SISTEMA DE VENDA E CONTROLE DE ESTOQUE: Levantamento e Análise de Requisitos

Gabriel Henrique da Silva
Discente do curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)

Rafael Alves da Silva
Discente do curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)

Alan Pinheiro Souza
Docente do curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)
Mestre em Informática pela Universidade Federal do Rio de Janeiro (UFRJ)

RESUMO
A utilização de tecnologias na sociedade contemporânea é imprescindível, pois essas ferramentas estão presentes em diversos aspectos da vida humana, inclusive nos sistemas críticos que afetam a nossa saúde e o nosso bem-estar. Quando compras semanais ou mensais são realizadas no mercado, torna-se incômodo quando algum desses estabelecimentos não possuem um sistema de automação eficiente, pois o cliente está sempre com pressa, seu desejo é ser atendido rapidamente e com qualidade. Dessa forma, esse projeto teve como objetivo realizar levantamento e análise de requisitos visando desenvolvimento de um sistema computacional que auxilie na venda e controle de estoque em estabelecimento.

PALAVRAS-CHAVE: *Software*; Mercado; Controle; Compra; Venda; Produto.

INTRODUÇÃO

A proposta do trabalho é desenvolver um *software* que utilize métodos e técnicas da engenharia de requisitos visando desenvolver um *software* para apoiar atividades gerenciais e operacionais de um estabelecimento comercial do ramo de conveniência. A empresa é composta apenas pelo proprietário e, por ser recente no mercado, não possui sistemas computacionais. Todos os processos do negócio são feitos manualmente. Feitas essas constatações nas conversas iniciais estabelecidas junto ao proprietário, foi proposta a elaboração de um projeto de automação da empresa, de modo a disponibilizar um melhor atendimento aos clientes. Nesse artigo, a primeira seção apresenta um plano de projeto com justificativas e objetivos, participantes, papéis e responsabilidades da equipe e informações do processo de desenvolvimento. A segunda seção descreve a etapa de levantamento de requisitos com identificação e modelagem das características do *software*.

1 PLANO DE PROJETO

A modernização tecnológica é aceita como um dos fatores fundamentais para dinamizar o processo de desenvolvimento corporativo. Para Pressman (2006, p.628), quanto maior o ritmo de desenvolvimento que se deseja imprimir em uma empresa ou em uma sociedade como um todo, mais elas devem estar vinculadas ao uso de tecnologias adequadas às suas matérias primas, aos seus recursos humanos e a sua realidade. O desafio das corporações é determinar o mais precisamente possível quais dos benefícios realmente são ofertados e desejados, pois esta identificação será a base para a confirmação de melhorias em seu desempenho.

No contexto de projeto de *software*, um engenheiro necessitará realizar um planejamento para identificar justificativas e objetivos, quais atividades deverão ser executadas, compreender riscos, estipular compromissos e tomar decisões de projeto. Essa guia oferece uma base sistemática de como conduzir o projeto e quaisquer modificações necessárias ao longo do processo de desenvolvimento, além de servir como eficiente mecanismo de comunicação entre os principais interessados no projeto (SILVA FILHO, 2008, p.22).

1.1 Justificativas e Objetivos

A partir do interesse de ampliar e automatizar a empresa, crescer futuramente seu quadro de funcionários e disponibilizar um melhor atendimento ao seu cliente, foi proposto ao cliente um sistema informatizado para fazer cadastro de funcionários, clientes e fornecedores, além de controle de entrada e saída de produtos, a partir de registro das operações de compra e venda. O sistema também permitirá geração de relatórios por parte da gerência para que possam ser realizadas análises mais precisas das atividades corporativas, visando tomadas de decisões mais assertivas. O objetivo desse projeto é criar um *software* que solucione os problemas encontrados na empresa. Espera-se que, a partir da adoção do sistema, a empresa consiga coletar, tratar e distribuir informações de maneira eficiente, além de alavancar novos produtos e serviços, assim como se destacar perante seus concorrentes, e promover satisfação dos seus clientes.

