UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

EDVANIA VOTRI LANZARINI

# ABORDAGENS DE REUSO DE REQUISITOS: ESTUDO DE CASO COM SISTEMAS DE SOFTWARE PARA GESTÃO ELETRÔNICA DE DOCUMENTOS

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2016

EDVANIA VOTRI LANZARINI

# ABORDAGENS DE REUSO DE REQUISITOS: ESTUDO DE CASO COM SISTEMAS DE SOFTWARE PARA GESTÃO ELETRÔNICA DE DOCUMENTOS

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – CSTADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof[a]. MSc. Alessandra Bortoletto Garbelotti Hoffman.

MEDIANEIRA

2016

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

## TERMO DE APROVAÇÃO

# ABORDAGENS DE REUSO DE REQUISITOS: ESTUDO DE CASO COM SISTEMAS DE SOFTWARE PARA GESTÃO ELETRÔNICA DE DOCUMENTOS

Por

**Edvania Votri Lanzarini**

Este Trabalho de Diplomação (TD) foi apresentado às 09h10min do dia 11 de novembro 2016 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O acadêmico foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. MSc. Alessandra B. G. Hoffman
UTFPR – *Campus* Medianeira
(Orientador)

Prof. Dr. Claudio L. Bazzi
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Nicholas E. L. dos Santos
UTFPR – *Campus* Medianeira
(Convidado)

Prof. MSc. Jorge A. Junior
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

# RESUMO

LANZARINI, Edvania Votri. Abordagens de reuso de requisitos: estudo de caso com sistemas de software para gestão eletrônica de documentos. Trabalho de diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira, 2016.

O presente trabalho apresenta um estudo de caso, realizado com sistemas para gestão eletrônica de documentos, Birô de Digitalização e Digitaldoc - produtos da empresa Anix Sistemas LTDA, a proposta é aplicar abordagens de reuso de requisitos nesses produtos. O objetivo é auxiliar/acelerar as fases de elicitação e especificação de requisitos mantendo a qualidade do software e reduzindo custos tanto no desenvolvimento quanto no produto final, bem como demonstrar que o reuso de requisitos é realizado, em muitos casos, de maneira inconsciente. Nesse contexto são descritas as definições, aplicações e vantagens do reuso de requisitos em sistemas de software. Foram aplicadas duas abordagens de reuso de requisitos: analogia entre sistemas de software e reuso de requisitos. Para demonstrar a aplicação da primeira abordagem foi feita a comparação entre as atividades (cadastro e digitalização) realizadas pelos sistemas por meio do diagrama de atividades. Como resultado dessa comparação foi comprovado o reuso, ainda que parcial, pois apesar das atividades de cadastro e digitalização serem idênticas, o sistema Digitaldoc possui outras funcionalidades implementadas. Para a aplicação da segunda abordagem o trabalho foi baseado nos requisitos funcionais dos sistemas, Birô de Digitalização e Digitaldoc, isto é, a comparação direta entre esses requisitos. E como resultado foi comprovado o reuso de requisitos já que os requisitos funcionais de ambos os sistemas são idênticos. Também é perceptível a semelhança entre alguns requisitos não funcionais. Para realizar o estudo de caso foi necessário desenvolver os diagramas de classes e diagramas de atividades referentes aos softwares citados, auxiliando na melhor compreensão dos mesmos e demonstração da aplicação das abordagens de reuso.

Palavras-chave: Requisitos, Reuso, Software.

## ABSTRACT

LANZARINI, Edvania Votri, Requirement reuse approaches. An electronic documents management software system study case. Final graduation work (Systems Analysis and Development Technology), Federal Technology University of Paraná. Medianeira, 2016.

The presente work presentes a study case carried out in an electronic documents managemet software, Birô de Digitalização and Digitaldoc – Anix Sistemas Ltda company products. The proposal is to apply requirements reuse at these products. The objetive is to help/speed up the elicitation and requirements specification phases, keeping the software quality and reducing the development and final product cost, show that unconsciously requeriments reuse are made in many cases.That surrounding context settings advantages, and applications are described at requeriment reuse software systems. Two requeriment reuse approaches were applied: software systems analogy and requeriments reuse. To demonstrate the application of the first approach it was made the comparison between activities (registration and scanning) performed by the system through the activity diagram. As a result of the comparison reuse was partially proven, even though the registration and scanning activities are identical, Digitaldoc has other features implemented. The second approach was based at system functional requirements, Birô de Digitalização and Digitaldoc, this is, the direct comparison of these requirements. As a result, requeriments reuse was proven, even though the functional requirements from both systems are identical. It was perceived similarity at some non functionl requirements. To perform the study case it was necessary to develop class diagrams and activity diagrams from the cited softwares, this improved the understanding and demonstration of the approach reuse aplication.



Keywords: Requirements, Reuse, Software.

## AGRADECIMENTOS

Primeiramente sou grata a Deus, pois sua graça e misericórdia me permitiram concluir mais essa etapa de minha vida.

Agradeço a meus pais, Idevanio e Marlene, que sempre me apoiaram e não mediram esforços para me auxiliar independente das circunstâncias. A meu irmão, Heleno (in memoriam) que apesar de não estar presente para participar desse momento sempre acreditou que o mesmo chegaria, e minha irmã Elizabeth agradeço pelo apoio. Quero agradecer também meu esposo, Lucas, pelo amor, paciência, incentivo, carinho e apoio para a finalização desse trabalho e meus filhos, Mariah e Josué, pela paciência e compreensão nos momentos em que não pude estar com eles. E a todos os familiares que, de alguma maneira, me apoiaram.

Quero agradecer à minha orientadora, Alessandra Hoffman, que me auxiliou e acompanhou também nessa etapa final de conclusão do curso. Aos professores Fernando Schultz pela orientação e ajudas iniciais e Juliano Rodrigo Lamb pela revisão desse trabalho meu "muito obrigado". Bem como a todos os professores que contribuíram, durante a universidade, compartilhando seu conhecimento para minha formação.

E por fim, agradeço a todas as amigas pessoais e irmãs em Cristo, das quais sempre pude contar com o apoio, e aos colegas de classe pelo companheirismo durante o tempo de universidade.

## LISTA DE SIGLAS

| | |
|---|---|
| DBC | Desenvolvimento Baseado em Componentes |
| GOF | *Gang of Four* |
| RARE | *Reuse-Assisted Requirements Engineering* |
| LPS | Linhas de Produtos de Software |
| IDIOM | *Informal Document Interpreter Organiser and Manager* |
| UML | Linguagem de Modelagem Unificada |
| OMG | *Object Management Group* |
| RF | Requisitos Funcionais |
| RNF | Requisitos Não Funcionais |
| ISO | *International Organization for Standardization* |
| CBSE | Engenharia de Software Baseada em Componentes |
| EJB | *Enterprise Java Beens* |
| JavaEE | *Java Enterprise Edition* |

## LISTA DE FIGURAS

## LISTA DE QUADROS

**SUMÁRIO**

# 1 INTRODUÇÃO

"Infraestruturas e serviços nacionais são controlados por sistemas computacionais, e a maioria dos produtos elétricos possui um computador e um *software* que o controla" (SOMERVILLE, 2011). Portanto, a engenharia de software é essencial para o funcionamento da sociedade como um todo (SOMERVILLE, 2011).

Um software deve ser produzido com o objetivo de atender as especificações, ser validado para acatar a demanda do cliente e quando necessário evoluir aprovando as necessidades de mudanças. Para isso é fundamental a consulta ao usuário a fim de se obter a especificação do sistema. Cometer erros nessa fase podem comprometer o projeto e a implementação do mesmo (PRESSMAN, 2011).

No que diz respeito à construção de um software, é essencial o planejamento do sistema como um todo, ou seja, o que será feito e de que maneira. Com isso, o resultado final será satisfatório sem a necessidade de ajustes (BROOKS, 2003).

Segundo Lobo (2008) os usuários estarão satisfeitos com o produto se este realmente atender às suas necessidades, dessa maneira as funcionalidades do software estão diretamente ligadas às atividades que ele deve realizar.

Existem várias maneiras de construir um software, porém, em todos elas se faz necessária uma documentação adequada, o que é possível por meio da Engenharia de Software (*Software Engineering*).

Dentre as áreas da Engenharia de Software se pode citar a engenharia de requisitos, cujo objetivo é obter o conhecimento dos requisitos de negócio, requisitos não funcionais e requisitos de usuário para o sistema (SOMERVILLE, 2011).

Júnior (2010) define requisito como um componente do sistema cujo objetivo é alcançar determinado fim, ou ainda, condição ou capacidade de um software que deve ser implementada. Para Somerville (2011), requisitos podem ser definidos como os serviços que o sistema oferece, seu detalhamento/descrição e até as restrições de funcionamento. Resumem-se às necessidades do cliente, servem como um "espelho", deixando claro sua finalidade.

Um dos principais obstáculos que o profissional encontra na obtenção dos requisitos do sistema diz respeito a entender as informações obtidas as quais são repassadas pelo cliente. Nesse momento as funcionalidades e restrições do software devem ser definidas. Por este

motivo é fundamental que haja tempo para verificar e organizar o que foi registrado, culminando na obtenção correta de requisitos (PRESSMAN, 2011).

Júnior (2010) afirma que requisitos sempre mudam, por este fato é necessário ter o controle destes bem como de suas mudanças, por meio do gerenciamento[1] e da engenharia de requisitos[2].

De acordo com Somerville (2011), nesse momento surge a importância da engenharia de requisitos que define este como um processo iterativo, envolvendo quatro atividades de alto nível, que resulta no documento de requisitos do sistema. Essas atividades definem se o sistema é útil, permitem a descoberta dos requisitos, traduzem os mesmos em um modelo padronizado, e verificam se é de fato o que o cliente deseja no tocante ao sistema.

A engenharia de requisitos possui tarefas e técnicas que permitem o melhor entendimento dos requisitos do sistema resultando na compreensão das necessidades do cliente, ou seja, oferecendo a todas as partes um entendimento escrito do problema. De maneira geral, esta analisa as necessidades, avalia a viabilidade do projeto, negocia uma solução razoável, especifica a solução sem ambiguidades ou redundâncias, valida a especificação e gerencia as necessidades à medida que toma forma de sistema operacional (PRESSMAN, 2011).

A Engenharia de Software tem como um de seus objetivos, almejado por meio de pesquisas, atender a demanda do mercado, isto é, construir software com qualidade, reduzir custos e cumprir prazos. Com isso, a reutilização de software tem sido utilizada para aumentar a qualidade e produtividade do mesmo (PIMENTA, 1998).

A reutilização de software consiste em aproveitar produtos e/ou conhecimentos, criados por especialistas envolvidos no processo de desenvolvimento de software, que durante a construção de novas aplicações em ambientes, em sua maioria, diferentes daqueles para os quais as partes reutilizadas foram concebidas (KANG & FRAKES, 2005).

Segundo Pimenta (1998) a reutilização de software é uma técnica que visa aproveitar informações produzidas, durante o desenvolvimento, de softwares anteriores e cujo objetivo é reduzir esforços para desenvolver um novo sistema.

Reuso de software é definido como o processo de incorporar produtos existentes em um novo produto, tais como: código, especificações de requisitos e projeto, planos de teste e

[1] Disciplina responsável pela definição formal de metodologia relacionada a licitação, organização, classificação, gerenciamento, controle e documentação de requisitos.
[2] Idem a definição da referência 1.

conhecimento. Bibliotecas, *frameworks*, componentes e padrões de software são exemplos de técnicas utilizadas para o reuso (MARIANI, 2016).

O objetivo deste trabalho é apresentar as técnicas de reuso de requisitos demonstrando como essas podem auxiliar as fases iniciais do desenvolvimento de software (elicitação e especificação de requisitos).

## 1.1 OBJETIVO GERAL

Realizar um comparativo em reuso de requisitos utilizando um estudo de caso de sistemas de gestão eletrônica de documentos.

## 1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um referencial teórico sobre conceitos de famílias e/ou linhas de produtos, padrões de software, *frameworks*, componentes, bibliotecas de classes, requisitos e reuso;
- Desenvolver um referencial teórico sobre abordagens de reuso de requisitos em famílias de software;
- Realizar um estudo de caso envolvendo famílias de sistemas para gestão eletrônica de documentos, baseado nas abordagens pesquisadas;
- Elaborar um estudo comparativo entre as abordagens por meio dos resultados obtidos do estudo de caso proposto.

## 1.3 JUSTIFICATIVA

Atualmente as responsabilidades que eram delegadas a pessoas estão sendo "transferidas" para os softwares. Devido a esse fato é importante que estes sejam planejados com cautela, diminuindo riscos e custos no desenvolvimento de um projeto, ou seja, é importante produzir software com qualidade (SOMERVILLE, 2011).

