicitação de aproveitamento.	
	2. Apresentar Formulário correspondente à criação.
3. Submissão do formulário completo.	
	4. Comitar o cadastro da solicitação atualizando a base de dados e informando o Coordenador via mensagem.
5. Assistente arquiva os documentos na pasta do aluno e encaminha ao coordenador	
Restrições	Dados Inválidos no Formulário devem ser tratados.
Validações	A disciplina a ser aproveitada deve ser apontada pelo aluno. O sistema só permite a criação de uma solicitação para o semestre atual. O Aluno não pode solicitar o mesmo aproveitamento duas ou mais vezes.

Fonte: Do autor

Caso de uso Encerramento de Solicitação Tabela 12, caso de uso onde o Assistente em Administração e o Coordenador encerram uma solicitação de aproveitamento no sistema.

Tabela 12 - Caso de Uso 5 Encerramento de Solicitação

5	
Nome do Caso de Uso	Encerramento de Solicitação
Caso de Uso Geral	
Ator Principal	Coordenador
Atores Secundários	Técnico em Assuntos Administrativos, Assistente em Administração.
Resumo	Coordenador analisa a solicitação e retorna o resultado ao Técnico em Assuntos Administrativos que altera a matricula do aluno e encerra a solicitação.
Pré-condições	Solicitação deve estar Deferida ou Indeferida.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Analisa os documentos do aluno e a solicitação do sistema e solicita opção de resposta na tela de gestão de solicitações	
	2. Apresentar Formulário correspondente ao coordenador com as funções de Deferir, Indeferir, Deferido com Prova.
3. Submissão do formulário completo, com a resposta.	
	4. Comitar o cadastro atualizando a base de dados das solicitações.
5. O Técnico em Assuntos Administrativos recebe a resposta	

e encerra o caso.	
	6. Informa o Assistente em Administração a atualização da solicitação e confirma o contato do aluno.
Restrições	Dados Inválidos no Formulário devem ser tratados.
Validações	

Fonte: Do autor

Caso de uso Alteração de Matricula Tabela 13, caso de uso onde o Técnico em Assuntos Administrativos efetua alteração da matricula do aluno com as informações de uma solicitação encerrada.

Tabela 13 - Caso de Uso 6 - Alteração de Matricula

6	
Nome do Caso de Uso	Alteração de Matricula
Caso de Uso Geral	
Ator Principal	Técnico em Assuntos Administrativos
Atores Secundários	Coordenador, Assistente em Administração.
Resumo	Operação final do processo feita pelo Técnico em Assuntos Administrativos, alterando os dados da matricula do aluno de acordo com o resultado do Coordenador.
Pré-condições	Deve-se haver resultados da análise do Coordenador.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Altera a matricula do aluno conforme a resposta da	

solicitação analisada Coordenador.	
	2. Esse sistema se limita à falta de integração com o sistema existente no Instituto, a alteração de matricula será realizada via o sistema do IFSUL e não será um requisito atendido por esse projeto.
3. O Técnico em Assuntos Administrativos encerra a solicitação no sistema.	
	4. Altera o status da solicitação na base de dados para finalizada.
Restrições	Dados Inválidos no Formulário devem ser tratados.
Validações	

Fonte: Do autor

Caso de uso Comunicação com Aluno Tabela 14, caso de uso onde o Assistente em Administração fornece informações referentes a solicitação de aproveitamento do aluno;

Tabela 14 - Caso de Uso 7 - Comunicação com Aluno

7	
Nome do Caso de Uso	Comunicação Com Aluno
Caso de Uso Geral	
Ator Principal	Assistente em Administração
Atores Secundários	Técnico em Assuntos Administrativos, Aluno.
Resumo	Comunicação feita pelo Assistente em Administração ao Aluno sobre o status da sua solicitação. Ao término do processo a emissão de um relatório com

	resultados finais e registro da solicitação.
Pré-condições	Deve-se haver resultados da análise do Coordenador e consulta do Aluno.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Na tela de gestão de solicitações o Assistente em Administração solicita a emissão de relatório.	
	2. Esse sistema se limita a falta de integração com o sistema existente no Instituto, a alteração de matricula será realizada via o sistema do IFSUL e não será um requisito atendido por esse projeto.
3. O Técnico em Assuntos Administrativos encerra a solicitação no sistema.	
	4. Altera o status da solicitação na base de dados para finalizada.
Restrições	Dados Inválidos no Formulário devem ser tratados.
Validações	

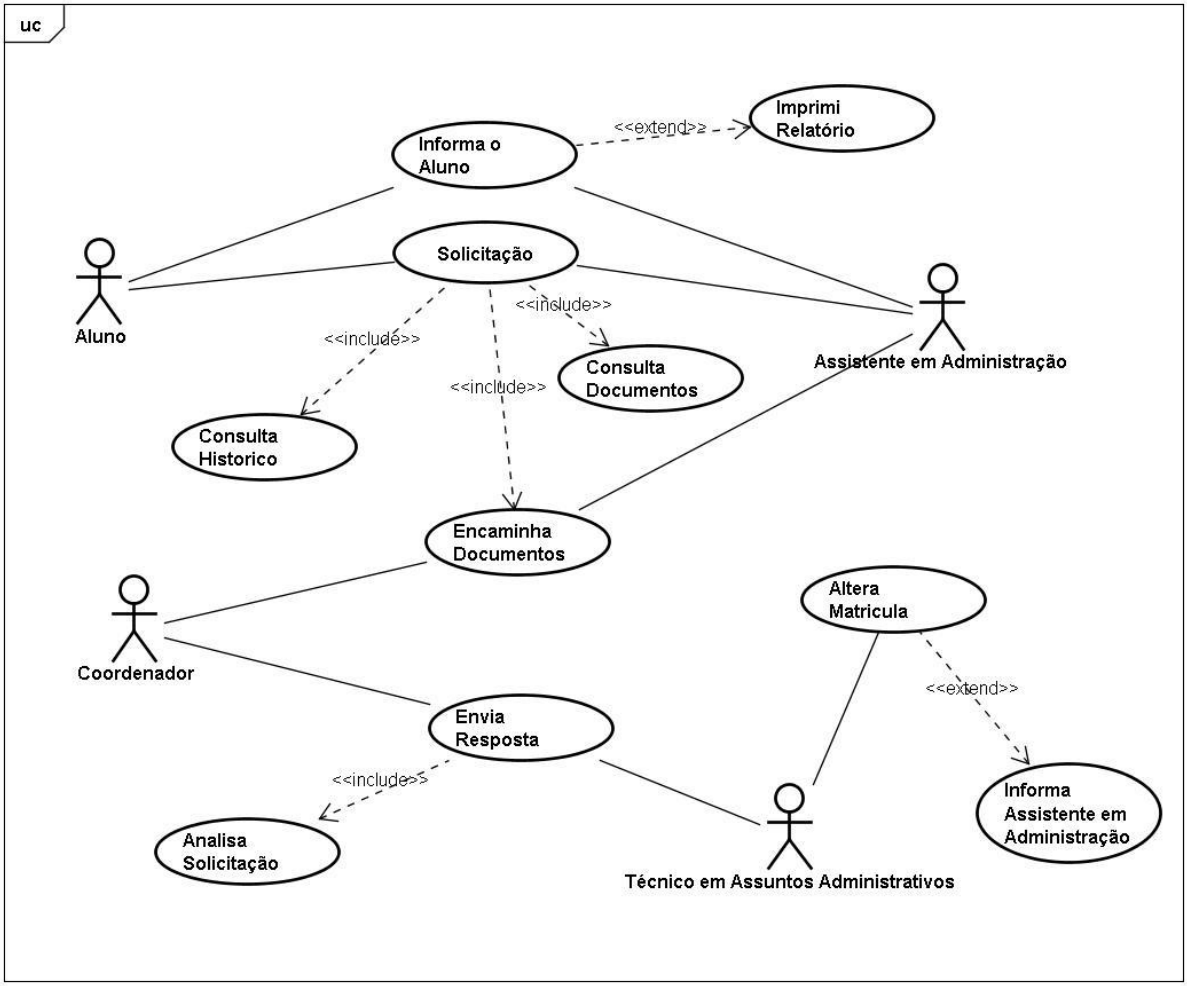
Fonte: Do autor

3.4 Diagrama de Casos de Uso

Com a análise do ambiente e a criação do enunciado do trabalho, identificou-se os atores e seus casos de uso (Figura 17).