1.2 Participantes, Papéis e Responsabilidades do Projeto

Essa seção apresenta os participantes do projeto, assim como seus papéis e responsabilidades. A equipe de desenvolvimento, como apresentada na Figura 1, será formada por dois membros que assumirão várias funções, em razão do número reduzido de integrantes envolvidos no projeto. O proprietário da empresa será cliente do projeto fornecendo apoio informacional necessário ao entendimento do contexto e validação da solução a ser proposta e desenvolvida. Segundo Silva Filho (2008, p.27), profissionais da área de desenvolvimento podem assumir as seguintes funções e atribuições:

- **Gerente de Projeto:** responsável pelo planejamento e monitoramento do projeto, verificação de equipe e orçamento, comunicação com o cliente sobre andamento do serviço e coordena interação entre equipe e cliente;
- **Analista de Negócio:** modelagem do negócio, análise e projeto, detalhamento de requisitos e elaboração de documentação;
- **Arquiteto de *Software*:** definição de arquitetura do sistema, integração de componentes e participação nos testes de integração;
- **Desenvolvedor:** envolvido na implementação de componentes, realização de testes unitários e integrados do sistema;
- **Engenheiro de Qualidade:** documentação e auditoria do processo.

Figura 1: Participantes do Projeto.

NOME DOS PARTICIPANTES	PAPÉIS (FUNÇÕES)
Gabriel Henrique da Silva	Gerente de Projeto, Analista de Negócio, Arquiteto de Software, Desenvolvedor e Engenheiro de Qualidade
Rafael Alves da Silva	Gerente de Projeto, Analista de Negócio, Arquiteto de Software, Desenvolvedor e Engenheiro de Qualidade
Marcelo Oliveira	Cliente (Gerente e Proprietário da Empresa)

Fonte: Elaborado pelos autores.

1.3 Processo de Desenvolvimento

Um modelo de desenvolvimento corresponde a uma representação abstrata do processo de desenvolvimento que vai, em geral, definir como as etapas relativas ao desenvolvimento do *software* serão conduzidas e inter-relacionadas para atingir o

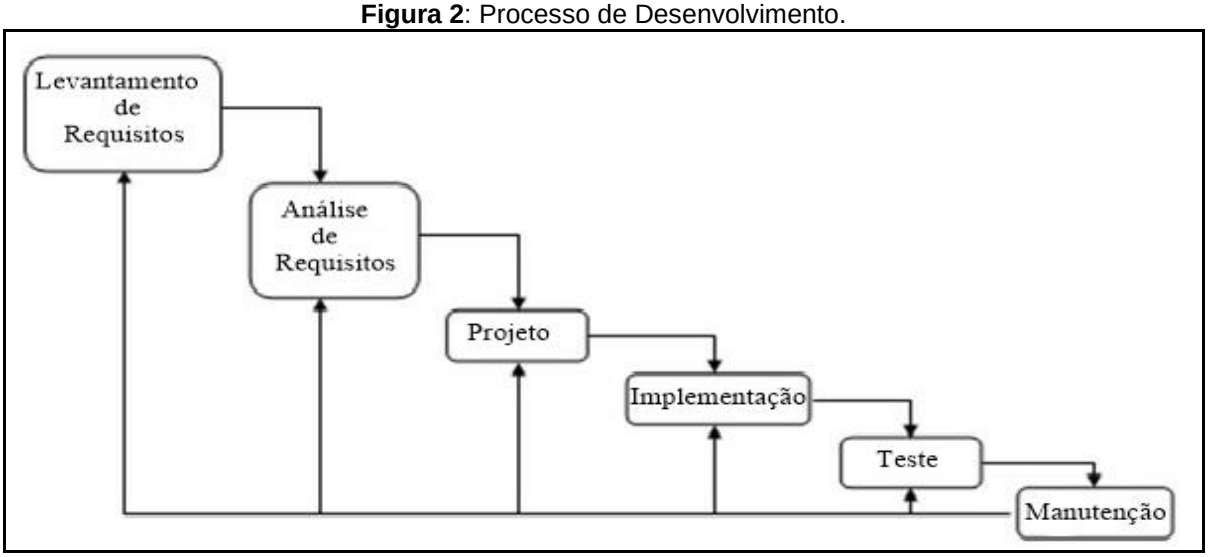
objetivo do desenvolvimento que é a obtenção de um produto de *software* de alta qualidade a um custo relativamente baixo.

