REVISTA CONEXÃO ELETRÔNICA

APLICAÇÃO DA UML EM UM SISTEMA DE CONTROLE PATRIMONIAL

Anderson Lopes de Moraes
Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas – FITL/AEMS

José Vitor dos Santos
Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas – FITL/AEMS

Kaique Rodrigues Correia
Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas – FITL/AEMS

André Aparecido Leal de Almeida
Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas – FITL/AEMS

Alan Pinheiro de Souza
Mestre em Sistemas de Informação
Docente das Faculdades Integradas de Três Lagoas – FITL/AEMS

RESUMO
Este artigo apresenta partes da análise de um sistema de controle patrimonial que procura melhorar o sistema de informação de empresas que não possui um controle automatizado sobre seu patrimônio. O objetivo deste trabalho é apresentar como a utilização da modelagem orientada a objetos pode melhorar a qualidade do desenvolvimento de um programa, apresentando as funções do sistema antes de iniciar a etapa de codificação. Nesse contexto, alguns eventos e partes do *software*[1] foram modelados a partir de diagramas para estruturar e especificar as características principais do projeto.

PALAVRAS-CHAVE: Engenharia de sistemas; Análise de sistemas; Controle patrimonial.

INTRODUÇÃO

A necessidade de automatizar uma empresa é uma estratégia vantajosa e muito requisitada na atualidade. Com fins de melhorar a administração patrimonial das companhias percebeu-se a necessidade de um controle de prestação de serviços para saber se estes são de qualidade e se atendem as necessidades de uma organização. Então, surgiu o interesse do desenvolvimento de um Sistema de Controle Patrimonial, que tem como objetivo registrar manutenções e a frequência em que são realizadas, troca de equipamentos, saber qual empresa melhor atende

as ocorrências de manutenções, em qual setor está localizado o item em que foi realizado a manutenção, com a finalidade de diminuir os gastos desnecessários na organização e registrar informações patrimoniais da empresa.

Com a implantação desse sistema, o cliente aprimorará a sua organização patrimonial de forma segura, ágil e econômica. Sempre que ocorrer a necessidade de consertar ou trocar um item da organização, pode ser verificado no sistema quais foram os problemas anteriores, a qualidade de um item e a partir destas informações, tomar decisões que são mais adequadas para a situação (STAIR, 1999).

Na primeira seção do trabalho são apresentados os materiais e métodos que foram utilizados para o desenvolvimento do projeto. A segunda seção apresenta a etapa de análise do projeto apresentando a estrutura inicial do sistema. A terceira seção apresenta a etapa de projeto mostrando as principais funções contidas no programa. A quarta seção aborda parte da implementação do sistema, mostrando as relações entre os componentes do programa e o projeto de banco de dados.

1 MATERIAIS E MÉTODOS

O artigo apresenta técnicas de engenharia de *software* em um sistema de controle patrimonial, serão mostradas as principais partes de sua análise, projeto e implementação através de diagramas que utilizam a UML[2] para modelar o sistema e seus principais objetivos. Na análise será apresentado o modelo de *software* utilizado, a lista de eventos do sistema e os atores, o diagrama de classe e os principais casos de uso. Na parte de projeto será apresentado o diagrama de estado. A etapa de implementação apresentará os diagramas de componentes e o projeto de banco de dados.

As ferramentas que foram utilizadas para desenvolver o sistema são:

- **Microsoft Visual Studio 2012:** Ferramenta para implementação do sistema. A linguagem escolhida para a codificação foi o *CSharp* (C#).
- **Microsoft SQL Server 2012:** Banco de dados para armazenar os dados do sistema.

- **Microsoft SQL Server Management Studio 2012:** SGDB[3] Utilizado para gerenciar o banco de dados.
- **StarUML:** Utilizado para desenvolver a modelagem do sistema.

2 ANÁLISE

Nesta seção será apresentada a análise do sistema, mostrando o modelo de processo de software utilizado, atores e eventos contidos no programa, o diagrama de classes e os diagramas de casos de uso das principais funções do sistema. Para que o projeto diminua quantidade de falhas e retorne bons resultados finais, foi escolhido um modelo de processo de *software*[3] para que o produto final seja de qualidade e consiga suprir necessidades do cliente. Esses modelos são complexos para serem elaborados, pois englobam várias técnicas e o sistema será baseado através dos resultados obtidos neles. Quando um processo é bem elaborado, a qualidade retornada é positiva, por isso é importante que todos os participantes do projeto se esforcem, garantindo um sistema eficaz (SOMMERVILLE, 2003).