Para o desenvolvimento de um conjunto de casos, as funções ou atividades realizadas por um ator foram listadas. Essas podem ser obtidas de uma lista de funções necessárias para o sistema. (PÁDUA, 2011, p.153).

Figura 18 - Diagrama Casos de Uso



Fonte: Do autor

3.5 Diagramas do Projeto

Com a criação e documentação dos casos de uso e com diagrama de casos de uso do projeto, construiu-se os diagramas que satisfazem nosso sistema, com base no estudo da UML 2.0 nos capítulos anteriores os diagramas seriam respectivamente nesta ordem, diagrama de classes, diagrama de sequência, diagramas de atividade e diagrama de pacotes.

Tomou-se como base o autor Horstmann, que dá ênfase a três objetivos da fase da modelagem ou projeto que são:

- Identificar as classes.
- Identificar as responsabilidades das classes.
- Identificar os relacionamentos entre as classes.

Esses objetivos fazem parte de um processo iterativo, leva a mudanças, descobertas e surgimento de novas classes e responsabilidades. (Horstmann, 2007, p. 55).

3.5.1 Diagrama de Classes

Segundo Horstmann projetista de software, uma boa regra prática para identificar as classes é buscar por substantivos funcionais, como por exemplo: (Horstmann, 2007, p. 55).

- Pessoas
- Disciplinas
- Alunos
- Matriculas
- Usuários
- Cursos

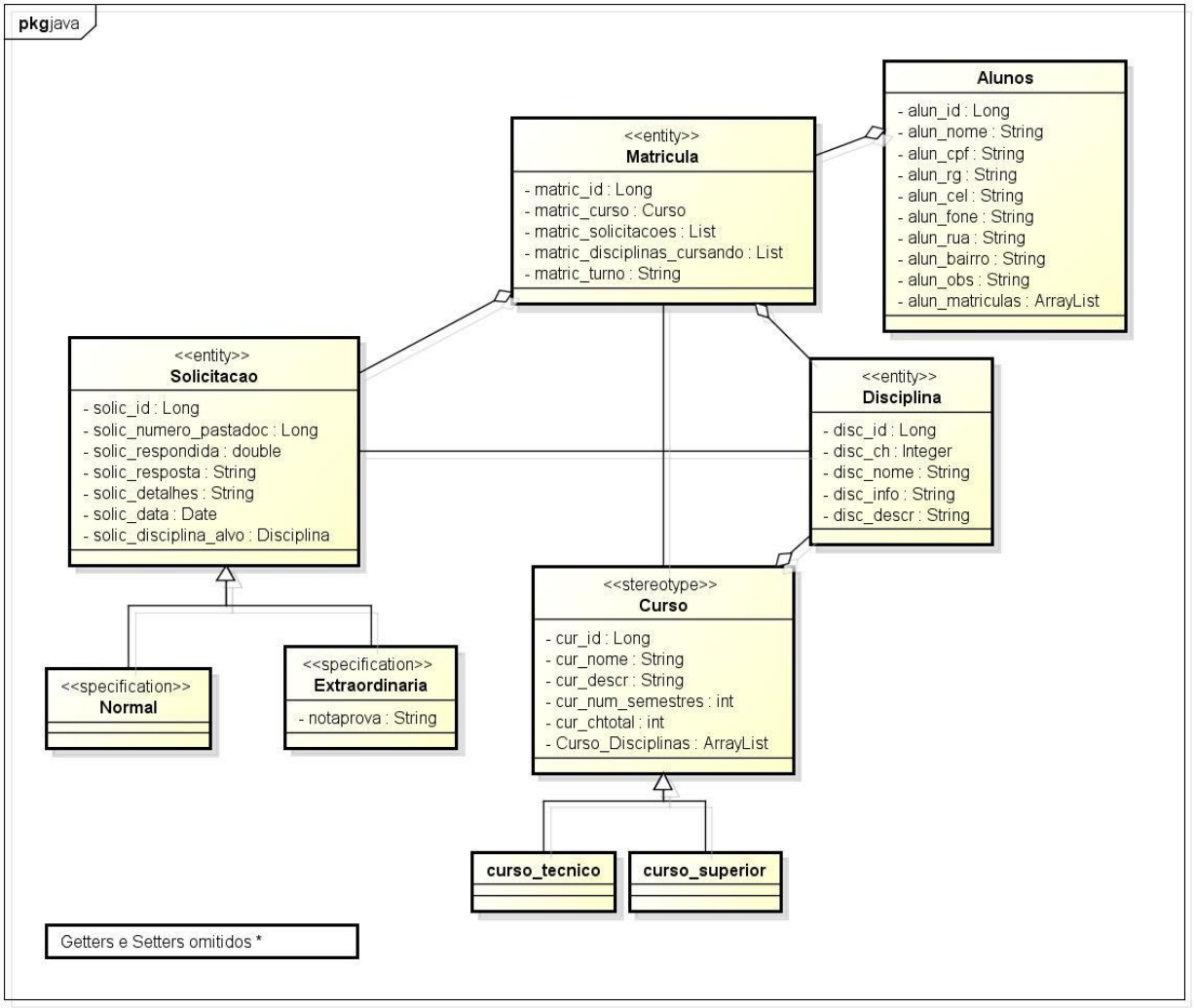
Com a descoberta destas classes que são óbvias, precisou-se mudar o foco e descobrir as classes que serão necessárias para efetuar as tarefas e que não são mapeadas como entidades, como por exemplo.

- Regras de Negócio
- Gerenciador de Login
- Impressão
- Gerenciador de Contas
- Validador
- Scanner

Existem algumas classes classificadas como “agentes”, são classes que possuem uma função e geralmente terminam por “er” ou “or’. (Horstmann, 2007, p. 55).

O resultado dessas práticas de Horstmann e a análise de requisitos realizada nos capítulos anteriores resultaram no diagrama de classes representado pela Figura 18 abaixo:

Figura 19 - Diagrama de Classes do Projeto

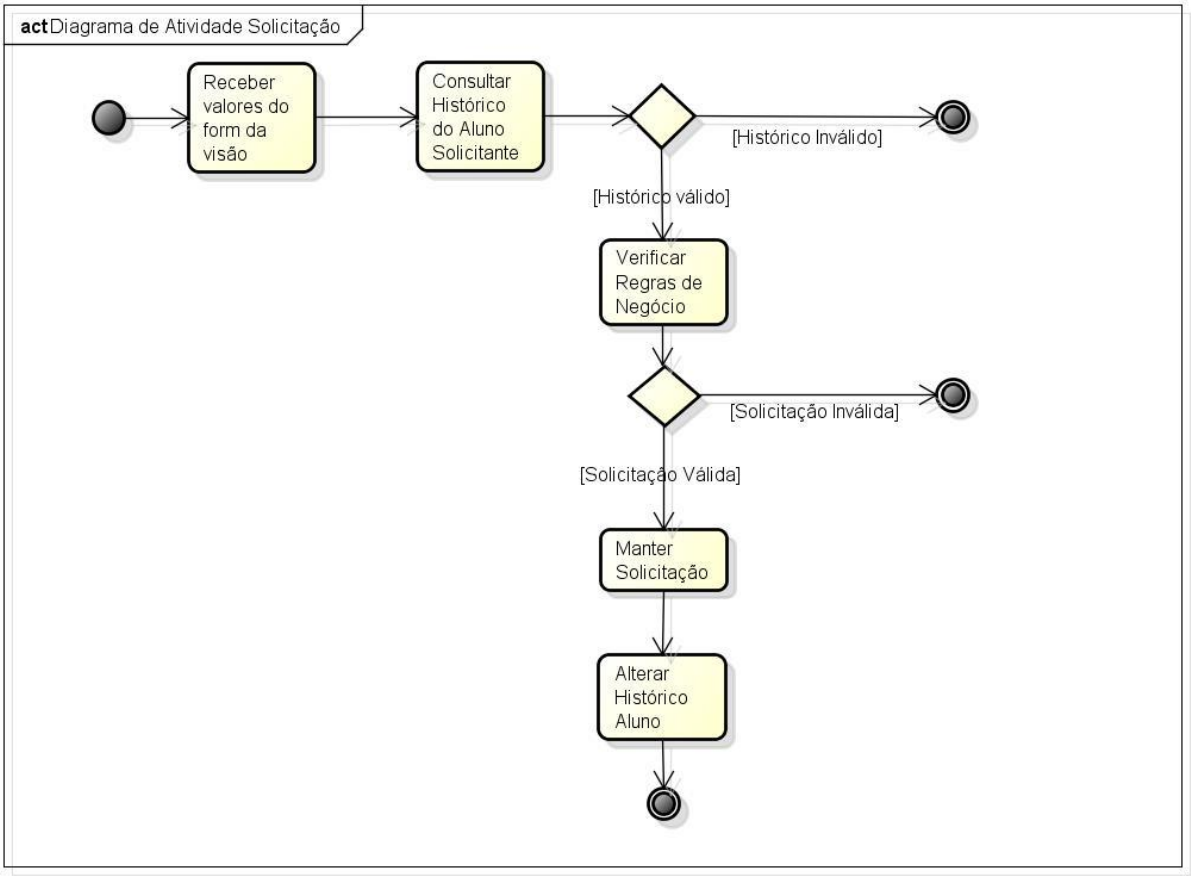


Fonte: Do autor

3.5.2 Diagrama de Atividade

O Diagrama de atividade do projeto tem como foco os passos do nosso problema principal a criação de uma solicitação ou Requisito R33 da tabela de requisitos, ilustrado na Figura 19 abaixo.

Figura 20 - Diagrama de Atividade Solicitação

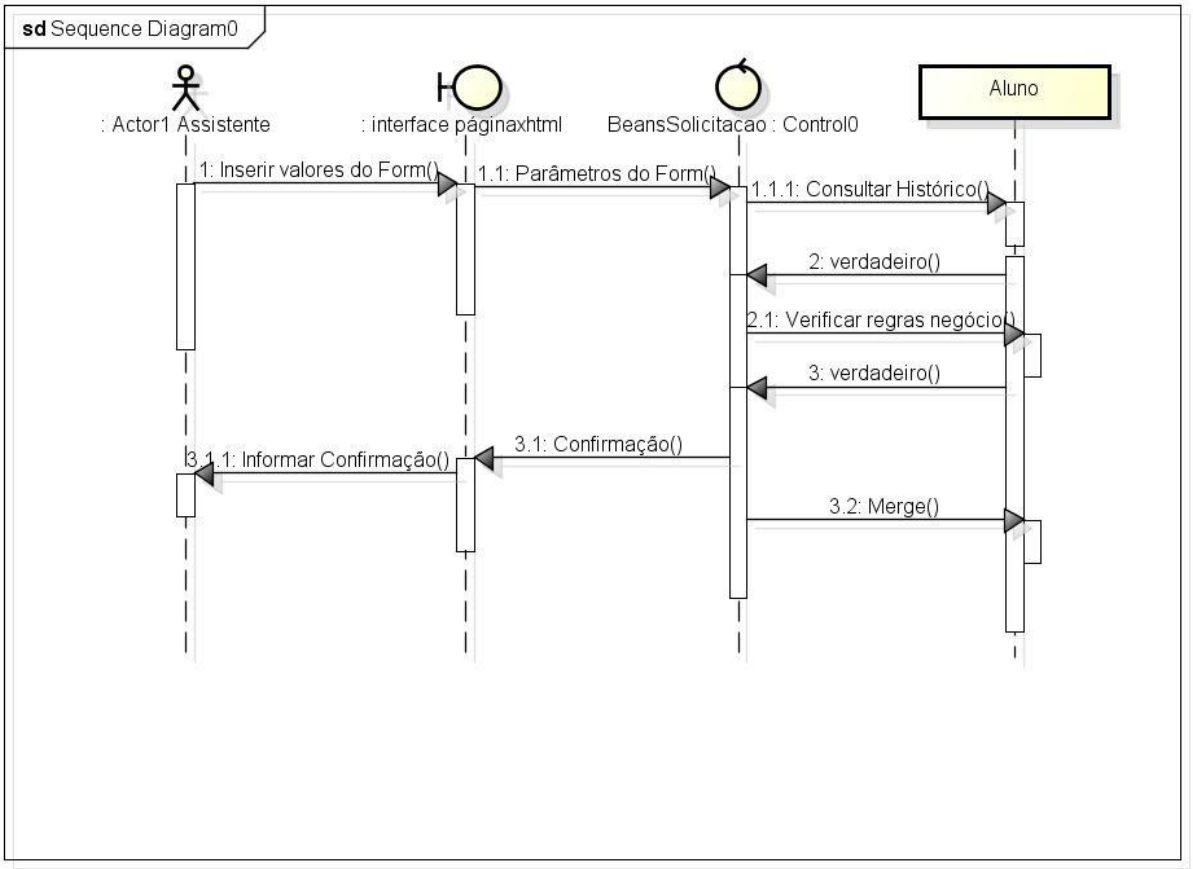


Fonte: Do autor

3.5.3 Diagrama de Sequência

Diagrama com as informações e ordem temporal do fluxo dos métodos para criação de uma solicitação, representado pela Figura 20.

Figura 21 - Diagrama de Sequência Solicitação



Fonte: Do autor

4 Implementação

Na implementação apresentou-se itens como recursos utilizados, tecnologias, linguagem, especificações e outras características do sistema do processo de desenvolvimento.

4.2 Padrões de Projeto

Como o desenvolvimento deste software não é completo e é aberto para continuação e aprimoramento do mesmo para outros programadores e alunos do instituto a fim de concretizar um possível uso no instituto, adotou-se práticas de organização, documentação e reutilização de código com os chamados padrões de projeto.

Com base nos autores Erich e Gamma, pesquisou-se e utilizou-se alguns padrões importantes para manter a organização e evitar problemas específicos como o re-projeto e requisitos futuros.

Erich, Gamma (2000, p. 17) cita que projetar software orientado a objetos é complicado, você deve identificar objetos, classes, granularidade, interfaces, hierarquias de herança e estabelecer a relação entre todos eles. O sistema deve ser especifico na resolução do problema, mas genérico ao ponto de satisfazer mudanças futuras e novos requisitos. A origem dos padrões de projeto é simples, quando encontra-se uma boa solução para um determinado problema especifico nós reutiliza-se repetidamente em vários projetos diferentes, pode-se dizer que essa solução estudada e aprimorada torna-se mais adiante um padrão de projeto.

4.2.1 Singleton

Garantir que uma classe tenha somente uma instância e fornecer um ponto global acesso para mesma. (Erich, Gamma, 2000, p. 130).

O padrão singleton oferece muitos benefícios: (Erich, Gamma (2000, p. 131)

- Acesso controlado a instância única: Podemos ter controle total de quando o cliente acessa essa única instancia.
- Espaço de nomes reduzido: Diminui o número de variáveis globais e evita a poluição de código.
- Permite um número variável de instancias: Podemos alterar o número de instancias que o singleton permite, somente alterando a sua classe.