O modelo clássico ou cascata, que também é conhecido por abordagem "*top-down*", foi proposto por Royce em 1970. Até meados da década de 1980, este foi o único modelo com aceitação geral entre engenheiros de sistema. Segundo Bezerra (2006, p.96), o resultado de cada fase envolve um ou mais documentos que são aprovados (assinados), assim a fase seguinte não deve iniciar até que aquela precedente tenha sido concluída. Na visão de Engholm (2010, p.183), esse modelo de desenvolvimento de *software* foi divido em seis fases que visam produzir maior estabilidade e confiança na construção de um *software*.

- **Levantamento de Requisitos:** trata das restrições e dos objetivos do sistema, estabelecidos por meio da consulta aos usuários do sistema. Em seguida, os detalhes para a especificação do sistema são averiguados;
- **Análise de Requisitos:** envolve estudo das necessidades do usuário para se encontrar uma definição correta ou completa do sistema ou requisito de *software*. Os requisitos colhidos devem ser quantitativos, detalhados e relevantes para o projeto;
- **Projeto:** realiza agrupamento dos requisitos em sistemas de *hardware* e *software*. Feito isso, ocorre a identificação e a descrição das abstrações fundamentais do sistema;
- **Implementação:** atividades de codificação, compilação, integração e testes. A implementação visa traduzir o *design* num programa, utilizando linguagens e ferramentas adequadas;
- **Teste:** nessa etapa, as unidades de programa são integradas e testadas como um sistema completo. Esse procedimento ocorre para garantir que todos os requisitos foram atendidos. O *software* é entregue ao cliente após os testes alcançarem resultado positivo;
- **Manutenção:** fase em que o sistema é colocado em operação. A manutenção envolve corrigir erros que não foram descobertos nas fases anteriores do ciclo de vida. Dessa forma, ocorrerá uma nova análise para melhorar a implementação das unidades do sistema.

A Figura 2 exibe o modelo de desenvolvimento de *software*, que aborda todas as fases de criação de um *software*. Com base nisso, decidiu-se utilizar um modelo adaptado de Engholm (2010, p.58), que envolve o modelo cascata (visto como um fluxo constante para frente), visto que as atividades são executadas e agrupadas em tarefas, sequencialmente, de forma que cada tarefa só poderá iniciar quando a anterior estiver encerrada. As adaptações ocorrem no momento em que retornos são permitidos para realizar as devidas correções em ações previamente realizadas.

Figura 2: Processo de Desenvolvimento.



Fonte: Adaptado de Engholm (2010, p.58).

2 LEVANTAMENTO DE REQUISITOS

Nessa seção serão apresentadas as características da proposta e os requisitos do sistema. Segundo Pressman (2006, p.133), “o levantamento de requisitos (também chamado de elicitação de requisitos) combina elementos de resolução de problemas, elaboração, negociação e especificação”. Essas informações permitem à equipe de projeto um entendimento sobre as necessidades do cliente e qual o impacto do *software* sobre o negócio. Para execução dessa etapa foi aplicada a técnica de entrevista orientada por questionários. Este artefato abrangia um conjunto de perguntas de modo a obter informações relevantes do contexto sendo analisado. O conteúdo capturado nas entrevistas forneceu subsídios para apoiar as tarefas de modelagem, detalhamento e documentação das características a formarem escopo do sistema computacional a ser desenvolvido.

2.1 Requisitos do Sistema

Após análise das informações coletadas na entrevista, iniciou-se processo de identificação e análise dos requisitos do sistema. Segundo Cordeiro (2010, p.181), os requisitos funcionais são aqueles que definem comportamento do sistema, capturados por meio de casos de uso, que documentam entradas, processos e saídas geradas. Os requisitos são fundamentais para elaborar um sistema ou *software* que atenda e satisfaça plenamente equipe desenvolvedora, clientes e usuários finais (LAWRENCE, 2004, p.82).