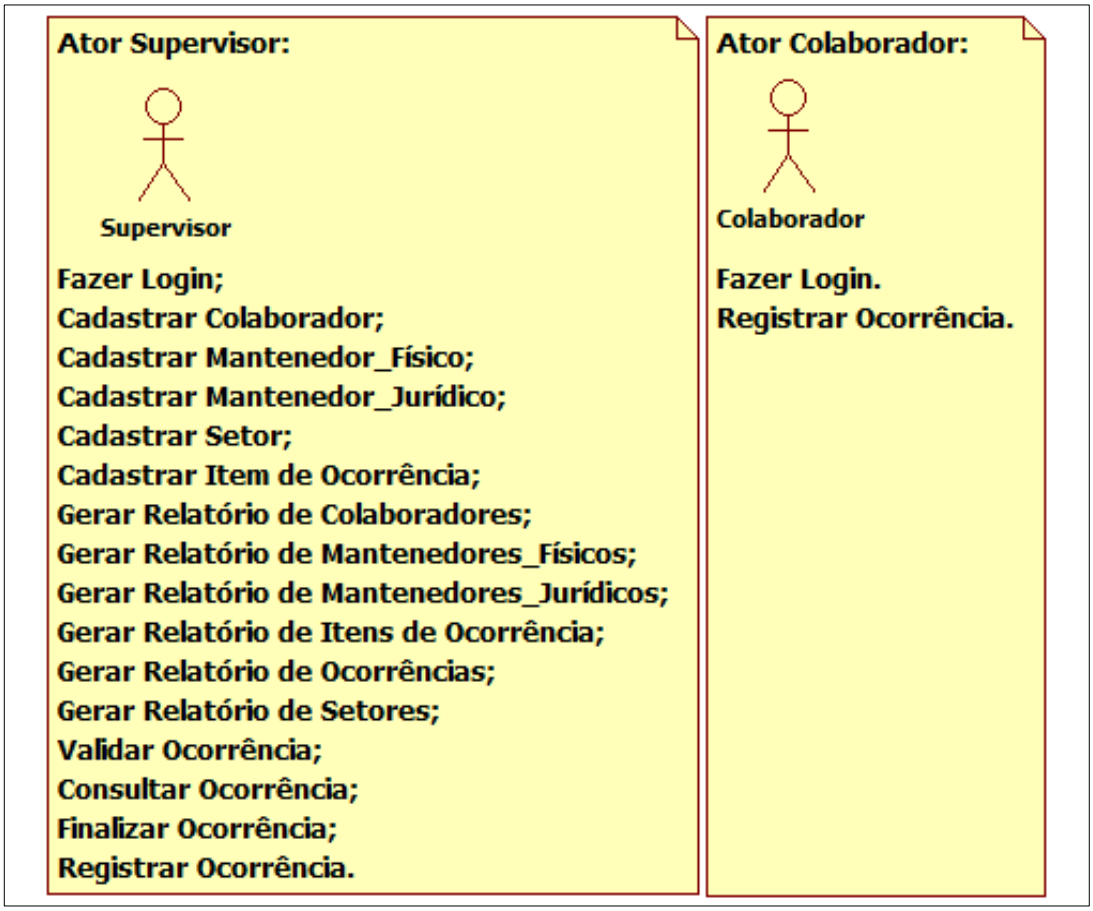
O desenvolvimento do sistema está utilizando o modelo de processo de *software* do tipo cascata com algumas adaptações. A Figura 1 mostra que as etapas do projeto são desenvolvidas sequencialmente e cada etapa inicia-se sempre após finalização da etapa anterior. Porém, algumas adaptações são agregadas a esse modelo, pois sempre que é necessário realizar uma alteração no projeto, ela é realizada em qualquer etapa e quando é finalizado o projeto continua em andamento (FILHO, 2009). Neste artigo serão apresentadas apenas as etapas de Análise de Requisitos, Projeto e Implementação.

Figura 1: Processo de software modelo cascata.

Fonte: Elaborado pelos autores (2015).

A Figura 2 apresenta os autores do sistema de acordo com os eventos em que possuem permissões para interagir. Há dois tipos de usuários que vão interagir com o sistema, Colaborador que apenas tem permissões para registrar as ocorrências e realizar o login e o Supervisor que possui todas as permissões de utilização do sistema, ele realizará cadastros de colaboradores, mantenedores físicos e jurídicos, setores, Itens de ocorrências e ocorrências. Será responsável por validar, finalizar e consultar ocorrências e também pela geração de relatórios de todos os cadastros realizados no sistema.

Figura 2: Autores e eventos do sistema.



Fonte: Elaborado pelos autores (2015).