O principal motivo do uso deste padrão refere-se diretamente a classe do Hibernate responsável pela criação de uma *factory* relacionada ao nosso banco de dados, devemos garantir que só seja instanciada uma factory em todo sistema e dar-mos um ponto de acesso global a ela.

Implementou-se esse padrão na classe do projeto *Hibernate.Util*, representada pela figura x.

4.3 Tecnologias

O S.A.D é um sistema-web orientado a objetos, com isso utilizou-se alguns frameworks, bibliotecas e ferramentas na sua fase de implementação com linguagem orientada a objetos.

A análise de requisitos resultou em um sistema complexo e tratando que futuramente o Instituto poderá utiliza-lo, tomou-se certas medidas como a adoção dessas tecnologias e práticas para torna-lo acessível para outros programadores e colaboradores.

4.3.1 Linguagem Java

Utilizou-se a linguagem programação Java, pois o autor trabalha com essa linguagem e teve aprendizagem de origem na disciplina de TOO (Tecnologia de Orientação a Objetos) cursada no Instituto.

4.3.2 Apache Tomcat

A escolha do servidor deve-se a simplicidade de configuração, utilizou-se o servidor Tomcat 7.0.1 provido pelo servidor lampp de licença livre. É importante destacar que versões anteriores a 7.0.1 do Tomcat não suportam o projeto, pois necessitam de importações de bibliotecas essências para o funcionamento correto do JSF.

4.3.3 Servidor de Dados Mysql

O projeto foi desenvolvido com persistência, isto é, mantendo os dados salvos em um banco. O servidor de dados escolhido foi o Mysql também oferecido pelo *lampp* de fácil gerenciamento e acesso, todas as *querys* executadas tem origem do Hibernate na linguagem (HQL) compatível com o mesmo.

4.3.4 Hibernate

Com o estudo da UML, atores e classes identificadas, deparou-se com a tarefa de modelar o banco de dados, substituiu-se essas funções por usar uma persistência automatizada com Hibernate.

O Hibernate é um framework ORM (objeto – relacional), que oferece persistência automatizada dos objetos em uma aplicação Java para um banco de dados relacional, utilizando de marcações que identificam o mapeamento entre os objetos e o banco de dados (King, 2007, p. 29).

Em termos de produtividade o Hibernate elimina tarefas tediosas e triviais relacionadas à persistência de dados e nos deixa concentrar esforços no problema de negócio (King, 2007, p. 29).

Enfrentou-se muitas dificuldades com Hibernate pelo fato de que existe um paradigma no objeto/relacional e o surgimento de disparidades onde certas soluções são visivelmente melhores para aplicação Java, mas ao mesmo tempo deficientes

para o banco e vice-versa, alguns exemplos práticos seriam assuntos como o mapeamento de heranças.

Utilizou-se Hibernate nas classes do pacote DAO, para a criação da persistência de todas as classes de entidades (CRUD), que consiste em quatro métodos básicos utilizados em banco de dados relacionais: *Create, Read, Update, Delet*e, a figura x representa um exemplo do uso na classe AlunosDAO.

4.3.5 JSF Framework

Como nosso projeto utiliza padrão MVC, o JSF (*Java Server Faces*) encaixou-se perfeitamente no desenvolvimento do projeto. O JSF é um framework de componentes para criação de interfaces web, orientado a eventos, com linguagem Java. Possui diversos serviços e é facilmente usado em um container autônomo como o Tomcat (Horstmann, 2007, p. 4).

A figura 22 ilustra uma visão de alto nível do framework JSF.

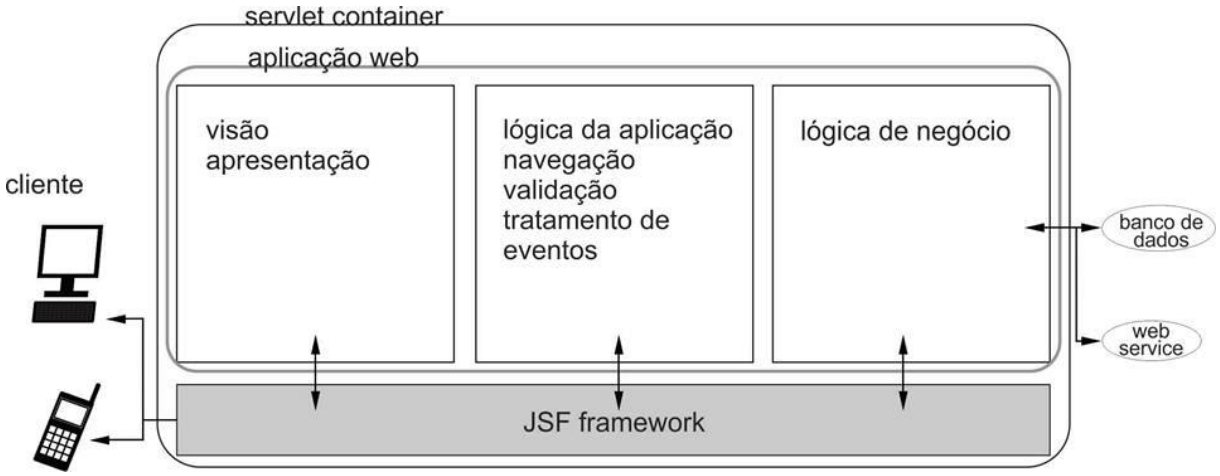
7.2.5.1 Características

Devido sua arquitetura MVC, o JSF é capaz de ligar componentes do modelo diretamente a uma propriedade da visão com o uso dos chamados *backbeans*.

Ele pode reagir a eventos do usuário, como por exemplo, um botão, uma ação do mouse ou do teclado. Oferece-nos inúmeras utilidades como validação e manipulação de erros, possibilitando a vinculação de regras de tipo a campos como, por exemplo, “Nome é obrigatório”, “Só números”, etc (Horstmann, 2007, p. 14).

A característica mais interessante é a capacidade de criarmos componentes sofisticados e bibliotecas que adicionam estilos, funcionalidades diferentes e enfeites ao JSF.

Figura 22 - JSF - Representação em alto nível



Fonte: Horstmann, 2007, p.24

Assim o JSF implementa o padrão MCV (modelo- controle- visão).

4.4 Requisitos Atendidos

Neste capítulo relatou-se cada requisito da tabela de requisitos do Capitulo 3.

Baseados na tabela de requisitos, enumerou-se os requisitos mais importantes a fim focar os requisitos vitais e essências para a criação de um solicitação.

Ignorou-se os requisitos opcionais como, calendário, visual, impressão, ordenação, enfim, os requisitos estéticos.