No entendimento de Engholm (2010, p.93), a análise de requisitos é uma tarefa de engenharia de *software* que efetua a ligação entre a alocação do *software* em nível de sistema, o projeto de *software* e os anseios do cliente e ou usuário. Para Fagundes (2011, p.75), o grau de prioridade de um requisito para os usuários normalmente é estabelecido em função da adição de valor que o desenvolvimento desse requisito no sistema trouxer aos usuários. Dessa forma, é importante conhecer o grau de prioridade, pois permite que a equipe de desenvolvimento, mais particularmente o gerente de projeto, decida em que momento cada um deve ser considerado durante o desenvolvimento.

A equipe de projeto adotou a prática de modelagem, por intermédio da linguagem de modelagem unificada UML (*Unified Modeling Language*), para apoiar as tarefas de visualização, especificação, construção e documentação do escopo. Nas seções seguintes serão apresentados dois diagramas: de classes com objetivo de fornecer uma representação de conceitos no domínio do problema; e diagrama de casos de uso para identificar regras do negócio e entender o ponto de vista do usuário na sua interação com o *software*. Nesse artigo serão apresentados casos de uso dos eventos “Controlar Compra” e “Controlar Venda”, considerado os mais importantes pela equipe de desenvolvimento e cliente, entretanto, é importante destacar que modelagem similar foi realizada para todos os eventos do sistema