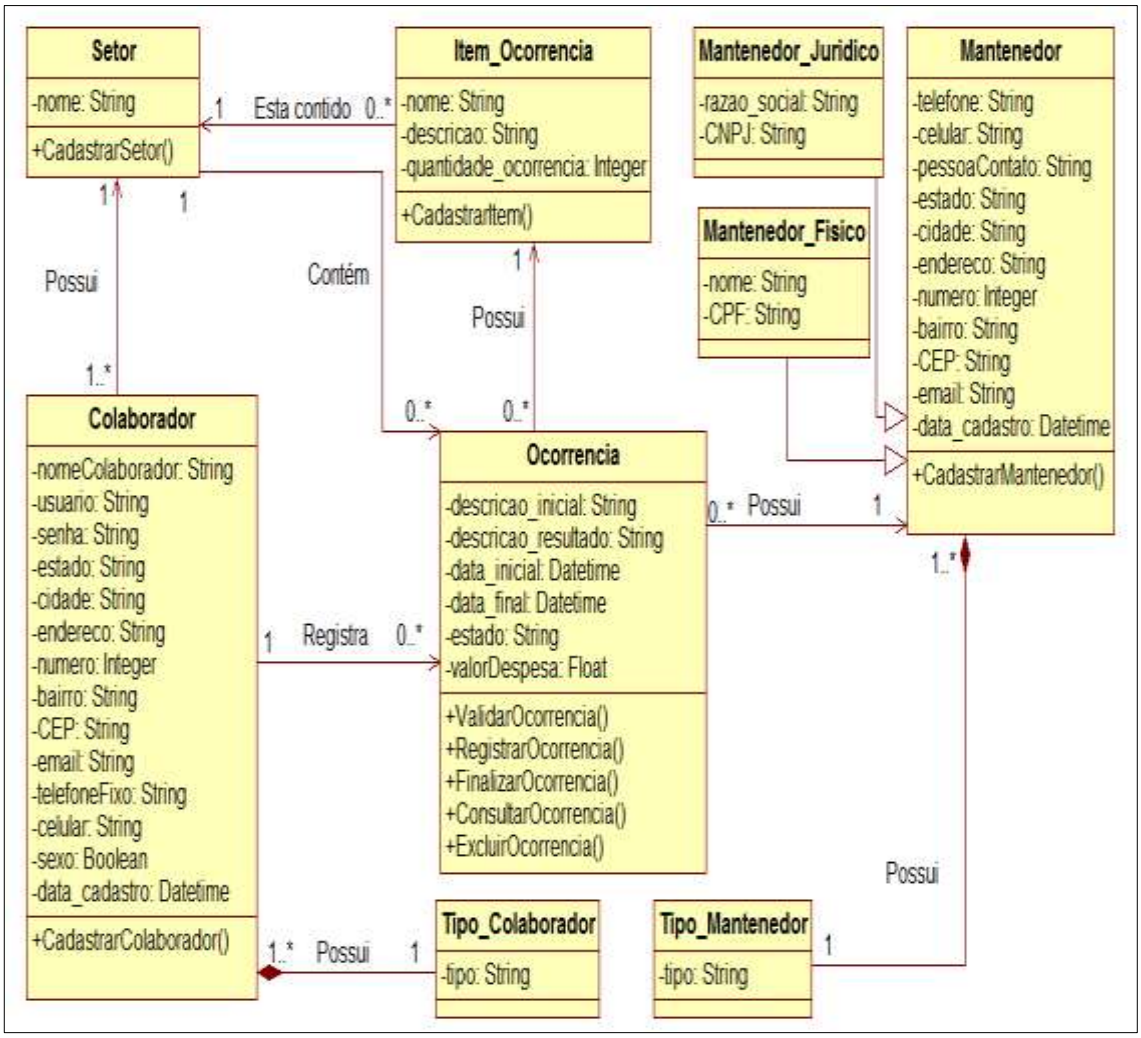
2.1 Diagrama de Classes

Para uma melhor visão estrutural do sistema é utilizado o diagrama de classes, que retrata as classes mostrando seus atributos, comportamentos e

relações entre as elas. Para a modelagem de alguns diagramas, é crucial que o diagrama de classes seja bem elaborado e possa servir de apoio (GUEDES, 2011).

A Figura 3 apresenta o diagrama de classes do Sistema de Controle Patrimonial contendo suas classes, atributos e métodos. As classes do sistema são:

Figura 3: Diagrama de classes do Sistema de Controle Patrimonial.



Fonte: Elaborado pelos autores (2015).

- Colaborador: registrará as ocorrências;
- Tipo_Colaborador: define o tipo de permissão que um colaborador irá receber quando for cadastrado no sistema, podendo ser permissão de supervisor ou de colaborador;
- Setor: cada colaborador e item tem seu setor;
- Item_Ocorrencia: itens contidos no setor;
- Ocorrencia: registra todos os detalhes das ocorrências;

- Mantenedor: realiza os serviços das ocorrências;
- Tipo_Mantenedor: define se o mantenedor é físico ou jurídico;
- Mantenedor_Juridico: essa classe herda os atributos da classe mantenedor e possui atributos próprios como CNPJ e razão social;
- Mantenedor_Fisico: essa classe herda os atributos da classe mantenedor e possui atributos próprios como CPF e nome.