4.4.1 Requisitos atendidos da entidade Aluno

A tabela 15 a seguir corresponde aos requisitos que envolvem a entidade Alunos, requisitos R1 ao R9:

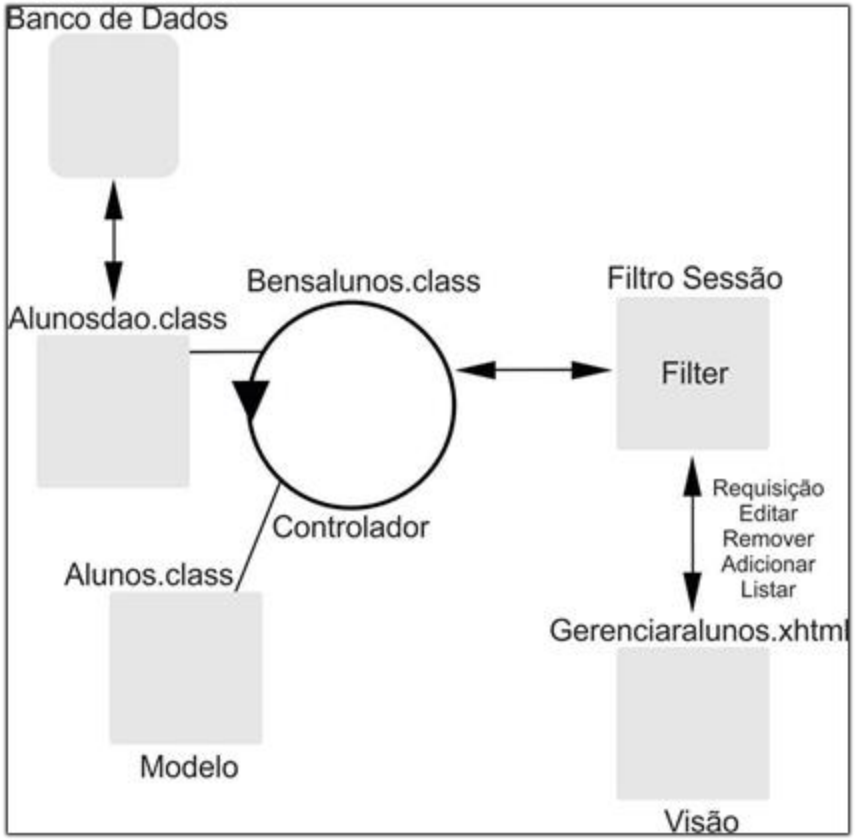
Tabela 15 - Requisitos Atendidos Aluno

ID	Nome	Prioridade	Estado	Status	Categoria
R1	Remover aluno	Essencial	Original	Implementado	Interação
R2	Inserir aluno	Essencial	Original	Implementado	Interação
R3	Atualizar aluno	Essencial	Original	Implementado	Interação
R4	Listar alunos	Essencial	Original	Implementado	Interface de usuário
R5	Tela controle de alunos	Essencial	Original	Implementado	Interface de usuário
R6	Mostrar as solicitações do Aluno	Essencial	Derivado	Implementado	Interface de usuário
R7	Imprimir relatório das informações do Aluno	Opcional	Derivado	Incompleto	Interação
R8	Enviar e-mail ao aluno	Opcional	Derivado	Implementado	Interação
R9	Mostrar todas Informações do Aluno	Essencial	Derivado	Implementado	Interface de usuário

Fonte: Do autor

A figura 23 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos:

Figura 23 - Fluxo Alunos



Fonte: Do autor

4.4.1.1 Descrição

Para um melhor entendimento de como o requisito foi atendimento e um visão mais próxima do código, enumerou-se cada requisito e suas iterações:

- R1 ao R5 – Criou-se uma página xhtml, “Gerenciaralunos.xhtml”, contendo um formulário com as campos das informações do aluno com função adicionar. Logo abaixo uma lista contendo todos os alunos, referente ao R4, contendo logo ao lado de cada cadastro as opções de interação editar e excluir, todos com diálogo de confirmação.

- As interações chamam os respectivos métodos no Bean, “Beanalunos.class”, passando os devidos parâmetros para ação.

- Por sua vez o BeanAlunos utiliza da classe DAO, “AlunosDAO”, para efetuar as alteração e atualizar o banco.

4.4.2 Requisitos atendidos da entidade Matrículas

A tabela 16 a seguir corresponde aos requisitos que envolvem a entidade Matriculas, requisitos R10 ao R16:

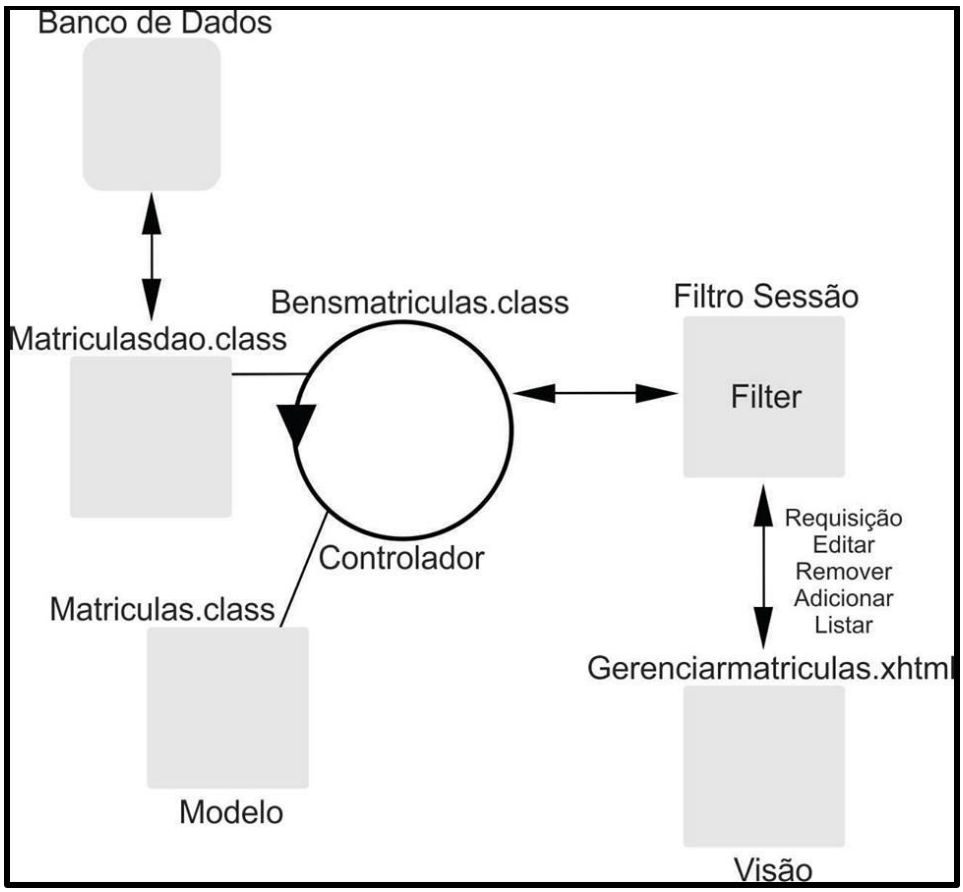
Tabela 16 - Requisitos Atendidos Matrículas

ID	Nome	Prioridade	Estado	Status	Categoria
R10	Remover matrícula	Essencial	Original	Implementado	Interação
R11	Inserir matrícula	Essencial	Original	Implementado	Interação
R12	Atualizar matrícula	Essencial	Original	Implementado	Interação
R13	Listar matrículas	Essencial	Original	Implementado	Interface de usuário
R14	Tela controle de matrícula	Essencial	Original	Implementado	Interface de usuário
R15	Mostrar as solicitações da matrícula	Essencial	Derivado	Incompleto	Interface de usuário
R16	Listar cursos da matrícula	Essencial	Derivado	Incompleto	Interface de usuário

Fonte: Do autor

A figura 24 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos:

Figura 24 - Fluxo Matrículas



Fonte: Do autor

4.4.2.1 Descrição

- R10 ao R16 – Criou-se uma página xhtml, “Gerenciarmatriculas.xhtml”, contendo um formulário com as campos das informações do aluno com função inscrição. Logo abaixo uma lista contendo todos os alunos e suas matriculas, referente ao R10, contendo logo ao lado de cada cadastro as opções de interação editar e excluir, todos com dialogo de confirmação.