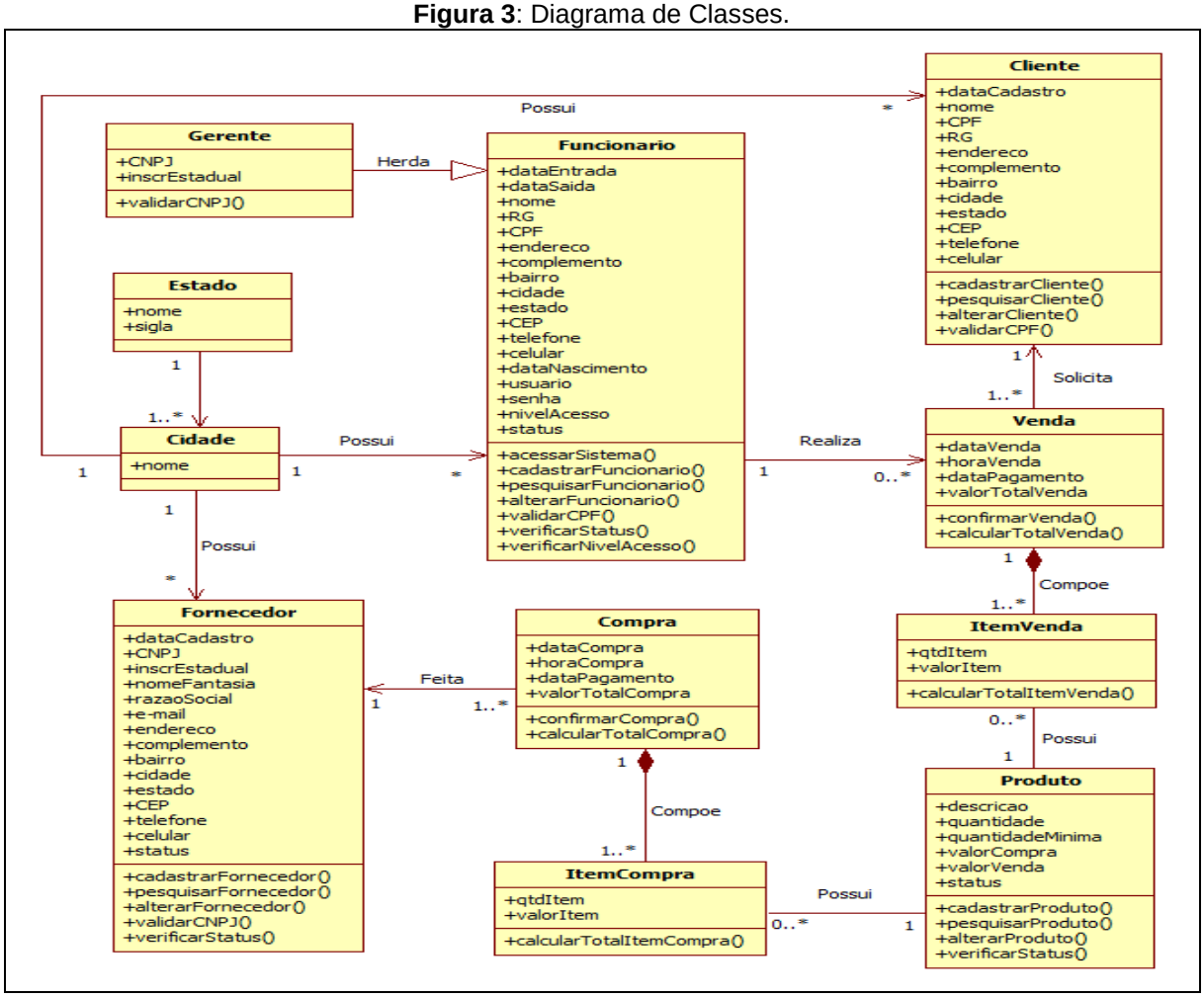
2.2 Diagrama de Classes

O diagrama de classes referencia as diferentes classes que fazem um sistema e como elas se relacionam. O diagrama de classes é classificado como um diagrama estático, pois apresenta as classes, com respectivos métodos e atributos,

e seus relacionamentos, o que permite identificar quais classes “conhecem” outras classes, entretanto não mostram a troca de mensagens entre essas classes. Segundo Booch, Rumbaugh e Jacobson (2000, p.96), os relacionamentos no diagrama de classes são compostos pelas seguintes formas: associação, agregação, composição, generalização (herança) e dependência.

A Figura 3 apresenta o diagrama de classes modelado para o contexto analisado. As seguintes classes foram identificadas: *Gerente*, *Funcionario*, *Cliente*, *Venda*, *ItemVenda*, *Produto*, *ItemCompra*, *Compra* e *Fornecedor*. A classe *Gerente* possui relacionamento de herança com a classe *Funcionario*. Já as classes *ItemVenda* e *Venda* possuem um relacionamento de composição, de maneira similar ao que ocorre para as classes *ItemCompra* e *Compra*. As demais classes possuem relacionamento de associação.

Figura 3: Diagrama de Classes.



Fonte: Elaborado pelos autores.

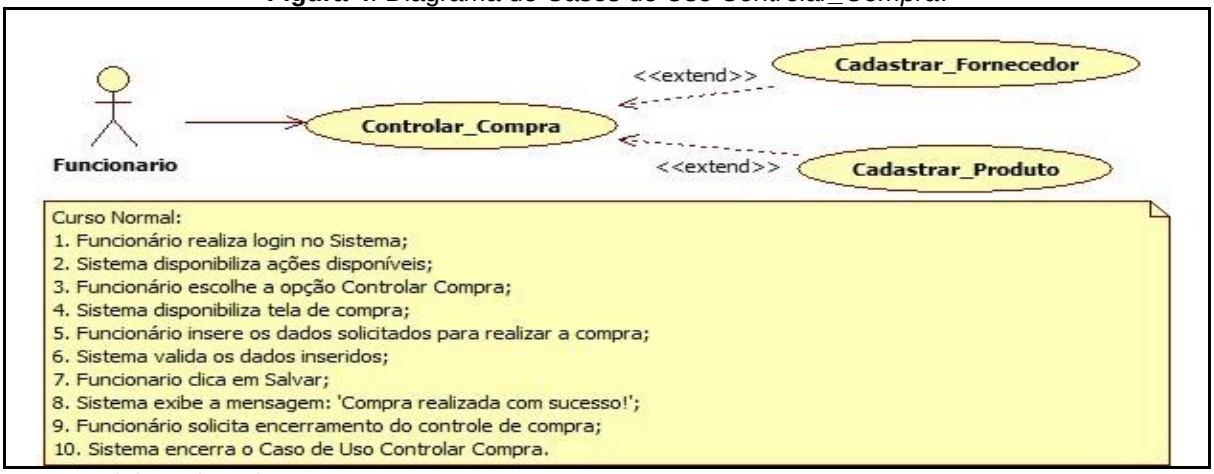
2.3 Diagrama de Casos de Uso

Os diagramas de casos de uso possuem a finalidade de descrever cenários de uma solução a ser desenvolvida e determinar às especificações funcionais de um sistema. Este diagrama comunica funcionalidades e comportamentos do sistema mediante seu uso pelo cliente. Seus principais elementos são os usuários (atores) e os roteiros de uso (curso normal e curso alternativo).