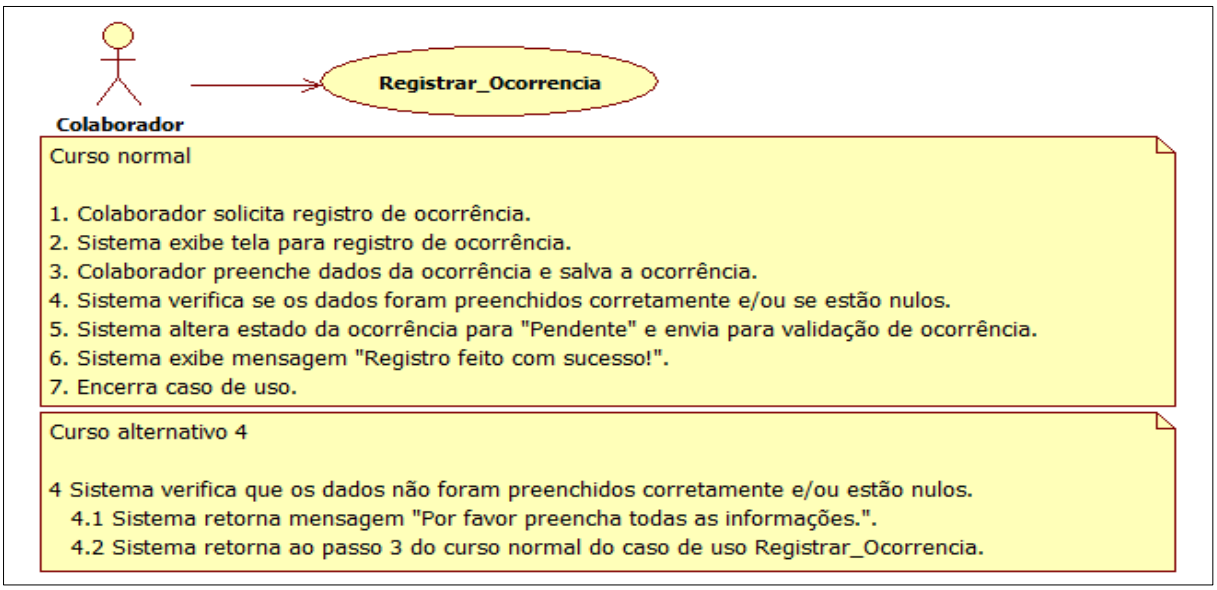
A classe Tipo_Colaborador está relacionada ao colaborador para definir se o colaborador cadastrado possui permissões de supervisor ou colaborador; O Setor é composto de colaboradores e itens; a Ocorrência deve possuir o Colaborador que a solicitou, o Setor em que ocorreu, o Item_Ocorrencia e o Mantenedor; Mantenedor possui seu tipo físico ou jurídico que é determinado na classe Tipo_Mantenedor.

2.2 Diagrama de Casos de Uso

Para melhor compreender o funcionamento do sistema serão apresentados os principais diagramas de casos de uso mostrando como será a execução normal e quais as exceções que poderão ocorrer durante o uso do produto final. Os casos de uso determinam quais funções que os usuários finais poderão utilizar e quais são as características presentes no sistema (PRESSMAN, 2011).

A Figura 4 mostra o caso de uso Registrar_Ocorrencia, o objetivo desta função é obter a descrição da ocorrência, data inicial do seu registro, o item que deve ser verificado e o seu setor, após registrada seu estado é alterado para “Pendente”.

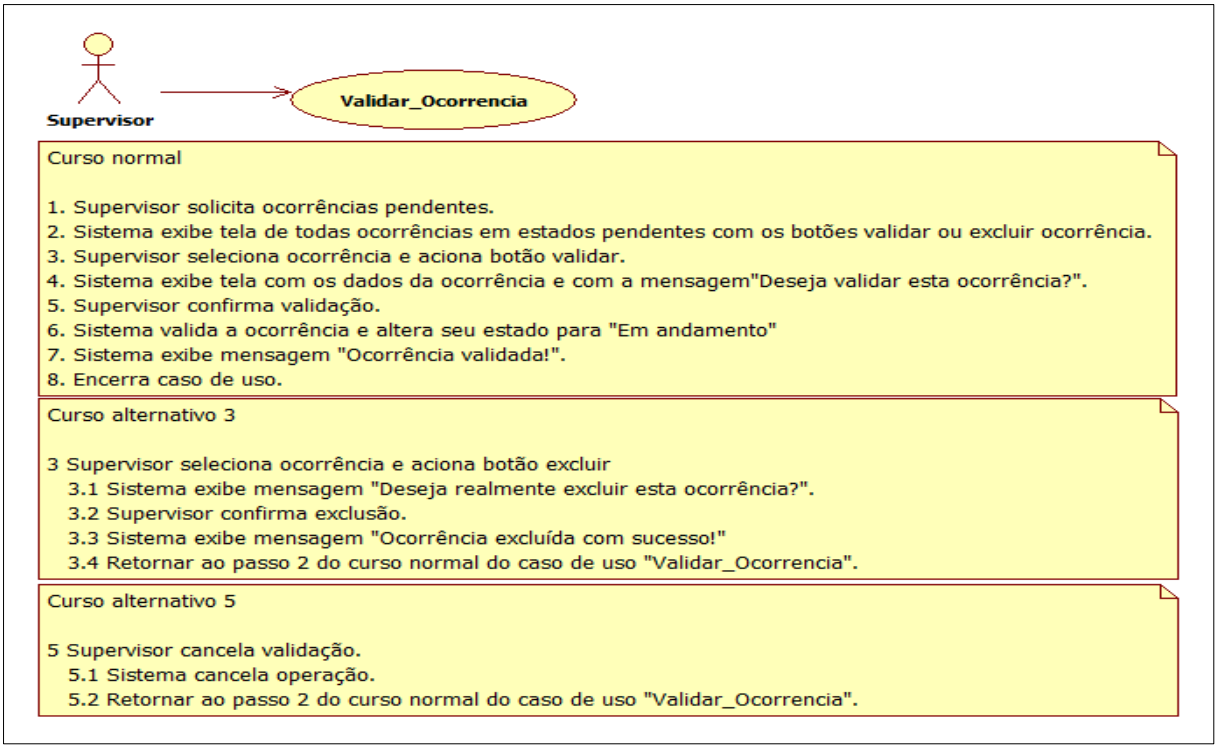
Figura 4: Diagrama de caso de uso Registrar_Ocorrencia.