No desenvolvimento de software pode ocorrer que cada indivíduo traduza a mesma ideia de maneiras diferentes, por este motivo é fundamental compreender o problema antes de desenvolver uma solução. "Projetar tornou-se uma atividade chave", o papel que anteriormente cabia a uma única pessoa passou a ser desempenhado por grandes equipes de software, devido à complexidade dos requisitos da tecnologia de informação (PRESSMAN, 2011).

Um dos maiores causadores dos problemas apresentados nos projetos envolvendo a Engenharia de Software é a falta de precisão na especificação dos requisitos. Um fato comum de ocorrer é a interpretação destes de maneira ambígua, simplificando aparentemente a implementação, e resultando em erros que podem até levar a reescrita dos requisitos.

A validação de requisitos é um processo de suma importância pelo fato de verificar se estes definem o sistema que o cliente realmente deseja, viabilizando a diminuição de custos e retrabalho no projeto. Já que as descobertas de erros nessa fase inicial geram correções mais acessíveis, se comparado à fase de desenvolvimento ou até na finalização do projeto, resultando no aperfeiçoamento do mesmo (SOMERVILLE, 2011).

Com o objetivo de examinar o documento de especificação, a validação de requisitos por meio da revisão técnica[3], visa garantir que estes tenham sido declarados com ausência de ambiguidade, que principalmente erros e inconsistências tenham sido detectados e que os artefatos estejam de acordo com os padrões estabelecidos para o processo, projeto e produto sendo avaliados quanto à qualidade durante essa etapa (PRESSMAN, 2011).

Portanto, utilizar reuso de requisitos visa auxiliar no aperfeiçoamento e melhora do processo da engenharia de requisitos, diminuindo o tempo de construção e aumentando a qualidade do produto desenvolvido. Também tem como objetivo dar assistência e orientação

[3] Revisores de Software são como um "filtro" para a gestão de qualidade, por isso as revisões são aplicadas em várias etapas durante o processo da engenharia de software e tem por finalidade revelar erros e defeitos que podem ser eliminados (PRESSMAN, 2011).

ao engenheiro de requisitos durante o processo de elicitação dos mesmos (ROLLAND e PRAKASH, 1999).

## 1.4 ESTRUTURA DO TRABALHO

O trabalho é composto por cinco capítulos, onde no primeiro é apresentada a introdução, os objetivos gerais e específicos bem como a justificativa.

O segundo capítulo descreve a fundamentação teórica com o conceito da Engenharia de Software, Linhas de produto, *Frameworks*, Componentes e Biblioteca de Classes, Padrões de Software, Engenharia de requisitos em reuso de requisitos.

No terceiro capítulo são apresentados os materiais e métodos da pesquisa.

No quarto capítulo é descrito o estudo de caso comparativo, isto é, a aplicação das abordagens de reuso nos sistemas de gestão eletrônica de documentos.

No quinto capítulo são detalhados os resultados e discussões, apresentando as conclusões do estudo de caso referentes ao trabalho.

# 2 FUNDAMENTAÇÃO TEÓRICA

As seções seguintes tratam de conceitos sobre Engenharia de Software, linhas de produto de software, *frameworks*, componentes e biblioteca de classes, padrões de software e engenharia de requisitos em reuso de requisitos.

## 2.1 ENGENHARIA DE REQUISITOS

Somerville (2011) afirma que a Engenharia de Software "é uma disciplina da engenharia cujo foco está em todos os aspectos da produção do software", incluindo as fases iniciais da especificação do sistema até a utilização, ou seja, sua manutenção. Surgiu com o objetivo de analisar questões de custo, prazo e confiança bem como as necessidades de todos os envolvidos no projeto, sejam eles clientes ou produtores de software.

Segundo a IEEE (1990) a Engenharia de Software é definida por meio da aplicação de engenharia ao software, isto é, consiste em uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e manutenção do software.

"Engenharia de Software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais" (BAUER APUD PRESSMAN, 2011).

A Engenharia de Software é uma tecnologia em camadas as quais podem ser citadas: foco na qualidade, processo, métodos e ferramentas (PRESSMAN, 2011).

Conforme a Figura 1 qualquer abordagem da engenharia, incluindo a Engenharia de Software, deve ter como fundamento o comprometimento organizacional com a qualidade. Dessa maneira o foco na qualidade é a chave de sustentação da Engenharia de Software. A camada de processo mantém as camadas de tecnologia coerentes, permitindo o desenvolvimento de software dentro do prazo bem como o gerenciamento dos mesmos. Os métodos, por sua vez, fornecem as informações técnicas para desenvolvimento de software incluindo tarefas como: comunicação, análise de requisitos, modelagem de projetos, construção de programa, testes e suporte. Por fim, as ferramentas fornecem suporte para o processo e para os métodos (PRESSMAN, 2011).


**Figura 1 - Camadas da Engenharia de Software**
**Fonte: Adaptado de Pressman (2011)**

O objetivo principal da Engenharia de Software segundo Júnior (2011) é utilizar princípios da engenharia, em um contexto geral, no desenvolvimento de software com a finalidade de aumentar a qualidade dos produtos oferecidos, diminuir riscos e custos relacionados criando processos eficazes para serem empregados nos ciclos de manutenção e desenvolvimento deste.

### 2.1.1 PROCESSOS DE SOFTWARE

De acordo com o dicionário Aurélio (2012) processo é definido como uma sequência contínua de fatos que apresentam certa unidade, ou que se reproduzem com certa regularidade, andamento, desenvolvimento.

No que diz respeito à Engenharia de Software, um processo tem como principal conceito a adaptação, permitindo que a equipe de software possa escolher o conjunto apropriado de ações e tarefas. O intuito é fazer entregas de software com qualidade e dentro do prazo satisfazendo as necessidades do cliente e consequentemente do usuário, ou seja, dos patrocinadores e futuros utilizadores do sistema (PRESSMAN, 2011).

Um processo de software é definido como um conjunto de atividades que se relacionam cujo objetivo é a fabricação de um produto de software (SOMERVILLE, 2011). Pressman (2011) afirma que processo tem sua definição como um “conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho.” O objetivo de uma **atividade** é obter um resultado amplo, por exemplo, por meio da comunicação com os

envolvidos no projeto. Uma **ação**, por sua vez, envolve um conjunto de tarefas cujo resultado é um artefato de software. E por fim, não menos importante, se tem uma **tarefa** à qual se concentra em um objetivo bem definido produzindo um resultado palpável. Com base nessa definição o autor afirma que processo de software é uma metodologia para desenvolver um software de alta qualidade.

Segundo Somerville (2011) atualmente existem diversos processos de software, apesar de serem distintos, todos devem incluir quatro atividades fundamentais para a Engenharia de Software, são elas:

Especificação de software: Definição de funcionalidades e restrições referentes ao software.

Projeto e implementação de software: Este deve ser produzido para atender as especificações.

Validação de software: O software deve ser validado para atender as necessidades do cliente.

Evolução do software: É importante que o software esteja apto caso exista necessidade de mudança requerida pelo cliente.

Pressman (2011) afirma que a existência de uma metodologia de processo mantém alicerçado o processo de engenharia de software auxiliando na completude deste. No caso de uma metodologia de processo genérica para a Engenharia de Software esta inclui cinco atividades:

Comunicação: Obter o conhecimento e a compreensão do projeto como um todo, por meio de diálogo tanto com o cliente para descobrir as necessidades deste quanto com a equipe envolvida no projeto, o que auxiliará na definição das funções e características do software.

Planejamento: Define principalmente um plano de trabalho contendo a descrição dos riscos prováveis, tarefas técnicas, recursos necessários e produtos resultantes.

Modelagem: A ideia principal é representar o software por meio de um modelo que defina suas necessidades fazendo referência ao projeto em questão e como consequência obter uma melhor compreensão do mesmo.

Construção: Atividade que associa o ato de gerar código seja de forma manual ou automatizada, e fazer testes com o intuito de refinar o software revelando erros na codificação.

Emprego: Considera-se o software resultante, ou seja, o momento da entrega ao cliente e este, por sua vez, avaliará o produto fornecendo uma opinião.

Somerville (2011) afirma que “um modelo de processo de software é uma representação abstrata de um processo de software”. Dessa maneira, cada modelo proporciona informações parciais sobre o processo. São exemplos de modelos de processo genéricos: cascata, desenvolvimento evolucionário, desenvolvimento formal de sistemas e desenvolvimento orientado a reuso.

Um modelo de processo de software define um fluxo de todas as atividades, ações e tarefas, o grau de iteração, os artefatos e a organização do trabalho a ser realizado. Nesse contexto, um modelo de processo de software fornece um guia específico para o trabalho de Engenharia de Software, cujo objetivo é tentar reduzir a “bagunça” existente no desenvolvimento de novos produtos de software. São exemplos de modelos de processos de software: cascata, incremental, evolucionário, concorrente unificado, ágil, desenvolvimento baseado em componentes, métodos formais e desenvolvimento baseado à aspectos (PRESSMAN, 2011).

## 2.2 LINHAS DE PRODUTO DE SOFTWARE

SEI (2013) afirma que linhas de produtos de software são definidas como um conjunto de sistemas que possuem características em comum. Tais características satisfazem as necessidades de um determinado segmento de mercado ou missão, e seu desenvolvimento é realizado de maneira preestabelecida. Por meio de linhas de produtos de software é possível realizar melhorias quanto ao custo, qualidade e produtividade, por exemplo. Por esse motivo estão emergindo como um paradigma de desenvolvimento de software.

“A *software product line* (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” (SEI, 2013).

LPS também chamadas de famílias de produtos de software são desenvolvidas a partir de um conjunto comum de ativos centrais, de forma sistemática cujo objetivo é satisfazer um segmento de mercado e correspondem a um conjunto de sistemas que contem características comuns e gerenciadas (CLEMENTS E NORTHROP, 2001).

De acordo com SEI (2013), a escassez de recursos de software é uma preocupação existente por parte das organizações.

Dessa maneira, utilizar linhas de produtos de software como uma estratégia de desenvolvimento pode permitir uma vantagem competitiva bem como produzir benefícios para as mesmas, os quais podem ser citados:

- Melhoria da qualidade;
- Melhoria na estimativa de custos;
- Desenvolvimento a custos reduzidos;
- Redução de esforços com manutenção;
- Capacidade de manutenção no mercado.

Queiróz (2009) afirma que o objetivo principal dessa técnica de desenvolvimento de software é a fabricação em larga escala. Por meio dela o reuso é possível em grande parte dos requisitos e também da arquitetura, evitando o esforço redundante em cada software independente. Ao desenvolver uma LPS se tem como foco o “todo”, ou seja, questões de arquitetura do sistema são projetadas de maneira a atender as necessidades de um conjunto de produtos fornecendo um contexto em que outros recursos são desenvolvidos para serem adicionados à linha e também para satisfazer os produtos.

Portanto, no que diz respeito ao desenvolvimento de linhas de produtos de software é importante reconhecer tanto as semelhanças quanto à variabilidade de um conjunto de software em determinado domínio;

A partir da engenharia de domínio são produzidos artefatos comuns (núcleo) aos produtos da linha, de maneira que os membros (produtos de software) dessa linha, por meio de engenharia de aplicação, possam ser desenvolvidos a partir de artefatos do núcleo, somando-se a este ou não artefatos que implementam as partes variáveis dos produtos de linha (QUEIRÓZ, 2009).

Para satisfazer as funcionalidades de um produto da linha vários artefatos são implementados, sendo que um determinado produto é composto por artefatos com diversas funcionalidades. Com isso, as diferentes escolhas de artefatos podem fazer com que dois produtos tenham funcionalidades diferentes ou as mesmas funcionalidades com características diferentes. Nesse contexto, o custo de desenvolvimento e manutenção que seria gasto em um

único produto é distribuído a todos os membros da linha de produtos (SHIMABUKURO JUNIOR, 2006).

## 2.3 PADRÕES DE SOFTWARE

Michaelis (2013) afirma que padrão é um modelo oficial de pesos e medidas. Também pode ser definido como metrologia, grandeza, tipo que serve para definir uma unidade. Tipo, modelo (AURÉLIO, 2008-2013).

Um padrão é definido como a descrição de um problema recursivo que ocorre em determinado ambiente, onde existe uma solução a qual seja possível utilizar várias vezes, sem que tal solução se repita por mais de uma vez. Ou ainda, reutilização de soluções advindas do conhecimento dos desenvolvedores, que podem ser aplicadas em distintas situações similares e que podem ser documentadas na forma de padrões (ALEXANDER, 1979).