> O diagrama de casos de uso permite uma visão global e de alto nível do sistema, sendo fundamental a definição correta de sua fronteira, são utilizados preferencialmente na fase de especificação de requisitos e na modelagem de processos de negócio (RAMOS, 2006, p.49).

A Figura 4 ilustra o diagrama de casos de uso *Controlar_Compra* e respectivo curso normal das interações do ator *Funcionario* com a funcionalidade para registro de compra de um determinado produto. O curso normal demonstra o funcionamento perfeito do sistema, para cada ação do *Funcionario* existe uma resposta do sistema até que todas as instruções sejam concluídas. As cláusulas *extend* representam variações na execução do caso de uso *Controlar_Compra*; sendo executadas no caso fornecedor (*Cadastrar_Fornecedor*) ou produto (*Cadastrar_Produto*) não tenham sido previamente cadastrados no sistema.

Figura 4: Diagrama de Casos de Uso *Controlar_Compra*.

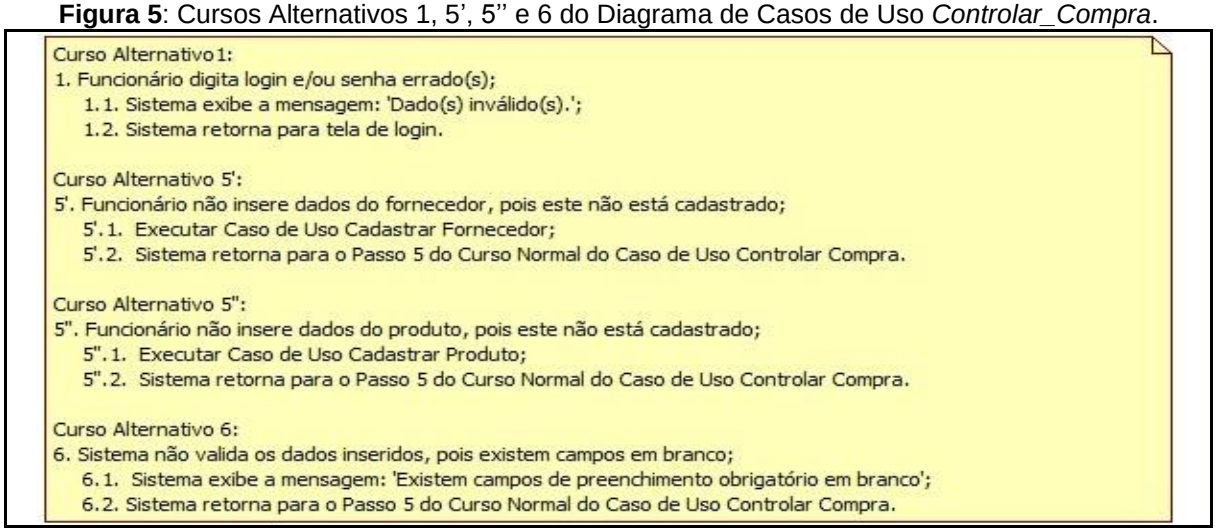


Fonte: Elaborado pelos Autores.

Os cursos alternativos 1, 5', 5" e 6 detalham as possíveis ações do ator *Funcionario* mediante variações de uso da funcionalidade de registro da compra (Figura 5). O curso alternativo 1 do diagrama de casos de uso trata da validação de *login* e senha. Já os cursos alternativos 5' e 5'' realizam, respectivamente, cadastros

de fornecedor e produto, casos esses não tenham sido registrados previamente. Nessas duas situações, após redirecionamento de execução do sistema para os casos de uso *Cadastrar_Fornecedor* e *Cadastrar_Produto*, o fluxo de ações retorna para o caso de uso *Controlar_Compra*. Por último, o curso alternativo 6 permite validação no preenchimento de campos obrigatórios.