Fonte: Elaborado pelos autores (2015).

A Figura 5 representa a validação da ocorrência com o caso de uso Validar_Ocorrencia, após verificado o problema se realmente existir o supervisor valida a ocorrência e é alterado seu estado para “Em andamento”.

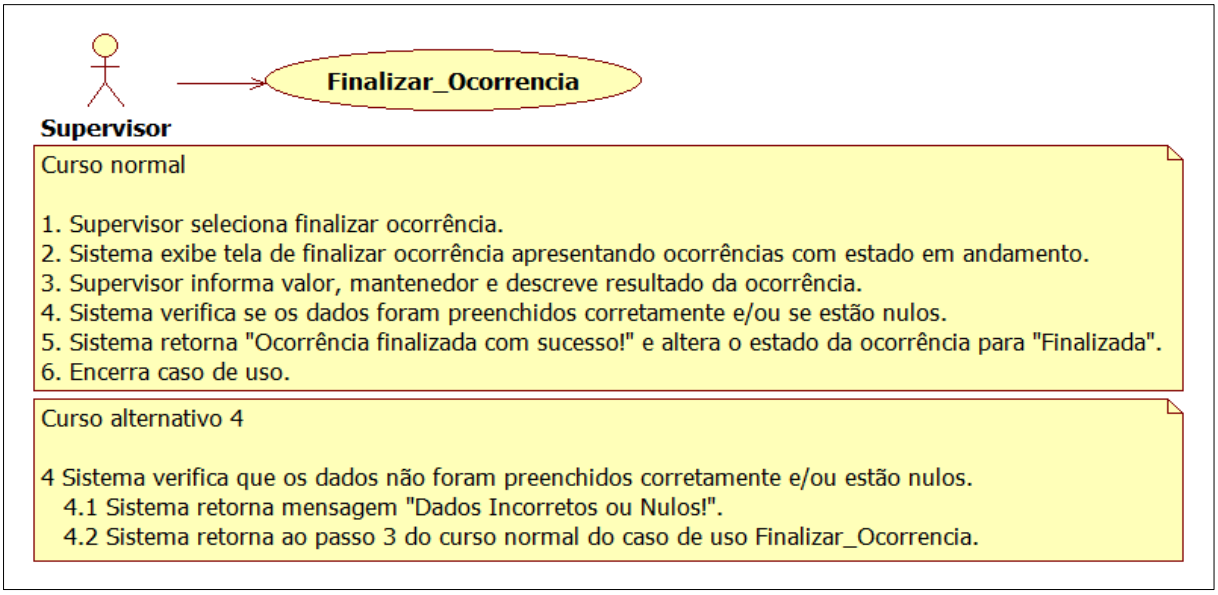
Figura 5: Diagrama de caso de uso Validar_Ocorrencia.



Fonte: Elaborado pelos autores (2015).

A Figura 6 apresenta o caso de uso Finalizar_Ocorrencia, pois é o momento em que o supervisor apresenta os dados finais que relatam o que foi feito na ocorrência e se gerou alguma despesa para a organização, após concluído seu estado se torna “Finalizada”.

Figura 6: Diagrama de caso de uso Finalizar_Ocorrencia.



Fonte: Elaborado pelos autores (2015).