Para Harrison et al. (1999), padrões identificam e especificam abstrações de nível mais alto que classes, instâncias ou componentes. É de suma importância que o desenvolvedor entenda o problema e o contexto que envolve esse problema, permitindo que uma solução mais adequada seja encontrada.

“Como um elemento no mundo, cada padrão é uma relação entre um certo contexto, um certo sistema de forças que ocorrem repetidamente naquele contexto, e uma certa configuração espacial que permite que estas forças se solucionem por si só” (ALEXANDER et. al., 1977).

Padrões de software devem ser documentados em um formato adequado, na literatura existem algumas propostas para documentar padrões como a de Alexander (1979). Este afirma que cada padrão deve ser apresentado em forma de uma regra, estabelecendo relação entre um contexto, um sistema de forças existente nesse contexto e uma configuração, que permite que as forças sejam equilibradas no contexto referido. Esse formato também é conhecido como forma alexandrina ou canônica. GOF (1995) define padrões com base em um gabarito, o qual é formado pelos seguintes componentes: nome e classificação do padrão, intenção, nome alternativo, motivação, aplicabilidade, estrutura, participantes, colaborações, consequências, implementação, exemplos de código, usos conhecidos e padrões relacionados. Esse formato é conhecido como formato GOF.

Para Souza & Andrade (2009) a utilização de padrões de software traz benefícios, os quais são citados a seguir:

- Permite um vocabulário comum, sucinto e principalmente compartilhado entre os engenheiros de software o que melhora a comunicação desses;
- Redução de tempo de desenvolvimento, já que os esforços se concentram em aspectos particulares e novos do sistema;
- Melhora a compreensão no que diz respeito à evolução do código e ao sistema propriamente dito, evitando também o retrabalho durante o processo de construção do software;
- Ocorre melhor distribuição de responsabilidades entre os envolvidos no projeto bem como a diminuição de "efeitos colaterais" causados por solicitações de mudanças;
- Auxilia na reestruturação do sistema, caso essa seja necessária.

Várias áreas da Engenharia de Software fazem uso de padrões com objetivo de documentar soluções. Tais padrões podem ser dispostos em categorias, são elas: padrões de projeto e análise, padrões arquiteturais, de código e usabilidade (SHIMABUKURO JUNIOR, 2006).

## 2.4 FRAMEWORKS

Um *framework* pode ser definido como um projeto genérico em um domínio, em que esse pode ser adaptado a aplicações específicas, no qual será utilizado como modelo para construção de aplicações como um todo (JOHNSON, 1990).

Souza (1998) afirma que *framework* "é um esqueleto de implementação de uma aplicação ou de um subsistema de aplicação em um domínio de problema particular".

*Frameworks* também são definidos por Wirfs & Johnson (1990) como "um projeto reutilizável de um programa ou uma parte de um programa, expresso como um conjunto de classes".

Segundo Silva (2000) *frameworks* são estruturas de classes com implementações incompletas e, quando essas são estendidas, produzem diferentes artefatos de software.

No início da década de 80 os *frameworks* de aplicação começaram a ser construídos acrescentando às bibliotecas de classes os relacionamentos e interação entre as classes que o compõem. Por meio dos *frameworks* é possível não só reutilizar linhas de código bem como o projeto abstrato que envolve o domínio da aplicação, isto é, além de utilizar componentes isolados também reutilizam a arquitetura como um todo de um domínio específico (BRAGA et al., 2001).

Para fornecer uma solução parcial a uma família de problemas, não é necessário que um *framework* esteja implementado em uma linguagem de programação, caso o objetivo dele seja a reutilização. A principal vantagem de uma implementação parcial é que a solução final está mais próxima de ser concluída pelo desenvolvedor da nova aplicação.

Segundo Roberts & Jonhson (2016) *frameworks* são classificados em dois grupos: frameworks de aplicação orientado à objetos e frameworks de componentes. O primeiro geram aplicações orientadas a objetos e são classificados quanto a seu escopo em: *frameworks* de infraestrutura de sistemas (ou *framework* horizontal), *frameworks* de integração de *middleware* e *frameworks* de aplicações corporativas (ou *framework* vertical). *Frameworks* de componentes é uma entidade de software que provê suporte a componentes seguindo determinado modelo. São exemplos de *frameworks* de componentes: *Tapestry e Java Server Faces.*

A principal vantagem de utilizar *frameworks* é devido ao reuso de código e projeto, com isso tempo e esforço são reduzidos no desenvolvimento do software em si. Por outro lado, desenvolver *frameworks* é um tanto complexo assim como a aprendizagem de sua utilização.

## 2.5 COMPONENTES E BIBLIOTECAS DE CLASSES

Componente é “uma unidade de composição com interfaces contratualmente especificadas e dependências de contexto explícitas. Esses podem ser duplicados ou estar sujeitos à alteração de terceiros.” Dessa maneira, desde que possua uma interface definida,

qualquer artefato de software pode ser considerado um componente (SZYPERSKI, 1996 - 1997).

Szyperski (2002) afirma que o DBC (Desenvolvimento Baseado em Componentes) é uma abordagem que possibilita a diminuição de custos, aumento da qualidade do software e redução do tempo de implementação.

Spagnoli e Becker (2003) consideram duas perspectivas no desenvolvimento de software a partir do DBC: desenvolvimento de componentes e desenvolvimento com componentes. A primeira engloba as atividades envolvidas na concepção e implementação do componente, sendo importante gerar a documentação necessária para posterior reutilização do mesmo. A segunda perspectiva analisa a existência de componentes e os inclui às atividades necessárias para o desenvolvimento de software por sua composição.

Nesse contexto, uma aplicação é construída partindo de um conjunto de componentes interligados (módulos) e à medida que são delegadas responsabilidade aos componentes, e esses sendo reutilizados, o esforço é reduzido aumentando a produtividade no desenvolvimento. A principal característica do desenvolvimento baseado em componentes é a separação entre especificação de um componente de software e sua implementação (SILVA, 2000).

O desenvolvimento de software baseado em componentes, bem como o desenvolvimento dos componentes de código é viável por meio do uso de tecnologias de componentes, o EJB (*Enterprise Java Beans*), por exemplo (SPAGNOLI e BECKER, 2003). Essa arquitetura permite que as regras de negócio sejam implementadas em componentes específicos denominados *Session Beans*[4]. O EJB é executado dentro de um EJB *Container*[5], sendo este o responsável pela implementação dos recursos oferecidos às aplicações (VOTRI, 2012).

O EJB é definido como "uma arquitetura de componentes do lado servidor para a plataforma Java EE. A tecnologia EJB permite o desenvolvimento simplificado e rápido de aplicativos distribuídos, transacionais, seguros e portáteis baseados na tecnologia Java (ORACLE, 2016). Com relação à segurança, o EJB tem suporte a autenticação e autorização. Dessa maneira os desenvolvedores das aplicações não precisam implementar essa lógica, pelo fato desta fazer parte da arquitetura do mesmo, bem como o acesso remoto por meio de protocolos de comunicação o que permite que aplicações EJB possam ser expostas como *Web*

[4] Nome atribuído aos componentes EJB.
[5] Local onde os *beans* são disponibilizados para serem utilizados e tem como função gerenciar todos os aspectos do *bean* durante a execução deste.

*Service* . Outra vantagem dessa arquitetura é o acesso múltiplo a sistemas corporativos, ou seja, vários usuários com aceso simultâneo, mas de maneira controlada evitando problemas de concorrência (CAELUM, 2011).

Figueiredo (2012) afirma que CBSE (Engenharia de Software Baseada em Componentes) “é um processo de definição, implementação e composição de componentes independentes” também definido como um modelo de processo orientado ao reuso, é baseado na existência de componentes reusáveis e o processo se concentra em integrar os componentes.

O reuso, além de utilizar código, passou a abranger estrutura de dados e isso devido à concretização do paradigma da orientação a objetos. Nesse contexto, bibliotecas de classes foram criadas a fim de resolver problemas em várias áreas, por exemplo: sistemas operacionais, computação gráfica, interfaces, banco de dados, etc.

A criação de módulos unida ao ocultamento de informações, onde esses são oferecidos pelas classes, facilitam o reuso (BRAGA et al., 2001).

De acordo com Ferguson et al. (2002) uma biblioteca de classes é uma forma eficiente de reutilizar e distribuir código. Também podem ser definidas como um conjunto de classes com um propósito geral (VOLPON, 2011).

## 2.6 ENGENHARIA DE REQUISITOS

É o processo de descobrir, analisar, documentar e verificar os serviços e restrições existentes em um sistema de software (SOMERVILLE, 2011).

De acordo com Junior (2010) a engenharia de requisitos é a subárea da Engenharia de Software responsável por definir formalmente uma metodologia que se relaciona com a elicitação, organização, classificação, gerenciamento, controle e documentação de requisitos, cujo objetivo é obter o controle sobre os mesmos.

No contexto de ciclo de vida do software é a maneira como se definem as atividades desenvolvidas as quais estão relacionadas com a definição de requisitos de um sistema (KOTONYA; SOMERVILLE, 1998).

> [...] o ramo da Engenharia de Software relacionado aos objetivos do mundo real estabelecidos para as funções e restrições aplicáveis a esses sistemas de software.

> Está também relacionada com a ligação entre esses fatores e a precisa especificação do comportamento do software e com sua evolução no tempo e através de família de produtos (ZAVE, 1997).

A engenharia de requisitos é definida como o resultado da especificação de características operacionais do software. Ela estabelece as restrições que esse deve atender e permite que sejam elaboradas as necessidades básicas estabelecidas na fase inicial do projeto. Dessa maneira afirma-se que o objetivo da Engenharia de Requisitos é fornecer uma descrição escrita do problema, facilitando o entendimento do mesmo para todas as partes envolvidas no desenvolvimento do software (PRESSMAN, 2011).

Portanto, a engenharia de requisitos tem como objetivo principal o controle sobre os requisitos por meio de um documento definido como especificação de requisitos.

E também pode ser dividida em desenvolvimento e gerenciamento de requisitos. O desenvolvimento de requisitos envolve quatro atividades: elicitação, análise, especificação e validação. O gerenciamento de requisitos, por sua vez, é composto por três atividades: identificar/organizar os requisitos, gerenciamento de mudanças e rastreabilidade dos requisitos (SOMERVILLE, 2011).

O resultado desse processo é o documento de especificação de requisitos onde estão dispostas as informações sobre o software, ou seja, suas funcionalidades, restrições, validações e dados referentes aos requisitos a serem implementados pelos desenvolvedores. É de suma importância que esse documento seja atualizado, pois é útil como um contrato para o desenvolvimento do software (SOMERVILLE, 2011).

### 2.6.1 REQUISITOS

Requisitos de um sistema são definidos como os serviços que esse oferece as descrições das funções que ele deve realizar, e as restrições existentes para seu funcionamento (SOMERVILLE, 2011).

Requisito é uma condição de um software a qual deve ser implementada por um sistema ou componentes de sistema para se atingir um objetivo determinado. Também definem as necessidades e expectativas dos envolvidos no projeto, independente se por parte da equipe, responsável por desenvolver o sistema, ou do próprio cliente (JÚNIOR, 2010).

Por meio dessa definição Somerville (2011) diferencia o termo requisitos de duas maneiras: requisitos de usuário e de sistema. O primeiro pode ser definido como uma declaração acerca dos serviços que o sistema fornecerá a seus usuários, essa pode ser feita em linguagem natural, bem como as restrições que este irá operar. O segundo, requisitos de sistema envolvem as funcionalidades, serviços e restrições de forma detalhada para que fique claro o que deve ser implementado.

No contexto de requisitos de software Somerville (2011), afirma que esses são geralmente classificados em:

Requisitos Funcionais: referem-se às funcionalidades, isto é, **o que** o sistema deve fazer. Estes definem os recursos específicos fornecidos pelo sistema.

Requisitos Não funcionais: Desempenho, proteção ou disponibilidade são requisitos que normalmente especificam ou restringem as características do sistema como um todo. No entanto, deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema.

Segundo Júnior (2010) requisitos funcionais definem ações ou funcionalidades fornecidas pelo sistema e que em geral podem ser visualizados pelos casos de uso do mesmo. Já os requisitos não funcionais descrevem atributos do sistema ou do ambiente do sistema, os quais podem ser extensibilidade, usabilidade, confiabilidade, desempenho, escalabilidade, reusabilidade, capacidade de manutenção, reutilização de código, performance, eficiência no desenvolvimento, confiabilidade nos dados apresentados.

Requisitos são classificados de duas maneiras: requisitos funcionais e não funcionais. Requisitos funcionais são descritos como uma lista de funcionalidades, ou seja, tudo **o que** o sistema deve fazer. Por sua vez, os requisitos não funcionais consistem em restrições que determinam **como** o sistema deverá realizar seus requisitos funcionais (WAZLAWICK, 2004).