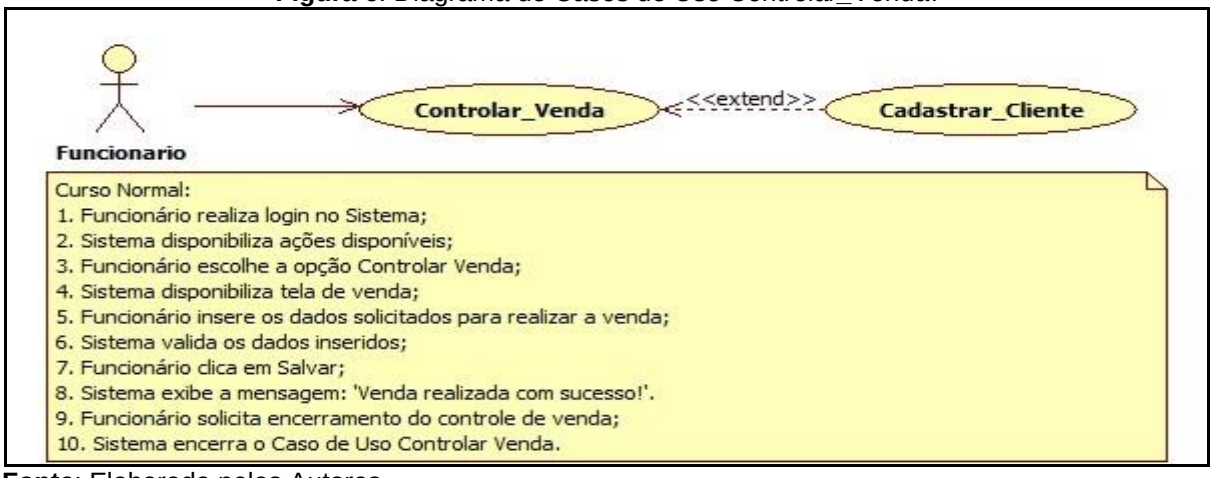
Figura 5: Cursos Alternativos 1, 5', 5'' e 6 do Diagrama de Casos de Uso *Controlar_Compra*.



Fonte: Elaborado pelos Autores.

Outro evento importante do negócio é o procedimento de venda, modelado por intermédio do diagrama de casos de uso *Controlar_Venda*, exibido na Figura 6. Visando incorporar padronização e facilidade de uso ao sistema, esse roteiro segue fluxo similar àquele apresentado no procedimento de compra. A cláusula *extend* para o caso de uso *Cadastrar_Cliente* ocorre somente se cliente não tenha sido previamente cadastrado no sistema.

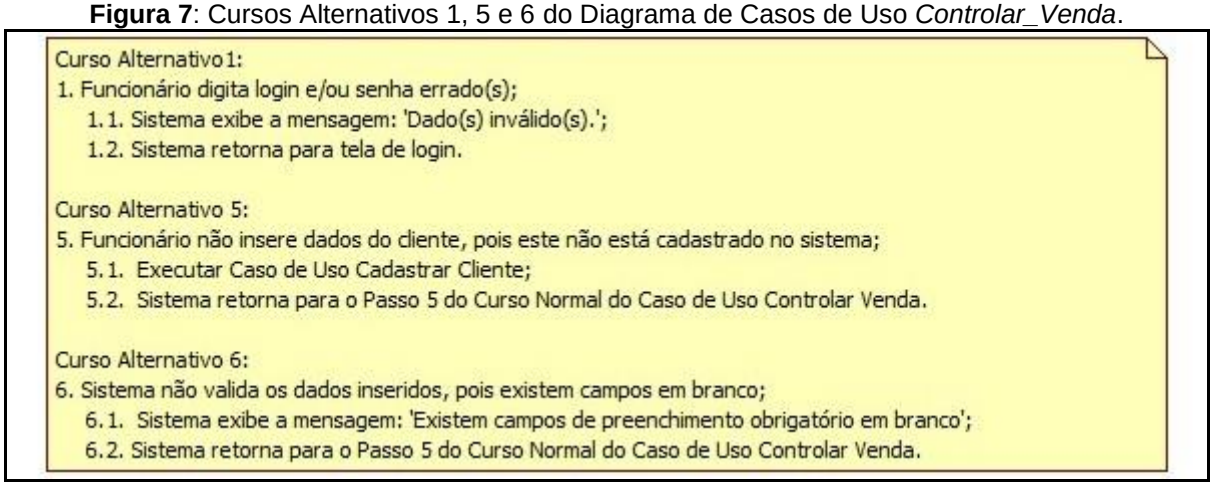
Figura 6: Diagrama de Casos de Uso *Controlar_Venda*.