3 PROJETO

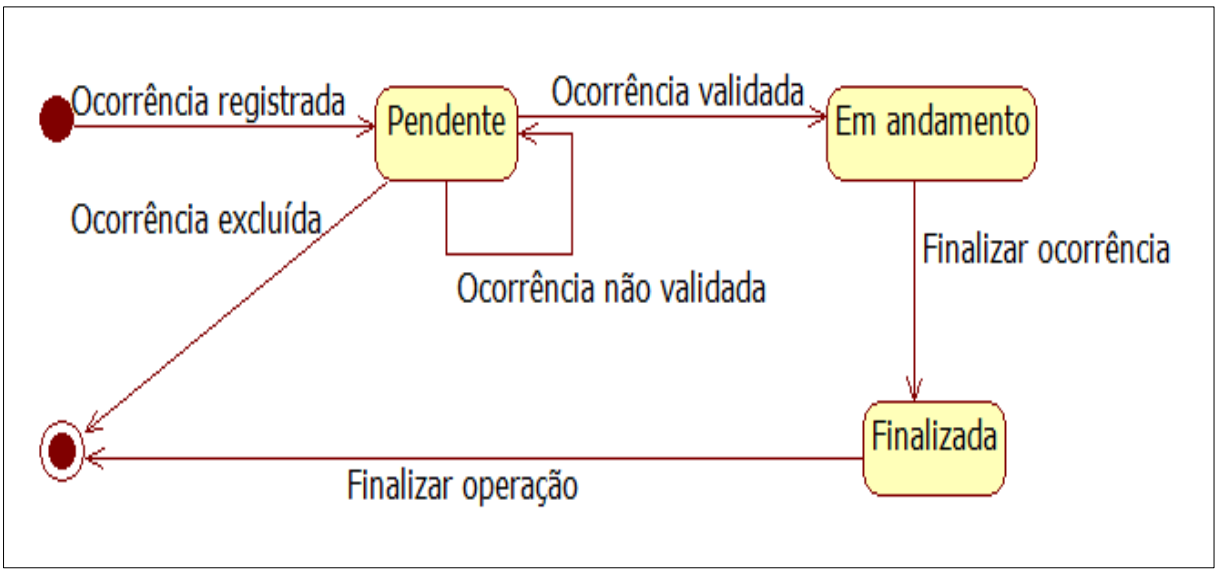
Nesta seção será apresentada partes da etapa de projeto do sistema, terá como foco apenas a apresentação do diagrama de estado que apresentará as mudanças de estado dos principais eventos do sistema. O projeto amplia e melhora a qualidade dos conjuntos de objetos e classes contidos no sistema, de modo que apresente como será representado ao usuário.

O diagrama de estado mostra como um artefato do sistema poderá se comportar em uma determinada situação e como pode muda-lo para um tipo de estado naquele momento da execução do sistema (PRESSMAN, 2011).

A Figura 7 apresenta o diagrama de estado das ocorrências, mostrando como é seu comportamento em relação aos eventos do sistema. No momento do registro da ocorrência seu estado se torna “Pendente”, caso for verificado que a ocorrência não é válida esse registro será descartado, se for válida a ocorrência

mudará seu estado para “Em andamento” e quando o processo da ocorrência for concluído será registrado os detalhes finais e o seu estado se tornará “Finalizada”.

Figura 7: Diagrama de estado de ocorrência.



Fonte: Elaborado pelos autores (2015).

4 IMPLEMENTAÇÃO

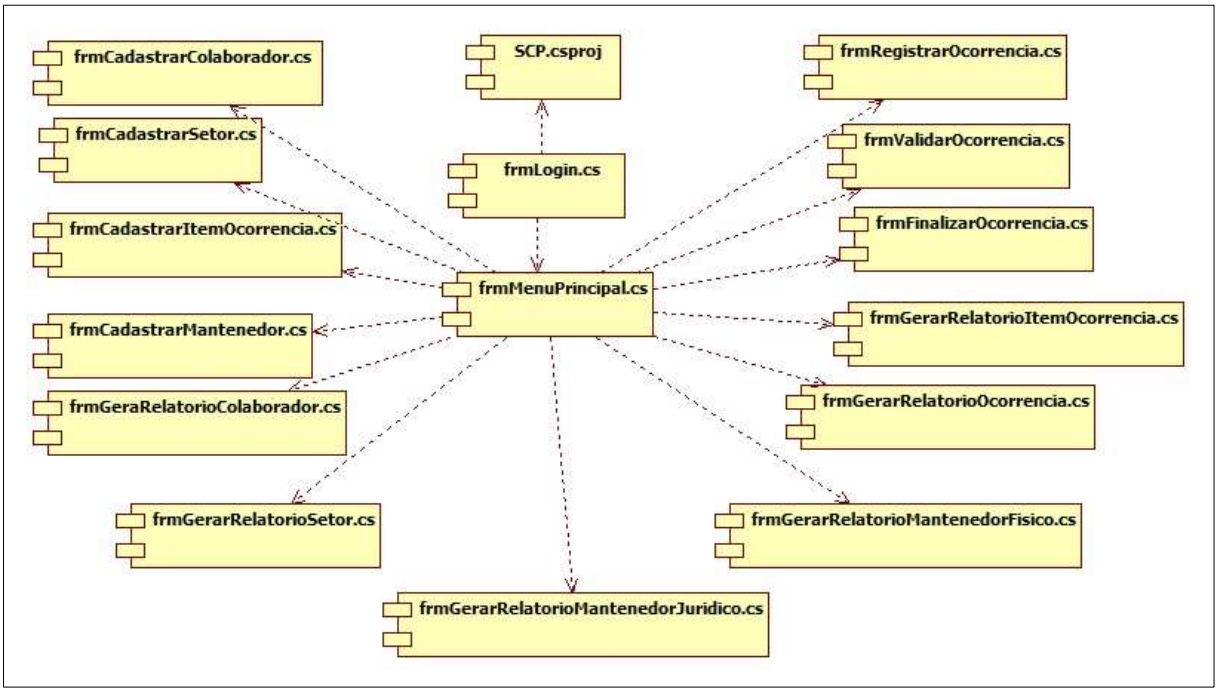
Nesta seção serão apresentados o diagrama de componente que mostra como cada componente está ligado um ao outro e o projeto de banco de dados que mostra quais os dados que estão contidos nas classes e quais serão manipulados pelos eventos do sistema.