No que diz respeito aos RFs são classificados em evidentes ou ocultos, ou seja, RF evidentes são efetuados com o conhecimento do usuário, já RF ocultos são efetuados pelo sistema sem o conhecimento do usuário. Por sua vez, os RNFs são classificados em permanentes (contrário de transitório), desejáveis (contrário de obrigatório), de categoria, associados à RF e suplementares. Dessa maneira, RNF de categoria são classificados em: interface, implementação, eficiência, tolerância a falhas, segurança, compatibilidade, etc. Por fim, os RFs suplementares são associados a qualquer RF sem distinção (SAMPAIO, 2007).

## 2.7 REUSO DE REQUISITOS

A tecnologia mais importante no cenário mundial continua sendo o software de computador, pois ele distribui o produto mais importante da nossa era – a informação. Por esse fato ele tem se incorporado à vida do ser humano e este por sua vez aderido aos recursos oferecidos pelo software.

Como prever que o software seria indispensável para negócios, ciência e engenharia ou que por meio dele seria viável a criação de novas tecnologias como engenharia genética e nanotecnologia, por exemplo; que produtos de software seriam comprados por consumidores em lojas de bairro ou que este evoluiria de forma lenta de produto para serviço. Há algum tempo não se podia prever que o software seria incorporado em sistemas de todas as áreas: transportes, medicina, telecomunicações, entretenimento, militar, industrial, entre outros (PRESSMAN, 2011).

“O mundo moderno não poderia existir sem o software” (SOMERVILLE, 2011), um exemplo disso é a manufatura e a distribuição industriais que atualmente são informatizadas bem como o sistema financeiro (SOMERVILLE, 2011).

Requisitos ou características do produto descrevem os recursos que o sistema deverá fornecer para atender as necessidades dos interessados, também são vistos como um serviço oferecido pelo sistema para atender uma ou mais necessidades e servem para auxiliar na definição da solução a ser implementada (JÚNIOR, 2010).

Com base nas definições citadas nota-se a importância do software e consequentemente a importância dos requisitos de software os quais são determinados para uma finalidade comum (exemplo: controlar um dispositivo ou encontrar informações). De maneira geral os requisitos atendem as necessidades do cliente, sendo assim são o ponto de partida para o desenvolvimento do software em si (SOMERVILLE, 2011).

Para um produto de software ser considerado adequado é necessário que ele esteja dentro dos padrões de qualidade existentes, sua entrega deve satisfazer as necessidades dos usuários por meio dos requisitos de software e deve também estar dentro do prazo estipulado (VILLEGAS e LAGUNA, 2001).

Nesse sentido “o reuso de software tem sido promovido para aumentar o retorno sobre investimentos”. As exigências por menores custos de produção e manutenção de software,

maior qualidade e entregas mais rápidas trouxeram como resposta o desenvolvimento baseado em reuso (SOMERVILLE, 2011).

"Reuso de software é o processo de criação de sistemas de software a partir de um software existente ao invés de construir sistemas de software a partir do zero. Na tentativa de entender o porquê, os pesquisadores têm renovado o interesse na reutilização de software e os obstáculos à sua implementação" (Krueger, 1992). Dessa maneira, "o reuso é visto pela comunidade de software como uma atividade fundamental em todos os processos relacionados com o desenvolvimento de software" (FRANCH et al. APUD CEZÁRIO, 2011) ampliando sua utilização, não se limitando ao reuso de código. Deste modo, o reuso de requisitos tende a auxiliar os engenheiros nas fases iniciais do processo de software (elicitação, análise, validação e documentação de requisitos), obtendo especificações de requisitos com maior qualidade (CEZÁRIO, 2011). Fator de destaque é que por meio do reuso de especificações de requisitos outros artefatos também se tornam reutilizáveis como modelos, código, documentação e planos de testes (CEZÁRIO, 2011).

Neighbors (1994) afirma que no tocante ao desenvolvimento de software, é de suma importância que sejam criados e/ou identificados recursos reutilizáveis nas fases iniciais, pois nas fases posteriores o impacto da reutilização será maior.

A pesquisa sobre reutilização/reuso tem como objetivo "reunir, organizar e facilitar o acesso a todo conhecimento sobre o desenvolvimento de software em um determinado domínio" (ZIRBES, 1995).

### 2.7.1 APLICAÇÕES DE REUSO DE REQUISITOS

No processo da engenharia de requisitos a reutilização é útil, principalmente no reuso de requisitos de sistemas similares e de modelos. No entanto, o produto das fases iniciais é relacionado com o domínio da aplicação, fazendo com que a reutilização de especificações, modelos de requisitos ou estratégias de projeto só possa ser feita com sucesso entre aplicações da mesma família, ou seja, aplicações que compartilhem requisitos e restrições (ZIRBES, 1995).

Trabalhar com famílias de produtos é uma ideia utilizada atualmente, que foi alcançada por Parnas em 1976. Ele afirmou que criar famílias de produtos é mais eficiente que trabalhar com sistemas de softwares individuais (ROMANOSKY et al., 2008).

Para que requisitos de sistemas anteriores possam ser reutilizados em novos sistemas faz-se necessário o estudo do domínio da aplicação e do problema, com o objetivo de identificar partes comuns e variáveis em ambos, sendo que estes devem fazer parte da mesma área de aplicação. É importante o desenvolvimento de uma abordagem para reutilização de artefatos na fase de especificação de requisitos, envolvendo dessa maneira as fases iniciais do processo de desenvolvimento de software (SILVEIRA e VIDAL, 2002).

Pimenta (1998) afirma que domínio na engenharia de software pode ser definido como uma família de sistemas os quais possuem características semelhantes e consequentemente um conjunto significativo de requisitos.

Na engenharia de requisitos as abordagens de reutilização utilizam diferentes perspectivas, são elas (SILVEIRA, 2006):

- A analogia[6] entre sistemas de software;
- A teoria de domínio;
- Combinar a reutilização de requisitos com a gestão da interação para diferentes níveis de abstrações;
- Raciocínio baseado em casos;
- Reutilização de componentes baseado em casos de uso;
- Discriminantes dentro de uma família de aplicações;
- Objetos do domínio;
- Classificação do domínio para facilitar o refinamento e a reutilização de requisitos: RARE com a ferramenta IDIOM;
- Reutilização por reciclagem de requisitos.

Somerville e Kotonya (1998) afirmam que a reutilização de requisitos é oferecida como uma técnica de levantamento de requisitos, considerando que ela ocorre em um valor superior a 50%, em muitos sistemas.

Segundo Somerville e Sawyer (1997) na prática, quando se tratam de sistemas semelhantes, em torno de 80% dos requisitos podem ser reutilizados. Em sistemas que

[6] A analogia utiliza a experiência para resolver novos problemas. Problemas similares têm soluções similares. O processo de raciocínio por analogia identifica determinados aspectos em novos e antigos problemas e procura utilizar as soluções encontradas para inferir uma nova solução.

possuem o mesmo domínio de aplicação, ERP[7] por exemplo, esse valor pode variar de 80% a 100% na reutilização de alguns processos ou componentes de dados (DANEVA, 2004).

A reutilização de requisitos pode ser consolidada nas seguintes situações (SOMERVILLE e KOTONYA, 1998):

Quando as informações fornecidas pelos requisitos são referentes ao domínio da aplicação, tais informações derivadas do domínio podem ser aplicadas a novos sistemas;

Quando os requisitos apresentam um modo consistente no estilo de apresentação da informação. Um exemplo disso são os requisitos que especificam características de interface do utilizador;

Quando os requisitos refletem políticas da empresa, tanto políticas de organização como de segurança podem estar inclusas nos requisitos do sistema e reutilizadas em outros sistemas posteriormente.

### 2.7.2 VANTAGENS DO REUSO DE REQUISITOS

Para Edelweiss (1994), o uso de abordagens de reutilização durante a fase de especificação de requisitos apresenta as seguintes vantagens:

- Redução do custo do desenvolvimento da especificação;
- Redução do custo de verificação e validação da especificação;
- Aumento da produtividade no desenvolvimento de especificações;
- Aumento da qualidade das especificações;
- Padronização de especificações;
- Facilidade de comunicação entre equipes que utilizam a mesma biblioteca.

Para Silveira (2006) a reutilização possui benefícios ao nível de tempo de desenvolvimento, redução de custos e aumento da qualidade e produtividade.

A reutilização de especificações melhora a produtividade no desenvolvimento de software oferecendo aos desenvolvedores um início mais rápido durante a análise de requisitos. Também pode beneficiar a engenharia de requisitos com a melhora da qualidade

[7] *Enterprise Resource Planning*: exemplo de sistema/produto de software genérico.

nas especificações resultantes. Finalmente, ela pode enriquecer a base de conhecimento própria do engenheiro de software, fornecendo a experiência necessária para entender e resolver problemas análogos (PIMENTA, 1998).

### 2.7.3 ABORDAGENS DO REUSO DE REQUISITOS

Os tópicos a seguir tratam de maneiras de reuso de requisitos citados na literatura.

#### 2.7.3.1 ANALOGIA ENTRE SISTEMAS DE SOFTWARE

A analogia tem como objetivo absorver conhecimento de um domínio e aplicá-lo a outro. Também definida como um paradigma usado em especificações de sistemas análogos, ou seja, resolve problemas transferindo conhecimentos anteriormente resolvidos para novos problemas, sendo que estes compartilham aspectos semelhantes ao da experiência anterior, permitindo que essa transferência possa construir novas soluções (MAIDEN e SUTCLIFFE, 1992).

Para Carbunell apud Pibber (1996) a analogia é um meio para explorar o reuso de especificações de requisitos em domínios diferentes. Em Mainden (1992) são apresentados estudos desenvolvidos a partir domínios diferentes demonstrando que por meio da analogia é possível explorar especificações reusáveis.

> Resolver um problema por analogia é um processo de transferir conhecimento de problemas passados para novos problemas que compartilham similaridades significativas. A transferência de conhecimento é então usada para construir soluções para novos problemas (CARBONELL, 1985).

Engenharia de Software, gerenciamento de sistemas, paradigmas de aprendizado de máquinas, raciocínio baseado em casos são áreas da Ciência da Computação que utilizam o conceito de analogia (MARTINS, 2006).

O processo de reutilização usando analogia pode ser dividido em duas partes: a primeira tem o objetivo de identificar um conjunto de domínios abstratos que representem famílias de sistemas cujas características são semelhantes. A segunda parte, por sua vez, utiliza o raciocínio por analogia, isto é, deve-se reconhecer a semelhança entre o problema e os domínios abstratos, compreender tal semelhança e finalmente transferir conhecimento.

O raciocínio por analogia, a segunda etapa, é composto de três passos principais: recuperação dos componentes, seleção do componente mais adequado e adaptação de componentes. Na prática, isto implica na existência de uma biblioteca, por meio da qual seja possível a identificação de similaridades e a consequente seleção dos recursos reutilizáveis adequados (ZIRBES, 1995).

As Figuras 2 e 3 apresentam uma analogia entre um sistema de reserva de restaurante e um sistema de matrícula universitária. O sistema de reserva do restaurante permite para os seus clientes reservar com antecedência um lugar, podem ser reservados um ou mais lugares. O pessoal do restaurante utiliza o sistema para responder aos pedidos e administrar às reservas. Uma lista de espera é criada sempre que todos os lugares são ocupados. Os clientes são transferidos da lista de espera para os lugares vagos. Um diagrama de contexto deste sistema de reserva é mostrado na Figura 2 (PIBBER, 1996).

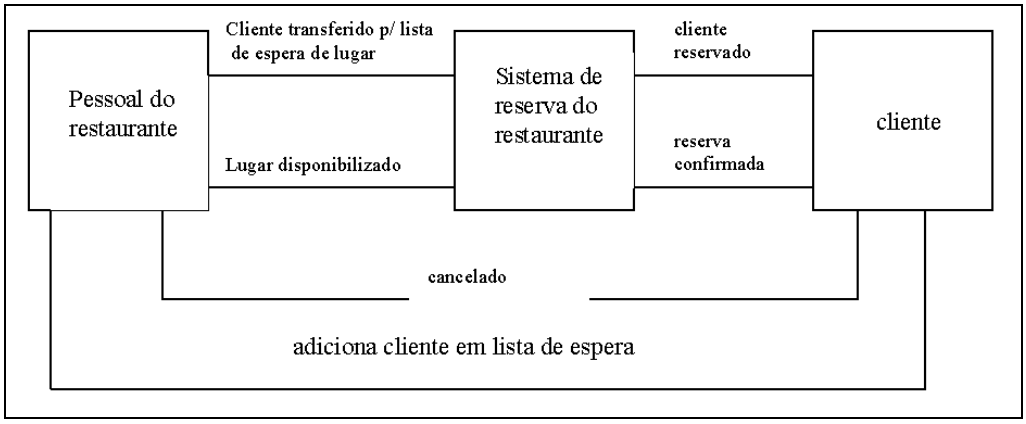


**Figura 2 - Diagrama de Contexto: Para o Sistema de Reserva de Restaurante Adaptado de Pibber (1996)**

Ambos os diagramas demonstram potencial para reuso, pelo fato de apresentar uma estrutura de conhecimento similar que pode ser explorado por meio da analogia.