Fonte: Elaborado pelos Autores.

Os cursos alternativos 1, 5 e 6 apresentam ações do ator *Funcionario* durante o registro da venda visando realizar, respectivamente, verificação nos passos de *login* e senha; redirecionamento de execução para o caso de uso *Cadastrar_Cliente*; e validação no preenchimento de campos obrigatórios. Esses roteiros alternativos estão representados na Figura 7.

Figura 7: Cursos Alternativos 1, 5 e 6 do Diagrama de Casos de Uso *Controlar_Venda*.



Fonte: Elaborado pelos Autores.

CONSIDERAÇÕES FINAIS

Essa pesquisa teve como objetivo realizar o desenvolvimento de um sistema a partir de métodos, técnicas e ferramentas de engenharia de *software*. As abordagens de engenharia de requisitos permitiram identificar e entender como deveria funcionar o futuro sistema de modo a atender às exigências do cliente e automatizar sob medida as rotinas da empresa analisada no projeto.

Foram utilizadas técnicas de modelagem orientada a objetos, especialmente, a linguagem de modelagem *UML* para elaborar diferentes tipos de diagramas que permitiram compreensão mais aprofundada dos eventos do sistema. Essa abordagem também contribuiu para a detecção de problemas no escopo do projeto, o reconhecimento de futuros desejos do cliente, a identificação de novos requisitos e a correção das informações de projeto conforme evoluções eram geradas.

Como trabalhos futuros serão realizadas as etapas seguintes de projeto, codificação e testes do sistema. Acredita-se que a adoção dessa abordagem disciplinada de desenvolvimento permitirá o alcance de um produto de qualidade a ser implantado no cliente. Nesse estágio haverá necessidade de capacitação dos

usuários finais da ferramenta e monitoramento no uso do sistema de modo a coletar reações das pessoas perante seu manuseio. Esse *feedback* será valioso para realizar correções e acrescentar novas características, se necessário.

REFERÊNCIAS

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 2ª Edição, Rio de Janeiro: Elsevier, 2006.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 2ª Edição, Rio de Janeiro: Elsevier, 2006.

CORDEIRO, J. C. M. **Gerenciando Projetos de Desenvolvimento de *Software* com PMI, RUP e UML**. 5ª Edição, Rio de Janeiro: Brasport, 2010.

ENGHOLM, J. R. **Engenharia de *Software* na Prática**. 3ª Edição, Rio de Janeiro, Novatec, 2010.

FAGUNDES, R. M. **Engenharia de Requisitos: Do Perfil do Analista de Requisitos ao Desenvolvimento de Requisitos com UML e RUP**. São Paulo: Atlas, 2011.

LAWRENCE, P. S. **Engenharia de *Software*: Teoria e Pratica.** 2ª Edição, São Paulo: Pearson Education, 2004.

PRESSMAN, S. R. **Engenharia de *Software***. 6ª Edição, São Paulo: McGraw-Hill, 2006.

RAMOS, R. A. **Treinamento Prático em UML**. São Paulo: Digerati, 2006.

SILVA FILHO, A. M. **Plano de Projeto: Um 'Mapa' Essencial à Gestão de Projetos de *Software***. Revista Engenharia de Software, vol.3, p.22-31, 2008.