4.4.3 Requisitos atendidos da entidade Cursos

A tabela 17 a seguir corresponde aos requisitos que envolvem a entidade Cursos R17 ao R25:

Tabela 17 - Requisitos Atendidos Curso

ID	Nome	Prioridade	Estado	Status	Categoria
R17	Remover curso	Essencial	Original	Implementado	Interação
R18	Inserir curso	Essencial	Original	Implementado	Interação
R19	Atualizar curso	Essencial	Original	Implementado	Interação
R20	Listar cursos	Essencial	Original	Implementado	Interface de usuário
R21	Tela controle de cursos	Essencial	Original	Implementado	Interface de usuário
R22	Mostras disciplinas do curso	Essencial	Derivado	Incompleto	Interface de usuário
R23	Listar alunos do curso	Essencial	Derivado	Incompleto	Interface de usuário
R24	Inscrever aluno no curso	Essencial	Original	Implementado	Interação
R25	Remover aluno do curso	Essencial	Original	Implementado	Interação

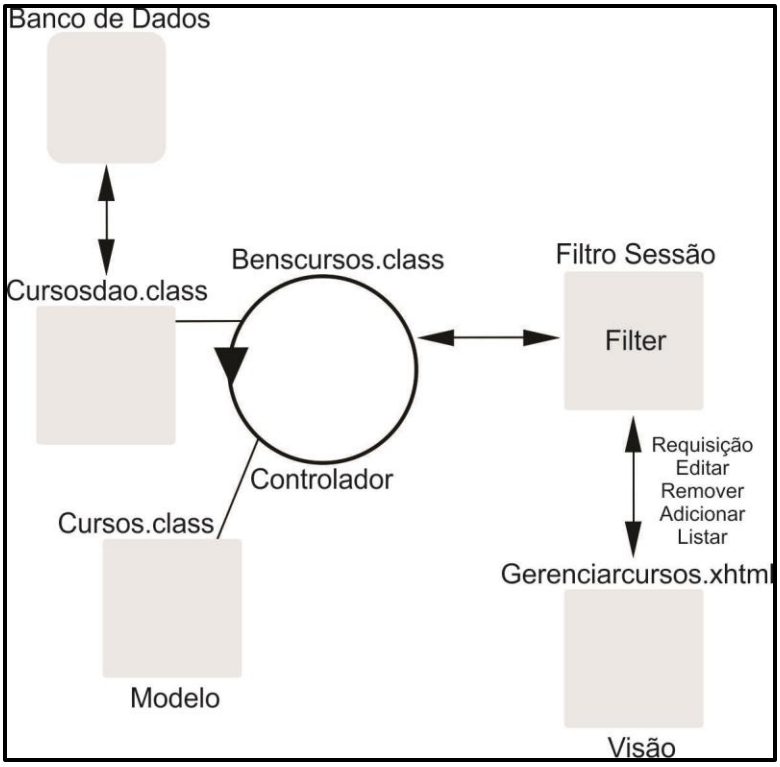
Fonte: Do autor

4.4.3.1 Descrição

- R17 ao R25 – Criou-se uma página xhtml, “Gerenciarcursos.xhtml”, contendo um formulário com as campos das informações do curso com função adicionar. Logo abaixo uma lista contendo todos os cursos, referente ao R17, contendo logo ao lado de cada cadastro as opções de interação editar e excluir, todos com dialogo de confirmação.

A figura 25 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos:

Figura 25 - Fluxo Cursos



Fonte: Do autor

4.4.4 Requisitos atendidos da entidade Disciplinas

A tabela 18 a seguir corresponde aos requisitos que envolvem a entidade Disciplinas, requisitos R26 ao R31:

Tabela 18 - Requisitos Atendidos Disciplinas

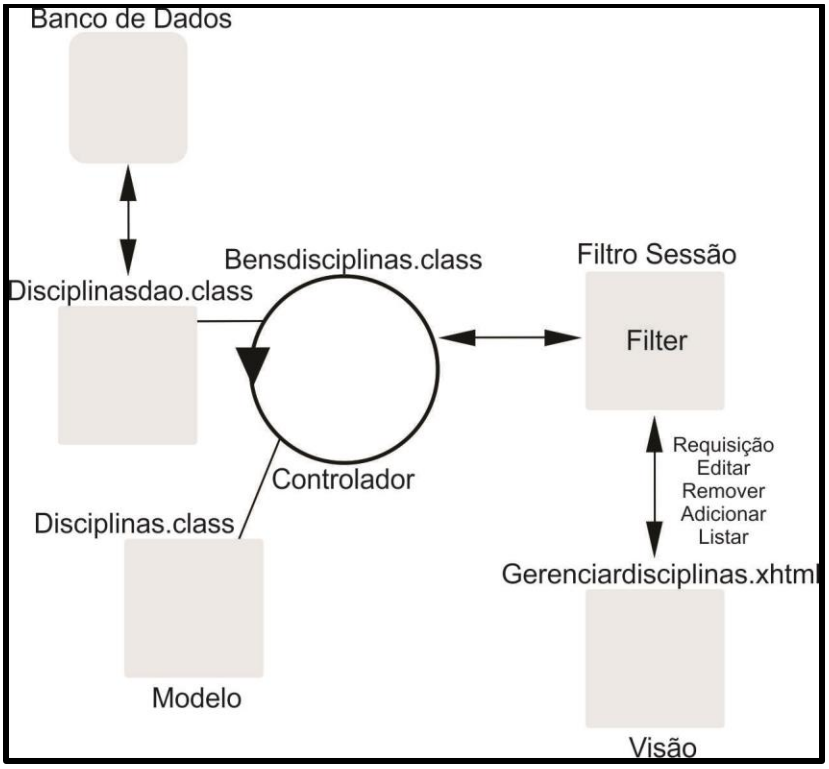
ID	Nome	Prioridade	Estado	Status	Categoria
R26	Remover disciplina	Essencial	Original	Implementado	Interação
R27	Inserir disciplina	Essencial	Original	Implementado	Interação
R28	Atualizar disciplina	Essencial	Original	Implementado	Interação
R29	Listar disciplina	Essencial	Original	Implementado	Interface de usuário
R30	Tela controle de	Essencial	Original	Implementado	Interface de usuário

	disciplina				
R31	Buscar disciplinas	Essencial	Derivado	Incompleto	Interface de usuário

Fonte: Do autor

A figura 26 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos:

Figura 26 - Fluxo Disciplinas



Fonte: Do autor

4.4.4.1 Descrição

- R26 ao R31 – Criou-se uma página xhtml, “Gerenciardisciplinas.xhtml”, contendo um formulário com as campos das informações da disciplina com função adicionar. Logo abaixo uma lista contendo todas as disciplinas, referente ao R26, contendo logo ao lado de cada cadastro as opções de interação editar e excluir, todos com diálogo de confirmação.

4.4.5 Requisitos atendidos da entidade Solicitações

A tabela 19 a seguir corresponde aos requisitos que envolvem a entidade Solicitações R32 ao R40:

Tabela 19 - Requisitos Atendidos Solicitações

ID	Nome	Prioridade	Estado	Status	Categoria
R32	Remover Solicitação	Essencial	Original	Implementado	Interação
R33	Inserir Solicitação	Essencial	Original	Implementado	Interação
R34	Atualizar Solicitação	Essencial	Original	Implementado	Interação
R35	Listar Solicitação	Essencial	Original	Implementado	Interface de usuário
R36	Tela Controle de Solicitações	Essencial	Original	Implementado	Interface de usuário
R37	Buscar Solicitação	Essencial	Derivado	Incompleto	Interface de usuário
R38	Responder Solicitação	Essencial	Derivado	Implementado	Interação
R39	Encerrar Solicitação	Essencial	Derivado	Incompleto	Interação
R40	Responsar Solicitação mediante avaliação	Essencial	Derivado	Incompleto	Interação