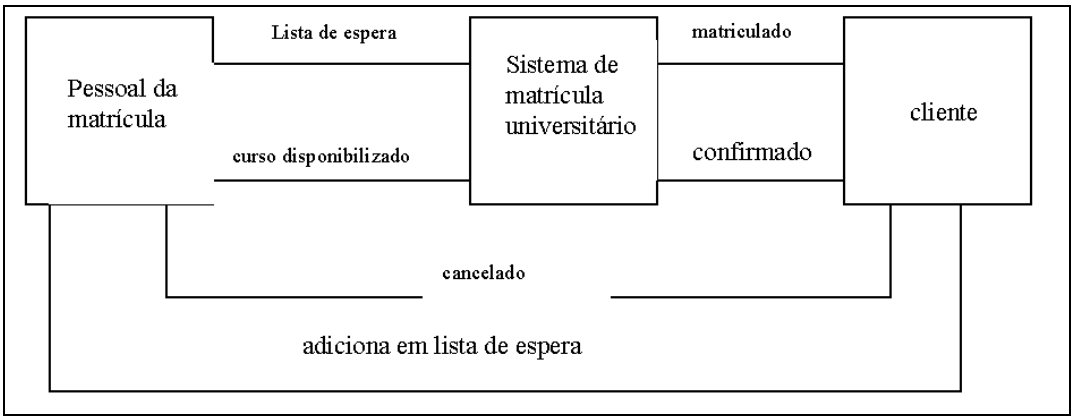


**Figura 3 - Diagrama de Contexto: Para o Sistema de Matrícula Universitária Adaptado de Pibber (1996)**

### 2.7.3.2 PADRÕES DE REQUISITOS PARA A ESCRITA DE REQUISITOS DE SOFTWARE

Padrões de requisitos propõem soluções reutilizáveis e genéricas para a escrita de requisitos, ou para problemas recursivos na engenharia de requisitos. O objetivo é referenciar padrões de requisitos a padrões cujo objetivo é a criação de especificação de requisitos. Desse modo, padrões de requisitos auxiliam na maneira de como escrever os requisitos, sugerem informações, ajudam com fatores e sugestões a serem consideradas. Além disso, aumentam a qualidade e reduzem tempo de desenvolvimento do software, pelo fato de não existir a necessidade de escrever o requisito do início, pois provê um ponto de partida, uma estrutura para sua construção (CEZÁRIO, 2011). Nesse contexto, de padrões para a escrita de requisitos de software existem dois tipos de padrões, os quais serão descritos a seguir.

#### A) O PADRÃO REQUISITOS ENCAPSULADOS

De acordo com Somerville (2003), a duplicação de informações é um risco ao custo de um sistema. A manutenção de um requisito torna-se cara quando o mesmo se encontra em diversos locais de um documento. Para modificar um requisito duplicado, por exemplo, um engenheiro de software precisa encontrar todas as instâncias desse requisito no documento como um todo para que o mesmo seja consistente. Portanto, o objetivo principal desse padrão

é eliminar duplicações que podem ter ocorrência em requisitos de software (RAMOS et al., 2006).

A duplicação de informações ocorre de duas maneiras:

- O requisito duplicado em diferentes estruturas de um documento de requisitos;
- O requisito duplicado na mesma estrutura de um documento de requisitos.

Há casos em que somente parte do requisito estará duplicada, dessa forma é necessário reescrevê-lo, retirando a parte duplicada para sua melhor compreensão. Com a utilização de uma estrutura única para encapsular as informações duplicadas, percebe-se uma diminuição no tamanho no documento de requisitos e maior facilidade para localizá-lo dentro do próprio documento (RAMOS et al., 2006).

Para Jacobson (2005), no que diz respeito à duplicação de informações no nível de requisitos e dentro do contexto de casos de uso, em algumas situações a duplicação é uma forma necessária de reuso de requisitos.

B) O PADRÃO REQUISITOS DA QUALIDADE

Conforme citado na seção 2.6.1, no que diz respeito a requisitos, segundo Somerville (2011) esses podem ser divididos em dois grupos: RF e RNF. Nesse momento, destacam-se os RNFs os quais mapeiam aspectos de qualidade de um software. Também conhecidos como atributos de qualidade, restrições, objetivos, etc. Dentre as características dos RNF se destacam: a dificuldade de serem detectados, não são mapeados diretamente nas funcionalidades, devem ser observados ao longo do desenvolvimento do software, se relacionam diretamente com o produto, suas funções e/ou com o ambiente onde será implantado (COUTINHO et al., 2008).

Na maioria das vezes RNF são conflitantes, isto é, para que um RNF A seja atendido o RNF B não deve ser atendido. A Figura 4 demonstra uma forma de conflito entre RNF, ou seja, para atender os requisitos de acessibilidade, tempo de resposta e usuários simultâneos, provavelmente será ultrapassado do valor do orçamento do projeto. Para resolver tal problema é necessário a não implementação de um dos RNF envolvidos no conflito ou a redução da qualidade de algum dos RNF para balancear os custos do projeto (COUTINHO et al., 2008).

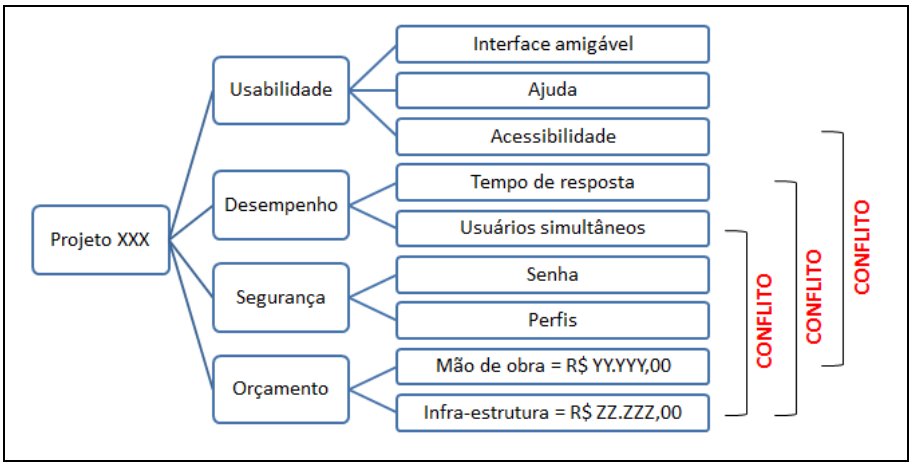


**Figura 4 - Demonstrativo de Conflitos entre RNFs**
**Fonte: Adaptado de Coutinho (2008)**

De acordo com Coutinho et al.(2008) na seleção dos RNF do projeto é importante considerar os requisitos conflitantes, de maneira a serem identificados e priorizados. Para tanto, segue-se o fluxo abaixo:

- Identificar os RNF conflitantes;
- Detalhar os conflitos dos mesmos;
- Comunicar a necessidade de negociação com o cliente;
- Informar os RNF conflitantes, o impacto de sua construção ou não, incluindo informações qualitativas e quantitativas;
- Negociar com o cliente alterações de escopo e alterações relacionadas ao gerenciamento do projeto (custos, prazo e qualidade);
- Realizar alterações necessárias sejam elas nos RNF e/ou no projeto.

Na Figura 5 é representada uma descrição da solução do padrão, desde a identificação e seleção dos RNF, a avaliação dos requisitos conflitantes e negociação com o cliente.

Nesse contexto, após a identificação e seleção dos RNF a serem utilizados é realizada a negociação com o cliente em paralelo com a identificação de conflitos existentes nesses RNF, essa detecção de conflitos é obtida por meio de uma matriz de rastreabilidade. Definidos os RNF, esses são avaliados e refinados junto com a arquitetura da aplicação.

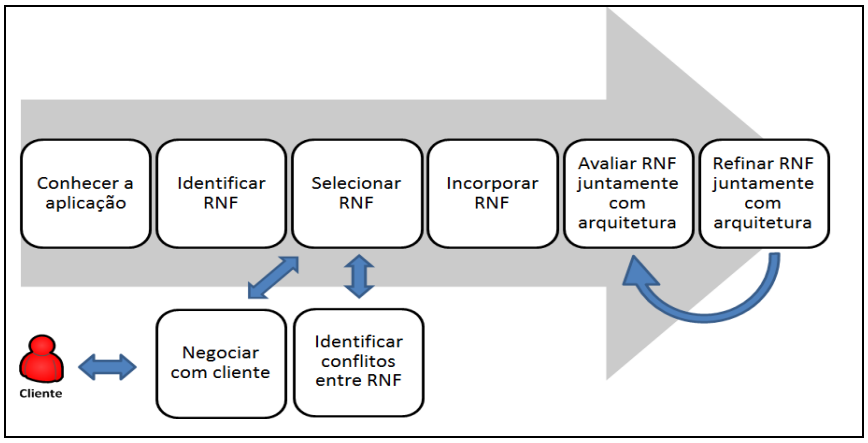


**Figura 5 - Descrição da Solução do Padrão**
**Fonte: Adaptado de Coutinho (2008)**

COUTINHO et al. (2008) afirma que a utilização do padrão citado traz consequências como maior aderência aos requisitos do cliente, adequação dos elementos de arquitetura prevenindo retrabalho, melhor análise acerca dos impactos do projeto culminando na satisfação do cliente.

# 3 MATERIAL E MÉTODOS

A proposta deste trabalho é aplicar as abordagens de reuso de requisitos apresentadas e empregá-las como estudo experimental nos sistemas: Birô de Digitalização e Digitaldoc, produtos da Anix Sistemas LTDA[8]. O sistema de software E-Atos é apresentado somente em nível de conhecimento, não sendo utilizado nesse estudo.

## 3.1 PRODUTOS DE SOFTWARE: BIRÔ DE DIGITALIZAÇÃO, DIGITALDOC E E-ATOS

Nas seções a seguir são descritos os produtos desenvolvidos pela empresa Anix Sistemas LTDA, incluindo seus requisitos funcionais e diagramas de classe.

Para o desenvolvimento e manutenção desses produtos a empresa utiliza tecnologias como Java, JNPL (*Java Network Lauching Protocol*), GWT (*Google Web Toolkit*), JWS (*Java Web Start*). A análise dos requisitos foi realizada baseada na UML (LEONHART, 2011).

### 3.1.1 BIRÔ DE DIGITALIZAÇÃO

A ferramenta foi criada para atender a demanda do setor de digitalização, ou seja, somente prestar serviços de digitalização, pois essa era a necessidade principal do público a ser atendido. O produto funciona apenas em rede local sendo vendido com uma licença única (DIGITALDOC, 2015).

Para o desenvolvimento do Birô de Digitalização foi realizado o levantamento dos requisitos/funcionalidades.

[8] Nome fantasia: Digitaldoc

A Tabela 1 apresenta os requisitos funcionais do sistema.

**Tabela 1 - Requisitos Funcionais: Birô de Digitalização**

| CÓDIGO | REQUISITOS FUNCIONAIS |
|---|---|
| F01 | CADASTRAR USUÁRIO |
| F02 | FAZER LOGIN |
| F03 | CADASTRAR PASTA |
| F04 | CADASTRAR TIPO DOCUMENTO |
| F05 | DIGITALIZAR |
| F06 | CADASTRAR DOCUMENTO |
| F07 | PESQUISAR DOCUMENTO |
| F08 | GERAR MIDIA PESQUISÁVEL |

**Fonte: Digitaldoc (2015)**

A Figura 6 apresenta o diagrama de classes do sistema Birô de Digitalização. Conforme a Figura 6, a classe *FormCadPadrao* é a base para todas as demais classes ,isto é, todos os outros *forms* são montados sob ele. A classe *TUsuario* é a parte servidor. Outra classe importante é *FormAssinar* ela é responsável por anexar o certificado digital, esse por sua vez é indispensável para a validade do documento. Por fim a classe *FormDigitaliza* "contém" o scanner, desse modo o documento é digitalizado.