4.1 Diagrama de Componentes

O diagrama de componentes relaciona e organiza os artefatos do sistema apresentando como estão interligados e como será a hierarquia dos objetos, facilitando o mapeamento das classes na implementação do sistema (RUMBAUGH; JACOBSON; BOOCH, 2004).

A Figura 8 apresenta o diagrama de componentes do Sistema de Controle Patrimonial mostrando a distribuição das relações dos componentes. Inicialmente, o sistema apresentará uma tela para identificação do usuário, após confirmado será exibido a tela principal do sistema.

Figura 8: Diagrama de componentes.



Fonte: Elaborado pelos autores (2015).

O componente “frmLogin.cs” representa a tela que realizará o login para entrar no sistema e acessar as demais funções, após o login o menu principal será exibido. O componente “frmMenuPrincipal.cs” é o menu principal do sistema que representa a tela que contém os menus de acesso as demais funções do sistema.

A partir do menu principal é possível realizar os cadastrados desejado pelo supervisor. O componente “frmCadastrarColaborador.cs” representa a tela que realizará os cadastros de colaboradores. O componente “frmCadastrarSetor.cs” representa a tela que realizará os cadastros de setores. O componente “frmCadastrarItem.cs” representa a tela que realizará os cadastros de itens. O componente “frmCadastrarMantenedor.cs” representa a tela que realizará os cadastros de mantenedores físicos e jurídicos.

Os eventos de ocorrências da mesma forma poderão ser acessados a partir da tela principal. O componente “frmRegistrarOcorrencia.cs” representa a tela que realizará os registros das ocorrências. O componente “frmValidarOcorrencia.cs” representa a tela que realizará a validação das ocorrências. O componente “frmFinalizarOcorrencia.cs” representa a tela que finalizará as ocorrências.

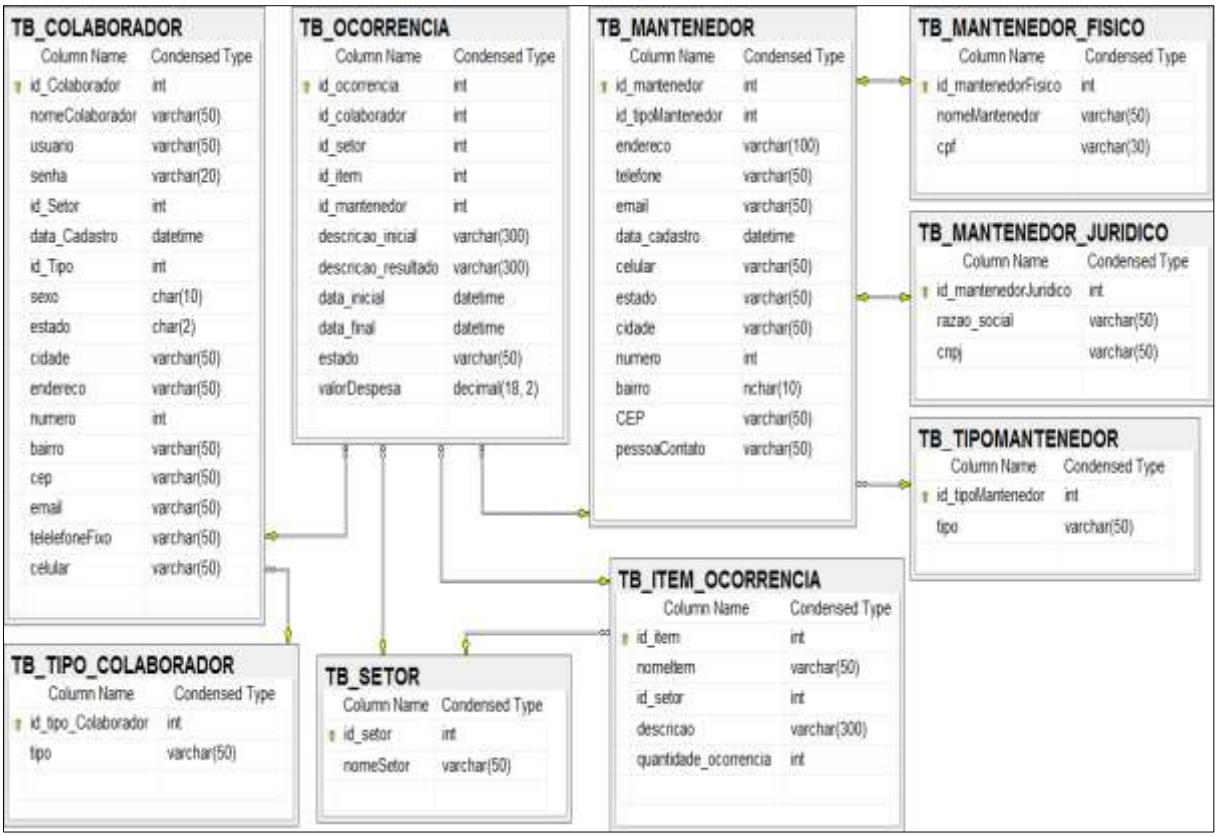
Os relatórios, assim como as outras funções do programa, são acessíveis a partir do componente de tela principal. O componente “frmRelatorioColaborador.cs” representa a tela de geração de relatórios dos colaboradores. O componente