Fonte: Do autor

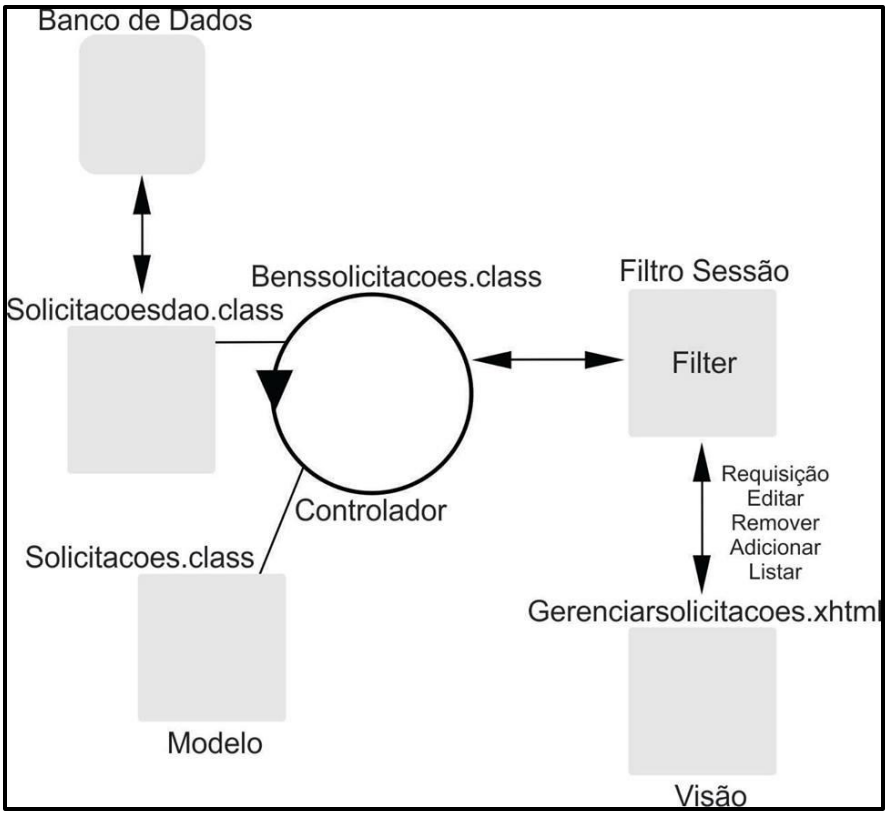
A figura 27 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos.

4.4.5.1 Descrição

- R32 ao R40 – Criou-se uma página xhtml, “GerenciarSolicitacoes.xhtml”, contendo um formulário com as campos das informações da solicitação com função criar apontando através de uma lista qual aluno está solicitando. Logo abaixo uma lista contendo todas as solicitações, referente ao R35, contendo

logo ao lado de cada cadastro as opções de interação editar e excluir, todos com diálogo de confirmação.

Figura 27 - Fluxo Solicitações



Fonte: Do autor

4.4.6 Requisitos atendidos da entidade Usuários

A tabela 20 a seguir corresponde aos requisitos que envolvem a entidade Usuários, requisitos R41 ao R49:

Tabela 20 - Requisitos Atendidos Usuários

ID	Nome	Prioridade	Estado	Status	Categoria
R41	Remover Usuário	Essencial	Original	Implementado	Interação
R42	Inserir Usuário	Essencial	Original	Implementado	Interação
R43	Atualizar Usuário	Essencial	Original	Implementado	Interação

R44	Listar Usuário	Essencial	Original	Implementado	Interface de usuário
R45	Tela Controle de Usuário	Essencial	Original	Implementado	Interface de usuário
R46	Buscar Usuário	Essencial	Derivado	Implementado	Interface de usuário
R47	Alterar Nível	Essencial	Derivado	Incompleto	Interação
R48	Histórico do Usuário	Essencial	Derivado	Incompleto	Interação
R49	Mostrar Solicitações criadas pelo Usuário	Essencial	Derivado	Implementado	Interação

Fonte: Do autor

A figura 29 ilustra o gráfico com o fluxo e detalhamento dos componentes do projeto envolvidos nesses requisitos.

4.4.6.1 Descrição

- R41 ao R45 – Criou-se uma página xhtml, “Gerenciarusuarios.xhtml”, contendo um formulário com as campos das informações do usúario com função adicionar. Logo abaixo uma lista contendo todos os usuarios, referente ao R45, contendo logo ao lado de cada cadastro as opções de interação editar e excluir, todos com diálogo de confirmação.

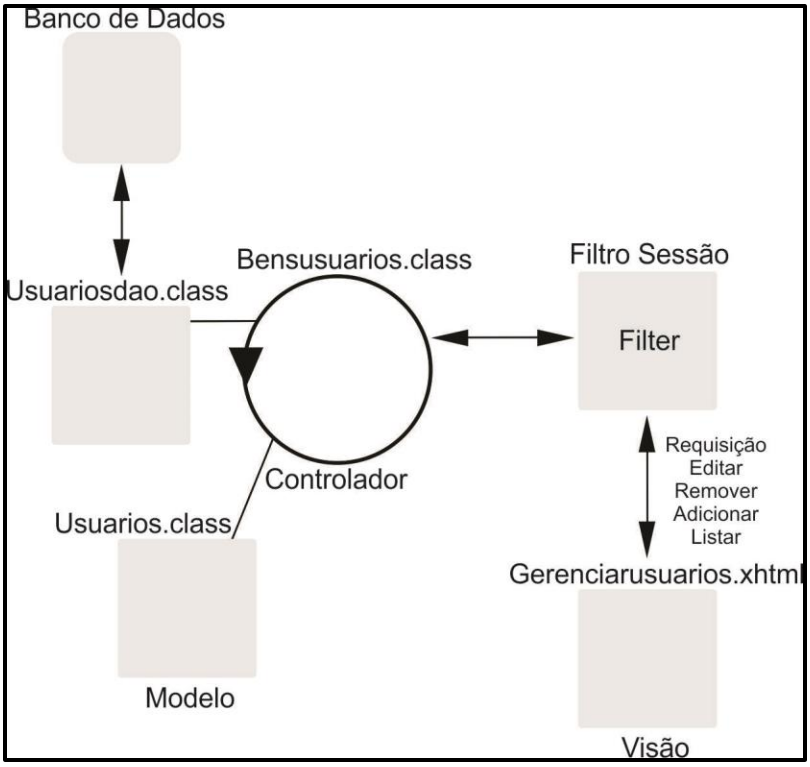
4.4.7 Menu do Usuário

Desenvolveu-se uma interface de usuário, hiperlink de um conjunto de páginas xHtml, contendo e reunindo todos os requisitos anteriormente mencionados representado pela figura 28.

Figura 28 - Menu do Usuário

Fonte: Do autor

Figura 29- Fluxo Usuários



Fonte: Do autor

5 Considerações Finais

Neste trabalho apresentou-se as três etapas básicas para implementação de um protótipo intitulado SAD (Sistema de Aproveitamento de Disciplinas). Análise de requisitos, projeto e a implementação. Baseados no estudo da engenharia de software, modelagem de sistema, UML 2.0 e tecnologias de implementação.

Destacando-se a importância de uma boa análise, um novo requisito pode sim, alterar toda a base do sistema. Passou-se por essas dificuldades em certos momentos como na modelagem do diagrama.

Implementou-se o protótipo com as tecnologias citadas e aplicou-se as regras da engenharia de software, que no momento é capaz de criar uma solicitação.

É referencial para estudos sobre projetos JSF, contém informações e classes muito comuns em todos os projetos como validação, sessão do usuário, padrões de projeto, estrutura em modelo MVC, frameworks recentes presentes no mercado de trabalho atual, devidamente utilizados.

Estudou-se cada tecnologia que foi utilizada, nos diagramas da UML 2.0, na análise de requisitos e nos frameworks mencionados, todo esse conhecimento será utilizado em trabalhos futuros.

O Link para download do projeto, diagramas e vídeo tutorias:

https://www.dropbox.com/sh/1w9sa23gte3onc7/o9_JmO109t

6 Referências

BEZERRA, Eduardo. *Princípios de análise e projeto de sistemas com UML.* Rio de Janeiro: Eksevier, 2007.