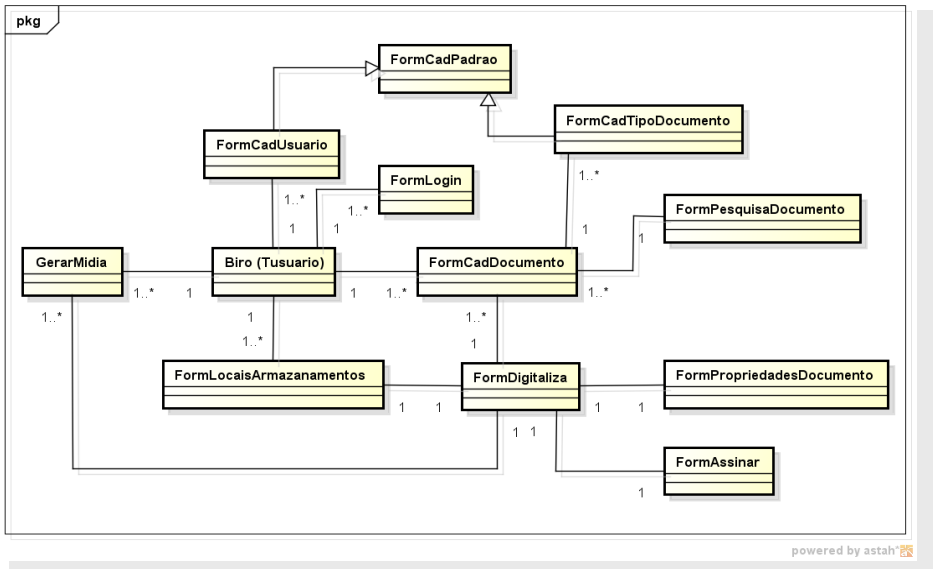


**Figura 6 - Diagrama de Classes: Birô de Digitalização**

### 3.1.2 DIGITALDOC

O produto foi desenvolvido para atender setores como digitalização, gestão de documentos e gestão de documentos da ISO. O principal objetivo da ferramenta era suprir as necessidades de empresas e organizações públicas viabilizando ferramentas que aumentassem a produtividade por meio da organização de seus documentos, ou seja, controlar o ciclo de vida de um documento. Dessa maneira, atenderia a necessidade do mercado sendo ofertado pela empresa como licença mensal – produto servidor/*web* (DIGITALDOC, 2015).

É um sistema na *web* cujo acesso é possível tanto com uso da internet quanto sem uso da mesma, sendo que para esse último é necessário fazer uso de um instalador próprio. Os documentos são digitalizados e de acordo com a função (cargo) dos usuários são delegadas permissões a esses para acessar determinado documento, isto é, elaborar, revisar, aprovar e /ou distribuir tal documento (DIGITALDOC, 2015).

O sistema Digitaldoc possui uma lista de requisitos/funcionalidades essenciais para sua implementação.

Na Tabela 2 são listados alguns dos requisitos funcionais do software.

**Tabela 2 - Requisitos Funcionais: Digitaldoc**

| CÓDIGO | REQUISITOS FUNCIONAIS |
|---|---|
| F01 | CADASTRAR USUÁRIO |
| F02 | FAZER LOGIN |
| F03 | CADASTRAR PASTA |
| F04 | CADASTRAR TIPO DOCUMENTO |
| F05 | DIGITALIZAR |
| F06 | CADASTRAR DOCUMENTO |
| F07 | PESQUISAR DOCUMENTO |
| F08 | LISTAR DOCUMENTO NA PASTA |

**Fonte: Digitaldoc (2015)**

Na Figura 7 o diagrama de classes apresenta a estrutura do sistema Digitaldoc.

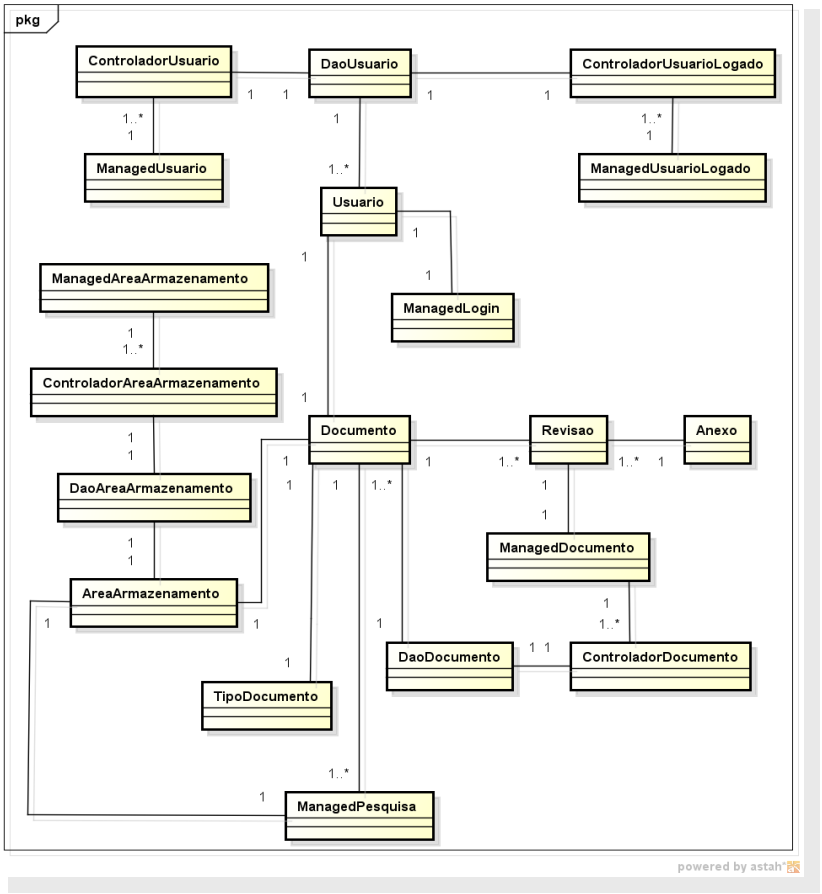


**Figura 7 - Diagrama de Classes: Digitaldoc**

O diagrama apresentado na Figura 7 é resumido da seguinte maneira: as classes *Managed's* e Controladores possuem as regras de negócio, isto é, como deve ser gerido o sistema. As classes *Dao* contém as informações de conexão com banco de dados, dessa maneira os dados ficam acessíveis. O restante das classes é denominado *beens*, cujas informações serão gravadas/persistidas no banco de dados.

### 3.1.3 E-ATOS

> Amplamente propagado por todos os meios de comunicação, a redução no emprego do papel proporciona celeridade no andamento dos processos e alinha-se estrategicamente às metas de sustentabilidade ambiental, gerando, além dos ganhos contábeis, forte apelo pedagógico (LEONHART, 2011).

O nome comercial da ferramenta E-Atos é proveniente da junção das palavras atos e eletrônico. Devido ao fato do mesmo realizar o controle de publicações para órgãos públicos - DOE[9], ou seja, permite armazenar e visualizar arquivos PDF, mediante acesso a somente um servidor *web*. A ideia consiste em cadastrar órgãos públicos e disponibilizar um acesso para o gestor municipal permitindo que esse faça o envio de publicações, dessa maneira e qualquer cidadão pode consultar e visualizar as publicações referentes à determinada prefeitura (LEONHART, 2011).

A seguir, na Tabela 3 são listados os requisitos funcionais do software E-Atos, no total são sete requisitos.

**Tabela 3 - Requisitos Funcionais: E-Atos**

| CÓDIGO | REQUISITOS FUNCIONAIS |
|---|---|
| F01 | MANTER PREFEITURA |
| F02 | CADASTRAR USUÁRIO DA PREFEITURA |
| F03 | MANTER CADASTRO DE ATOS OFICIAIS |
| F04 | PESQUISAR PUBLICAÇÕES |
| F05 | VISUALIZAR PUBLICAÇÕES |
| F06 | LOGIN NO SISTEMA |
| F07 | RECUPERAR SENHA |

**Fonte: Digitaldoc(2016)**

Na Figura 8 é apresentado o diagrama de classes referente ao sistema E-Atos.

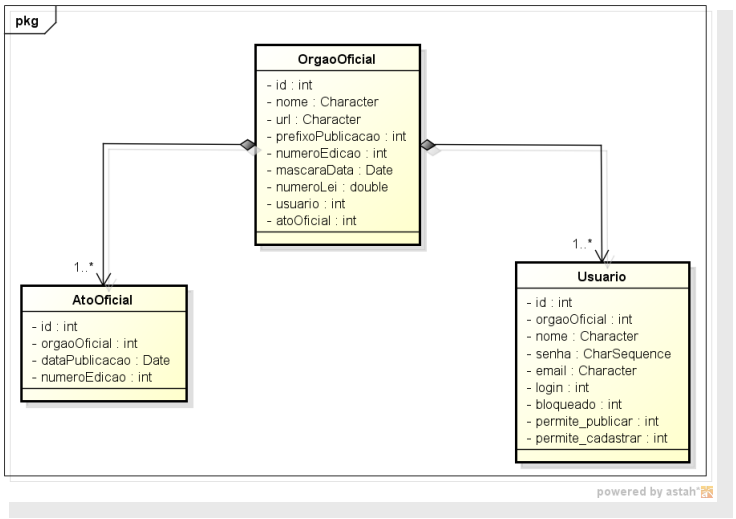


**Figura 8 - Diagrama de Classes: E-Atos**
**Fonte: Leonhart (2011)**

[9] Diário oficial eletrônico

De acordo com o diagrama de classes apresentado na Figura 8 um órgão oficial pode publicar um ou mais atos oficiais, dessa maneira um órgão oficial é composto por um ou vários atos oficiais. Nesse mesmo contexto, um órgão oficial pode ser acessado por um ou muitos usuários.

## 3.2 FERRAMENTAS

As ferramentas utilizadas nesse estudo são *Astah Community* e UML (*Unified Modeling Language*). Os quais irão ajudar na compreensão dos softwares por meio de diagramas de classes e atividades.

### 3.2.1 ASTAH COMMUNITY

A ferramenta *Astah Community* é utilizada no desenvolvimento da modelagem de software, é *open source*. Dentre esses podem ser citados a possibilidade de desenvolver diagramas como: diagramas de caso de uso, diagramas de classes, diagramas de sequência, diagrama de estados, diagrama de atividades, diagrama de componentes, diagrama de implantação, diagrama de estrutura composta, diagrama de comunicação, e diagrama de pacote (ASTAH COMMUNITY, 2014).

### 3.2.2 UML

É a especificação da OMG[10] mais usada, é a forma de modelar não só a estrutura da aplicação, comportamento e arquitetura, mas também os processos de negócios e a

[10] OMG – organização responsável pela aprovação de padrões para aplicações orientadas a objetos.

estruturação de dados. “A UML é uma linguagem-padrão para descrever/documentar projeto de software” (PRESSMAN, 2011).

Segundo PRESSMAN (2011) os diagramas UML são criados com o objetivo de auxiliar os desenvolvedores na construção do software em si, dessa maneira tendo a compreensão de elementos visuais do diagrama e seus significados há também a possibilidade de especificar um sistema bem como esclarecer tal projeto para terceiros.

A UML fornece técnicas para demonstrar aspectos tanto estruturais quanto comportamentais do sistema. Referente à estrutura do sistema podem ser citados os diagramas de classe e disponibilização.  E no que diz respeito ao aspecto comportamental têm-se os diagramas de estados, sequência, colaboração, atividades e casos de uso, sendo que os de estados enfatizam principalmente nos componentes individuais (valores de variáveis em um determinado instante), e os demais ressaltam os elementos estruturais, totalizando treze diagramas para uso na modelagem de software.

# 4 RESULTADOS E DISCUSSÕES

A seção seguinte trata de duas das abordagens de reuso descritas nesse trabalho e que foram aplicadas nos sistemas Birô de Digitalização e Digitaldoc respectivamente. As abordagens de reuso: padrão de requisitos encapsulados e padrão de qualidade não serão utilizadas, pois a primeira se refere principalmente a casos de uso, isto é, extrai duplicidades do documento de requisitos. A segunda, por sua vez, é referente à RNF e conforme descrito anteriormente esses são mais complexos de serem detectados porque não são mapeados diretamente nas funcionalidades, mas ao longo do desenvolvimento do software. E também porque o escopo do trabalho é baseado em RFs.

## 4.1 PROCEDIMENTOS DE APLICAÇÃO

As seções 4.1.1 e 4.1.2 apresentam a aplicação de duas técnicas de reuso de requisitos: analogia entre sistemas de software e o reuso de requisitos. Essas técnicas foram utilizadas porque se ajustam nas definições descritas na literatura, isto é, é perceptível um padrão quanto aos softwares citados (Birô de Digitalização e Digitaldoc) no que diz respeito à suas atividades e RFs.