"frmRelatorioSetor.cs" representa a tela de geração de relatórios dos setores. O componente "frmRelatorioMantenedorFisico.cs" representa a tela de geração de relatórios de mantenedores físicos. O componente "frmRelatorioMantenedorJuridico.cs" representa a tela que irá gerar os relatórios dos mantenedores jurídicos. O componente "frmRelatorioOcorrencia.cs" representa a tela que irá gerar os relatórios das ocorrências. O componente "frmRelatorioItem.cs" representa a tela que irá gerar os relatórios dos itens.

4.2 Projeto de Banco de Dados

Nesta seção será apresentado como o banco de dados do sistema foi estruturado e como estão seus relacionamentos entre suas tabelas, apresentando seu modelo lógico para que seja implementado em um SGDB (HEUSER, 1998). A Figura 9 mostra a ligação das tabelas do banco de dados, mostrando as chaves primárias e chaves estrangeiras.

Figura 9: Projeto de banco de dados.



Fonte: Elaborado pelos autores (2015).

A tabela “TB_COLABORADOR”, possui uma chave estrangeira com as tabelas “TB_TIPO_COLABORADOR” e “TB_SETOR”, pois o colaborador quando for cadastrado precisa indicar o tipo de permissão que ele possui e o setor em que está contido.

A tabela “TB_ITEM_OCORRENCIA”, possui uma chave estrangeira com a tabela “TB_SETOR”, pois o item será cadastrado em algum setor existente.

A tabela “TB_MANTENEDOR” possui uma chave estrangeira com a tabela “TB_TIPO_MANTENEDOR”, para indicar se o mantenedor é do tipo físico ou jurídico. As tabelas “TB_MANTENEDOR_FISICO” e “TB_MANTENEDOR_JURIDICO” recebem suas chaves primárias a partir da tabela “TB_MANTENEDOR”, quando é escolhido o tipo de mantenedor a tabela que é correspondente a aquele tipo receberá a chave primária da tabela “TB_MANTENEDOR” herdando também seus atributos.

A tabela “TB_OCORRENCIA”, possui quatro chave estrangeiras das tabelas “TB_COLABORADOR”, “TB_MANTENEDOR”, “TB_ITEM_OCORRENCIA”, “TB_SETOR”, pois quando for registrado uma ocorrência deverá ser informado todos esses relacionamentos para identificar a ocorrência.

CONSIDERAÇÕES FINAIS

A pesquisa mostra que a aplicação da UML pode melhorar a qualidade do desenvolvimento de um sistema, pois antes de chegar aos desenvolvedores, o sistema já está modelado com o objetivo de mostrar ao programador todas suas funções para que ele se preocupe apenas em como implementar através de códigos e dar funcionamento ao *software*. O artigo apresenta apenas algumas etapas e diagramas que podem ser utilizados para modelar e documentar um sistema, há outros diagramas que também auxiliam na modelagem como, diagrama de sequências, atividades, implantação, entre outros. Para trabalhos futuros, podem ser apresentados esses outros diagramas, gerando modelagem para todas as funções do sistema, além das etapas seguintes de desenvolvimento destacadas no modelo de processo, por exemplo, teste e manutenção.

NOTAS

[1] *Software*: Um programa, conjunto de instruções interpretadas por um computador com o objetivo de solucionar problemas específicos (PRESSMAN, 2011).

[2] UML: Linguagem de modelagem que auxilia na construção dos objetos contidos no sistema, com a finalidade de especificar e documentar esses artefatos de forma que os interessados no projeto possam compreender os objetivos do *software* (MATOS, 2002).

[3] SGDB: Sistema de Gerenciamento de Banco de Dados, permite organização, manipulação e acesso aos dados do repositório (HEUSER, 1998).

REFERÊNCIAS

FILHO, W. P. P. **Engenharia de Software:** fundamentos, métodos e padrões. 3ª Ed., Rio de Janeiro: LTC, 2009.

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

HEUSER, C. A. **Projeto de Banco de Dados.** 6ª ed. Porto Alegre: Bookman, 2008.

MATOS, A. V. **UML Prático e Descomplicado.** São Paulo: Érica, 2002.

PRESSMAN, R. S. **Engenharia de Software:** uma abordagem profissional. 7ª ed. Porto Alegre: Afiliada, 2011.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. ***The Unified Modeling Language Reference Manual.*** 2ª ed. *Boston: Pearson Education,* 2004.

SOMMERVILLE, I. **Engenharia de *Software.*** 6ª ed. São Paulo: Pearson Education, 2003.

STAIR, M. R. **Princípios de Sistema de Informação:** uma abordagem gerencial. 2ª ed. Rio de Janeiro: LTC, 1999.