GUEDES, Gilleanes T. A. *UML 2* : uma abordagem prática. 2 ed. São Paulo: Novatec Editora, 2011.

HORSTMANN, Cay. *Core Java Server Faces*. 2 ed. Do original Core Java Server Faces Copyright: Editora Alta Books, 2007.

HORSTMANN, Cay. *Padrões e projeto orientados a objeto*. 2 ed. Porto Alegre: Bookman, 2007.

MEDEIROS, Ernani Sales de. *Desenvolvendo software com UML 2.0: definitivo.* São Paulo: Pearson Makron Books, 2004.

PÁDUA PAULA FILHO, Wilson de. *Engenharia de Software*: fundamentos, métodos e padrões. 3 ed. Rio de Janeiro: LTC, 2011.

PRESSMAN, Roger S. *Engenharia de software. 6 ed.* Porto Alegre: AMGH, 2010.

PRESSMAN, Roger S. *Engenharia de software.* São Paulo: Pearson Makron Books, 1995.

SOMMERVILLE, Ian. *Engenharia de software.* 8 ed. São Paulo: Pearson Addison Wesley, 2007.

7 Anexos

ANEXO A: Modelo de solicitações Gerais

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE Campus Passo Fundo	SOLICITAÇÃO Nº 001/2012
Nome do solicitante:	Servidor:
Curso: Turma: Turno:	Data:___/___/_____
Tipo de solicitação ( ) Abono de Faltas ( ) Ajuste de disciplina ( ) Aproveitamento de estudo ( ) Atestado ( ) Autorização ( ) Cancelamento ( ) Certificado ( ) Diploma ( ) Habilitação para estágio curricular ( ) Reabertura de Matrícula ( ) Trancamento ( ) Troca de turno ( ) Outros Especificar:_____ _____ _____ Motivo:_____	Encaminhado para: _____ Assinatura: _____

Picote

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE Campus Passo Fundo	SOLICITAÇÃO Nº 001/2012

Nome do solicitante:			Servidor:
Curso:	Turma:	Turno:	Data:___/___/_____

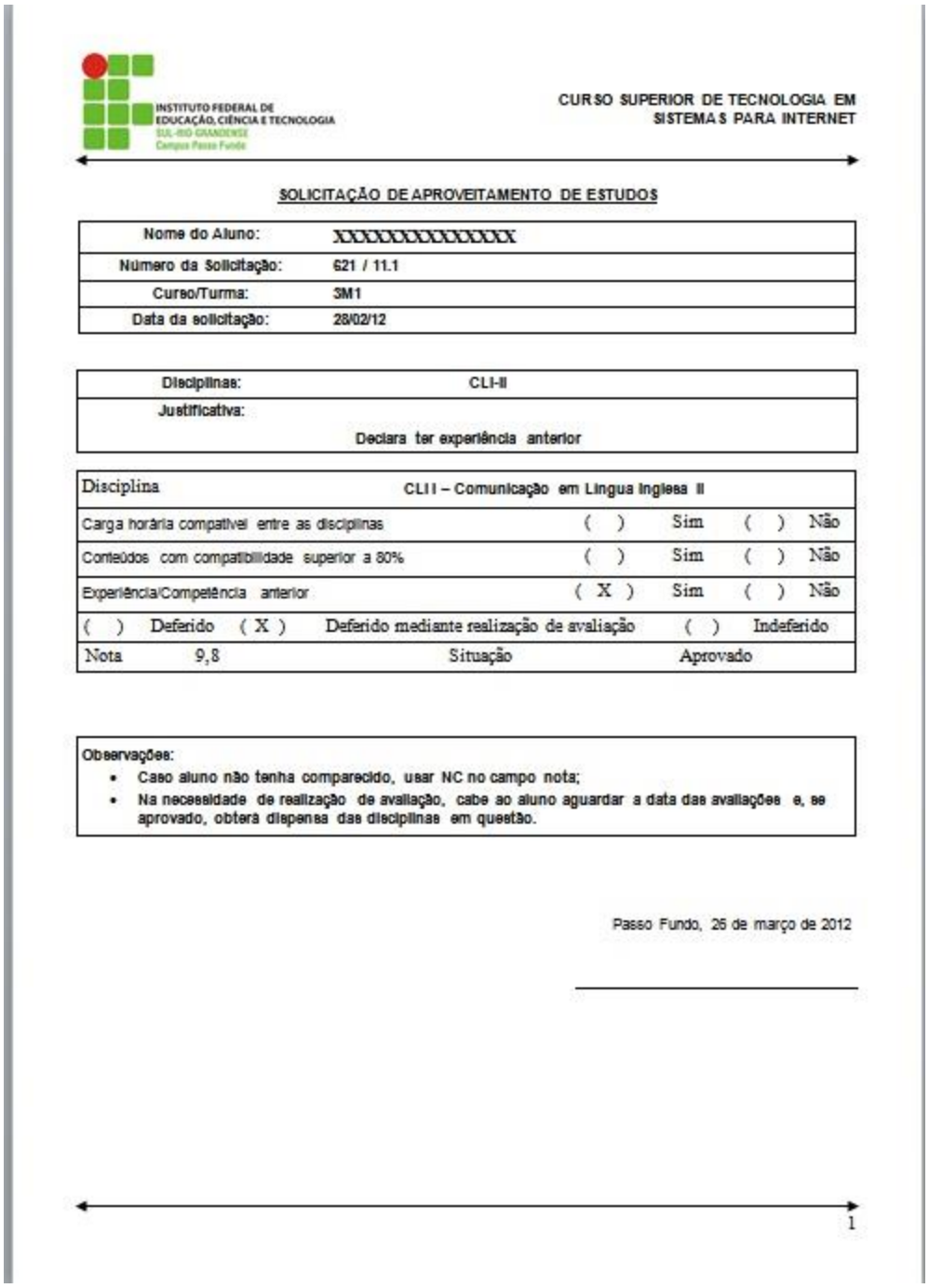
Tipo de solicitação	
( ) Abono de Faltas ( ) Ajuste de disciplina ( ) Aproveitamento de estudo ( ) Atestado ( ) Autorização ( ) Cancelamento ( ) Certificado + ( ) Diploma ( ) Habilitação para estágio curricular ( ) Reabertura de Matrícula ( ) Trancamento ( ) Troca de turno ( ) Outros Especificar:_____ _____ Motivo:_____ Assinatura: Data:__	Parecer/Despacho: _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ Assinatura: _____ Data: ___/___/_____ Situação Q-Acadêmico: ( ) Atualizado

Picote

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE Campus Passo Fundo	SOLICITAÇÃO Nº 001/2012

Solicitação:	Curso:	Data: ___/___/_____	Servidor:

ANEXO B – Solicitação de Aproveitamento



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

SOLICITAÇÃO DE APROVEITAMENTO DE ESTUDOS

Nome do Aluno:	XXXXXXXXXXXXXX
Número da Solicitação:	621 / 11.1
Curso/Turma:	3M1
Data da solicitação:	28/02/12

Disciplinas:	CLI-II
Justificativa:	Declara ter experiência anterior

Disciplina	CLI I – Comunicação em Lingua Inglesa II				
Carga horária compatível entre as disciplinas		( )	Sim	( )	Não
Conteúdos com compatibilidade superior a 80%		( )	Sim	( )	Não
Experiência/Competência anterior		(X)	Sim	( )	Não
( ) Deferido	(X) Deferido mediante realização de avaliação			( )	Indeferido
Nota 9,8	Situação		Aprovado		

Observações:
- Caso aluno não tenha comparecido, usar NC no campo nota;
- Na necessidade de realização de avaliação, cabe ao aluno aguardar a data das avaliações e, se aprovado, obterá dispensa das disciplinas em questão.

Passo Fundo, 26 de março de 2012

1