### 4.1.1 ANALOGIA ENTRE SISTEMAS DE SOFTWARE APLICADOS NOS SISTEMAS: BIRÔ DE DIGITALIZAÇÃO E DIGITALDOC

Esse método de reuso será demonstrado por meio do diagrama de atividades dos sistemas Birô de Digitalização e Digitaldoc, já que ambos os sistemas suprem a necessidade de digitalização e cadastro de documentos.

Na Figura 9 é apresentado o diagrama de atividades referente ao sistema Birô de Digitalização.

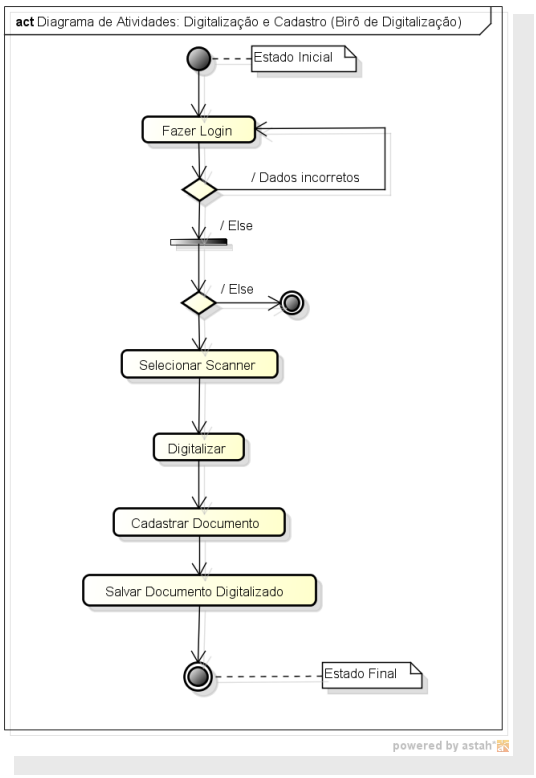


**Figura 9 - Diagrama de Atividades: Birô de Digitalização**

Na Figura 10 é apresentado o diagrama de atividades referente ao sistema Digitaldoc.

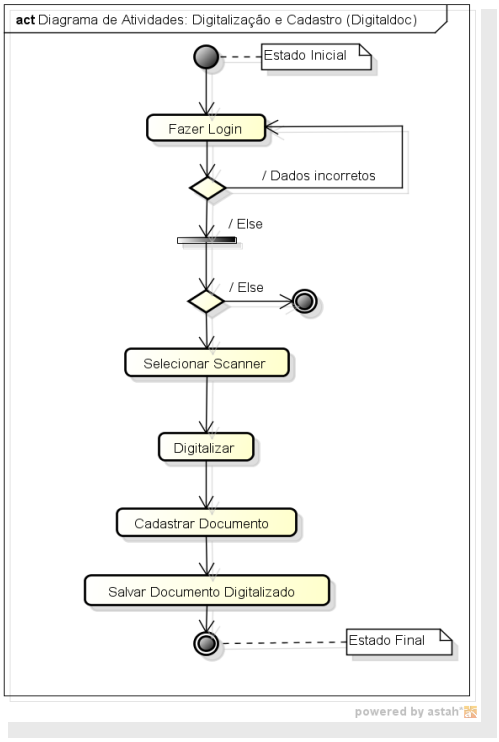


**Figura 10 - Diagrama de Atividades: Digitaldoc**

Mediante os diagramas apresentados nessa seção é demonstrada a atividade que cada sistema realiza. Dessa maneira, na Figura 9 é demonstrado o processo do sistema Birô de Digitalização - que conforme citado na seção 3.1.1 foi desenvolvido exclusivamente para prestar serviços de digitalização, isto é, converter o documento para o formato digital facilitando o acesso ao mesmo. Por sua vez, na Figura 10 é demonstrado o processo do sistema Digitaldoc – que de acordo com a seção 3.1.2 foi desenvolvido para realizar a organização de documentos como um todo, ou seja, uma de suas funções é realizar a digitalização dos documentos mantendo esses em formato digital, eliminando a necessidade do uso de papel permitindo maior acessibilidade ao mesmo. Em ambos os sistemas é notório a semelhança nas atividades de cadastro e digitalização, tanto no sistema Birô de Digitalização como no Digitaldoc é necessário cadastrar e digitalizar o documento. Também é fundamental selecionar o scanner em ambos os sistemas para realizar o processo de digitalização, caso contrário o processo é finalizado. E por fim, é de suma importância que o documento digitalizado seja salvo, garantindo a integridade do mesmo.

Verificou-se que a analogia entre esses dois sistemas Birô de Digitalização e Digitaldoc é comprovada de forma parcial, por meio das atividades de cadastro e digitalização de documentos, pois em ambos os sistemas é necessário realizar o cadastro do documento após realizar a digitalização. E também pelo fato de que o sistema Digitaldoc possui outras funções implementadas.

### 4.1.2 REUSO POR RECICLAGEM DE REQUISITOS

A seguir serão apresentados seis RFs: os três primeiros quadros referem-se ao sistema Digitaldoc e os demais ao sistema Birô de Digitalização. Nesses quadros são listadas informações detalhadas sobre os requisitos, RF e RNF, e a maneira como são utilizados no sistema.

As siglas C, D e P se referem à classificação dos RNFs e significam: categoria, desejável e permanente respectivamente.

Na coluna categoria estão definidos como são utilizados cada RNF, isto é, interface (I), usabilidade (U), performance/desempenho (P) ou segurança (S). É importante destacar que algumas ações são realizadas pelo sistema, ou seja, sem o conhecimento do usuário.

Dessa maneira, quando a categoria é de interface significa que são informações apresentadas na tela do sistema, para o usuário. A categoria de usabilidade deve auxiliar o usuário na utilização do sistema. Por sua vez, a categoria de performance é uma resposta do sistema à ação do usuário. Por fim, não menos importante, a categoria de segurança serve para manter a integridade dos dados garantindo a segurança referente aos dados do usuário. Quando a coluna D (desejável) está marcada, é importante que essa ação seja efetuada, porém, não obrigatória. E quando a coluna P (permanente) está marcada significa que a ação para determinado requisito é permanente, isto é, será efetuada sempre.

No Quadro 1 é apresentado o RF do sistema Digitaldoc, cadastrar usuário, bem como os RNFs que correspondem à esse requisito.

| F 01 – Cadastrar usuário | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema realiza o cadastro de usuários para acesso ao sistema e armazenamento de dados | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 1.1 Identificação do usuário | A identificação do usuário é feita por meio dos dados pessoais (nome, e-mail, CPF, telefone, senha, confirmação de senha), incluindo a permissão para o tipo de usuário e o grupo que o mesmo pertence. | I | ( ) | (X) |
| NF 1.2 Pesquisa de usuário | O sistema possui campos de pesquisa que são realizados por meio do nome do usuário e e-mail. | U | ( ) | (X) |
| NF 1.3 Janela única | As funções relacionadas ao cadastro do usuário serão efetuadas em uma única janela. | I | ( ) | (X) |
| NF 1.4 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 1.5 Campos Obrigatórios | O sistema possui como campos obrigatórios nome, e-mail, senha, confirmação de senha e associação à uma empresa. | S | ( ) | (X) |

**Quadro 1 – Digitaldoc - Requisito Funcional: Cadastrar Usuário**

No Quadro 2 é apresentado o RF, "Fazer *Login*"**,** bem como os RNFs que o correspondem.

| F 02 – Fazer Login | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema permite ao usuário realizar o login para acesso ao mesmo | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 2.1 Identificação do usuário | A identificação do usuário é feita por meio dos seguintes dados: e-mail e senha | I | ( ) | (X) |
| NF 2.3 Janela única | As funções relacionadas ao login do usuário no sistema serão efetuadas em uma única janela. | I | ( ) | (X) |
| NF 2.4 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 2.5 Campos Obrigatórios | O sistema possui como campos obrigatórios: e-mail e senha. | S | ( ) | (X) |

**Quadro 2 - Digitaldoc - Requisito Funcional: Fazer Login**

No Quadro 3 é apresentado o RF cadastrar documento, juntamente com os RNFs .

| F 06 - Cadastrar documento | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema realiza o cadastro de documentos para digitalizaçao, pesquisa e armazenamento de dados | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 6.1 Dados do documento | Os dados do documento a serem preenchidos são: pasta de origem e destino, titulo, tipo de documento, descrição e palavras-chave. | I | ( ) | (X) |
| NF 6.2 Janela única | As funções relacionadas ao cadastro do documento serão efetuadas em uma única janela. | I | ( ) | (X) |
| NF 6.3 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 6.3 Campos Obrigatórios | O sistema possui como campos obrigatórios os campos pasta de origem, pasta de destino, título e tipo de documento. | S | ( ) | (X) |

**Quadro 3 - Digitaldoc - Requisito Funcional: Cadastrar Documento**

No Quadro 4 é apresentado o RF, cadastrar usuário, e os RNFs referentes ao Birô de Digitalização.

| F 01 - Cadastrar usuário | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema realiza o cadastro de usuários para acesso ao sistema e armazenamento de dados | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 1.1 Identificação do usuário | A identificação é feita por meio dos dados: nome de usuário (*login*), senha e nome completo, incluindo a permissão para o tipo de usuário e o grupo que o mesmo pertence (administrador, digitalizador, indexador ou somente visualizador. | I | ( ) | (X) |
| NF 1.2 Pesquisa de usuário | O sistema possui campos de pesquisa que são realizados por meio do nome do usuário(login) e pelo nome completo. | U | ( ) | (X) |
| NF 1.3 Janela única | As funções relacionadas ao cadastro do usuário serão efetuadas em uma única janela. | I | ( ) | (X) |
| NF 1.4 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 1.5 Campos Obrigatórios | O sistema possui como campos obrigatórios usuário, senha, nome completo e selecionar uma permissão. | S | ( ) | (X) |

**Quadro 4 - Birô de Digitalização - Requisito Funcional: Cadastrar Usuário**

No Quadro 5 é apresentado o RF, "Fazer *Login*", seguidos dos requisitos RNFs do sistema Birô de Digitalização**.**

| F 02 – Fazer Login | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema permite ao usuário realizar o login para acesso ao mesmo | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 2.1 Identificação do usuário | A identificação do usuário é feita por meio dos seguintes dados: login e senha. | I | ( ) | (X) |
| NF 2.3 Janela única | As funções relacionadas ao login do usuário no sistema são efetuadas em uma única janela. | I | ( ) | (X) |
| NF 2.4 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 2.5 Campos Obrigatórios | O sistema possui como campos obrigatórios: login e senha | S | ( ) | (X) |

**Quadro 5 - Birô de Digitalização - Requisito Funcional: Fazer Login**

No Quadro 6 é apresentado o RF, cadastrar documento, juntamente com seus RNFs.

| F 06 - Cadastrar documento | Oculto () | | | |
|---|---|---|---|---|
| Descrição: O sistema realiza o cadastro de documentos para digitalizaçao, pesquisa e armazenamento de dados | | | | |
| Requisitos Não-funcionais | | | | |
| Nome | Restrição | C | D | P |
| NF 6.1 Dados do documento | Os dados do documento a serem preenchidos são: tipo de documento, referência (ou código), data de cadastro, descrição, palavras-chave. | I | ( ) | (X) |
| NF 6.2 Janela única | As funções relacionadas ao cadastro do documento serão efetuadas em uma única janela. | I | ( ) | (X) |
| NF 6.3 Tempo de registro | O tempo para registrar o cadastro deverá ser efetuado em um tempo inferior a três segundos. | P | (X) | ( ) |
| NF 6.3 Campos Obrigatórios | O sistema possui como campos obrigatórios tipo de documento, referência e data do documento. | S | ( ) | (X) |

**Quadro 6 - Birô de Digitalização - Requisito Funcional: Cadastrar Documento**

Nos Quadros 1 e 4, em ambos os sistemas, o RF cadastrar usuário é idêntico bem como sua descrição. Nos RNFs há certa semelhança entre os mesmos nos campos de identificação e pesquisa do usuário, por exemplo.

Nos Quadros 2 e 5 que descrevem o acesso ao sistema, os RFs também são idênticos. Nesse caso a semelhança entre os RNFs é ainda maior, isto é, o que diferencia esses requisitos são os campos e-mail e *login*.

Por fim, nos Quadros 3 e 6, nos quais é descrito o RF cadastrar documento, nota-se que sua descrição é idêntica. Apesar da semelhança entre os RNFs, esses são distintos pelos dados do documento a serem fornecidos em cada sistema.

Como resultado do comparativo entre os RFs dos sistemas Digitaldoc e Birô de Digitalização, é comprovado o reuso de requisitos, já que os mesmos são idênticos em todas as situações apresentadas. No tocante aos RNFs, no caso dos sistemas apresentados, são semelhantes. Contudo esses poderiam ser totalmente distintos, ainda assim a aplicação do reuso seria possível.

# 5 CONSIDERAÇÕES FINAIS

## 5.1 CONCLUSÃO

Após o desenvolvimento da pesquisa referente às abordagens de reuso de requisitos e do estudo de caso abrangendo os sistemas de software para gestão eletrônica de documentos, Birô de Digitalização e Digitaldoc, conclui-se que:

Foi importante o estudo de conceitos referentes à LPS, padrões de software, *frameworks*, componentes, bibliotecas de classes e reuso de requisitos. Nesse contexto, foram percebidos alguns objetivos em comum: redução de tempo, custos e esforços.

Por meio do desenvolvimento do estudo sobre as abordagens de reuso de requisitos em famílias de software foi confirmada sua aplicação: a reutilização de requisitos é útil em sistemas similares, isto é, é possível sua aplicação em sistemas de softwares que compartilhem requisitos e restrições e que possuam a mesma área de aplicação.

O desenvolvimento do estudo de caso sobre os sistemas de gestão eletrônica de documentos permitiu maior conhecimento sobre os mesmos, mais especificamente sobre seus requisitos funcionais. De um total de oito requisitos, sete foram reaproveitados e com isso a possibilidade de aplicação do estudo realizado nos sistemas de software apresentados.

Nas duas abordagens de reuso aplicadas: analogia entre sistemas de software e reuso de requisitos foi comprovada a aplicação das mesmas. Na primeira foi possível uma aplicação parcial, pois apesar dos softwares serem da mesma família (área de aplicação) suas atividades diferem porque o sistema Digitaldoc possui mais funções. Na segunda, foi possível o reuso de alguns RFs, aproximadamente 87,5% dos requisitos foram reusados. É importante destacar que os softwares Birô de Digitalização e Digitaldoc foram desenvolvidos sem analisar a possibilidade de reuso. Em ambas as situações foram comprovadas a aplicação dos conceitos de reuso ainda que, em algumas ocasiões de maneira parcial.

Não é possível apresentar resultados, no que diz respeito a valores percentuais, quanto à redução do tempo ou melhoria da qualidade, devido ao fato que a maioria dos desenvolvedores dos softwares citados não se encontra mais na empresa.

De maneira geral, o trabalho foi proveitoso, porém, moroso pelo fato dos softwares apresentados não possuírem documentação necessária para aplicação no tema em estudo.

Também foi necessário o desenvolvimento dos diagramas apresentados, o que resultou na busca pelas informações diretamente com alguns dos desenvolvedores dos softwares.

## 5.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Sugerem-se para trabalhos futuros os temas:

- Desenvolver um procedimento de análise para comparar os requisitos, no que diz respeito à similaridade e quantidade dos mesmos.
- Aplicação da técnica de reuso: o padrão requisitos encapsulados que auxilia na melhor compreensão dos requisitos retirando as duplicações, caso essas existam.
- Aplicação da técnica de reuso o padrão requisitos encapsulados no sistema E-Atos, aplicado diretamente nos casos de uso do sistema.
- Estudo e possível aplicação da técnica de reuso: o padrão requisitos da qualidade que utiliza como base requisitos não funcionais.

# REFERÊNCIAS BIBLIOGRÁFICAS

610.121990, I. S. IEEE Standard Glossary of Software Engineering Terminology. **IEEE**, New York, 28 set. 1990. Disponivel em: <http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf>. Acesso em: janeiro 2013.

ALEXANDER, C. **The Timeless Way of Building**. Oxford University Press. New York. 1979.

ASTAH. Disponivel em: <http://astah.net/editions/community>. Acesso em: 06 setembro 2016.

AURÉLIO, D. Dicionário do Aurélio. **Dicionário Aurélio Online**, 2008-2013. Disponivel em: <http://www.dicionariodoaurelio.com/>. Acesso em: 11 julho 2013.

CARBONELL, J. G. **Derivational Analogy: A Theory of reconstructive Problem Acquisition**. University Mellon-Carnegie Computer Science Department. [S.l.]. 1985.

CLEMENTS, P.; NORTHROP, L. **Software Products Line:** Pratices and Patterns. Boston: [s.n.], 2001.

DANEVA, M. **ERP Requeriments Engineering Pratice: Lessons Lernead**. IEEE Software. [S.l.]. 2004.

DIGITALDOC. **Documentos da Empresa**, 03 outubro 2013.

EDWEISS, N. **Sistemas de Informação de Escritórios: Um modelo para especificações temporais**. CPGCC - UFRGS. [S.l.]. 1994.

EMANUEL COUTINHO; CARLA MOREIRA; ROSSANA ANDRADE; JOSÉ NEUMAN DE SOUZA. **Requisitos de Qualidade: Um Padrão para Identificação de Requisitos Não-Funcionais Conflitantes**. Universidade Federal do Ceará. [S.l.]. 2008.

FERGUSON, J. et al. **C# Bible**. [S.l.]: [s.n.], 2002.

FIGUEIREDO, E. **O Processo CBSE**. Disponivel em: <http://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/cbse-processos_v01.pdf>. Acesso em: outubro 2016.

FILHO, J. L.; LOCHPE, C. Mecanismos de Reutilização em Sistemas de Informação. **Reuse Mechanisms in Infomation Systems**.

GROUP, O. M. OMG We Set the Standart. **About OMG**, 2012. Disponivel em: <www.omg.org>. Acesso em: 10 jul. 2012.

J., C. Software Product Lines. **Software Engineering Institute**, 2013. Disponivel em: <http://www.sei.cmu.edu/productlines/>. Acesso em: 30 junho 2013.

J.R.LOBO, E. **Curso de Engenharia de Software - Métodos e processos para garantir a qualidade no desenvolvimento de softwares**. São Paulo: Universo dos Livros Editora LTDA, 2008.

JACOBSON, I. **The Unified Software Development Process**. [S.l.]. 2005.

JÚNIOR, H. E. **Engenharia de Software na Prática**. São Paulo: Novatec, 2010.

KANG, W. F. &. K. **Software Reuse Research: Status and Future**. [S.l.]. 2005.

KONSTANTIN ROMANOVISKY; DIMITRY KOZNOV; LEONID MINCHIN. **Refactoring the Documentation of Software**. Saint-Petesburg University. [S.l.]. 2008.

KOTONYA, P.; SOMERVILLE, I. **RequerimentsEngineering:** Processes and Techniques. [S.l.]: [s.n.], 1998.

KRUGER, C. W. Software Reuse. **Revision Zero - Pesquisas de Computação da ACM**, 2011. Disponivel em: <http://www.revision-zero.org/reuse>. Acesso em: 2012.

LEONHART, N. D. S. **Certificação Digital e Carimbo do Tempo no Modelo do Diário Oficial Eletrônico dos Municípios**. Universidade Tecnológica Federal do Paraná. Medianeira. 2011.

MAIDEN, N.; SUTCLIFFE, A. **Exploiting Reusable Specifications Through Analogy**. [S.l.]: [s.n.], v. 35, 1992.

MARIANI, T. Reuso de Software. Disponivel em: <http://www.inf.ufpr.br/silvia/ES/reuso/reuso.pdf>. Acesso em: 08 ago. 2016.

MARTINS, M. R. **Ferramenta de Suporte a Reuso de Casos de Uso**. Universidade Regional de Blumenau. Blumenau. 2006.

MICHAELIS, D. M. Michaelis Moderno Dicionário. **Dicionário Online - Dicionários Michaelis - UOL**, 1998-2009. Disponivel em: <http://michaelis.uol.com.br/>. Acesso em: 11 julho 2013.

NEIGHBORS, J. M. **An Assesment of Reuse Technology**. International Conference on Software Reuse. California. 1994.

ORACLE. Enterprise Java Beans Technology. **Oracle Technology Network**. Disponivel em: <http://www.oracle.com/technetwork/java/javaee/ejb/index.html>. Acesso em: 11 ago. 2016.

PIBBER, R. V. **Reuso de Modelos de Dominio de Processos de Negócio Através de Analogia**. Instituto Andreas Palladio. [S.l.]. 1996.

PIMENTA, A. **Especificação Formal de uma Ferramenta de Reutilização de Especificações de Requisitos**. Universidade Federal do Rio Grande do Sul. Porto Alegre. 1998.

PRESSMAN, R. S. **Engenharia de Software - Uma Abordagem Profissional 7ª edição**. São Paulo: AMGH, 2011.

QUEIRÓZ, P. G. G. **Uma Abordagem de Desenvolvimento de Linha de Produtos Orientada a Serviços**. Instituto de Ciências Matemáticas e Computação - ICM/USP. São Carlos. 2009.

RENAULT, S.; BONILLA, O. M.; FRACH, X. 8ªMostra Acadêmica Unimep. **8º Congresso de Pós-Graduação - Reuso de Requisitos para Produtos em Sistemas Embarcados**, 2010. Disponivel em: <http://www.unimep.br/phpg/mostraacademica/anais/8mostra/5/376.pdf>. Acesso em: 12 nov. 2012.

ROBERTS, D.; JONHSON, R. Envolving Frameworks: A Pattern Language Developing Object-Oriented Frameworks. Disponivel em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/evolve.html>. Acesso em: 11 agosto 2016.

ROLLAND, C.; PRAKASH, N. From Conceptual Modelling to Requirements Engineering, 1999. Disponivel em: <http://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/488/369>. Acesso em: 11 dez. 2012.

SAMPAIO, M. C., 2007. Disponivel em: <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Analise/>. Acesso em: 14 set. 2016.

SILVA, R. C. D. **UMA ABORDAGEM PARA O REUSO DE REQUISITOS**. Universidade do Vale do Itajaí. São José. 2011.

SILVA, R. C. D. **Uma Abordagem para o Reuso de Requisitos Baseada em Padrões e Rastreabilidade**. Universidade do Vale do Itajaí. São José (SC). 2011.

SILVA, R. P. E. **Suporte ao Desenvolvimento e Uso de frameworks e componentes**. Universidade Federal do Rio Grande do Sul - Programa de Pós-graduação em Computação. [S.l.]. 2000.

SILVEIRA, M. C. D. S. P. **A Reutilização de Requisitos no Desenvolvimento e Adaptação de Produtos de Software**. Universidade do Porto. [S.l.]. 2006.

SILVEIRA, M. C.; VIDAL, R. M. **Reutilização de Casos de Uso no Desenvolvimento de Sistemas de Software**. Orlando, EUA: [s.n.], v. I, 2002.

SOMERVILLE, I.; SAWYER, P. **Requirements Engineering: A Good Practice Guide**. [S.l.]. 1997.

SOMMERVILLE, I. **Engenharia de Software 9ª edição**. São Paulo: Pearson, 2011.

SPAGNOLI, L. D. A.; BECKER, K. Um Estudo Sobre Desenvolvimento Baseado em Componentes. Disponivel em: <http://www.pucrs.br/facin-prov/wp-content/uploads/sites/19/2016/03/tr026.pdf>. Acesso em: 11 ago. 2016.

SZYPERSKI, C. **Sumary of the First International Workshop on component-oriented programming**. [S.l.]: [s.n.], 1996.

SZYPERSKI, C. **Component Software**. 2. ed. [S.l.]: [s.n.], 2002.

SZYPERSKI, C. **Sumary of the Second International Workshop on component-oriented programming**. [S.l.]: [s.n.], 997.

TONIOLO, C. M. 8ªMostra Acadêminca Unimep - Anais. **8º Congresso de Pós-Graduação - Reuso de Requisitos para Produtos em Sistemas Embarcados**, 2010. Disponivel em: <http://www.unimep.br/phpg/mostraacademica/anais/8mostra/5/376.pdf>. Acesso em: 12 novembro 2012.

VILLEGAS, O. L.; LAGUNA, M. A. Requirements Reuse for Software Development, 2001. Disponivel em: <http://giro.infor.uva.es/Publications/2001/LL01/Doc-Workshop.pdf>. Acesso em: 10 nov. 2012.

WAZLAWICK, R. S. **Análise e Projeto de Sistemas de Informação Orientados a Objeto**. São Paulo: Elsevier, 2004.

ZAVE, P. **classification of Research Efforts in Requeriments Engineering**. [S.l.]: ACM Computing Surveys, v. 29, 1997.

ZIRBES, S. F. **A Reutilização de Modelos de Requisitos de Sistemas por Analogia: Experimentação e Conclusões**. Universidade Federal do Rio Grande do Sul. Porto Alegre. 1995.