UTILIZANDO ICONIX NO DESENVOLVIMENTO DE APLICAÇÕES DELPHI

Dr. George SILVA; Dr. Gilbert SILVA; Gabriel GUIMARÃES; Rodrigo MEDEIROS; Tiago ROSSINI;

Centro Federal de Educação Tecnológica do Rio Grande do Norte – CEFET-RN, Av. Sen. Salgado Filho, 1559, Tirol, Natal-RN, Fone/Fax: (84) 4005-2600 / 4005-2637, e-mail: george@cefetrn.br, gilbert@cefetrn.br

RESUMO

Este trabalho de pesquisa apresenta um estudo sobre a utilização da metodologia ICONIX, desenvolvida pela ICONIX *Software Engineering*, na construção de aplicações com o ambiente integrado de desenvolvimento *Borland Delphi*. O objetivo principal do artigo é expor os benefícios da associação deste processo de desenvolvimento de software, reconhecido pelos profissionais da área como um processo simples e ágil, com a funcionalidade e robustez da ferramenta *Borland Delphi*. Através de um estudo de caso - um Sistema de Apoio à Análise de Projetos de Combate a Incêndio, atualmente em implementação - serão abordados os conceitos e mecanismos do ICONIX, passando por todas as suas fases, apresentando os diagramas UML básicos utilizados no processo e relacionando seus artefatos com a estrutura organizacional de uma aplicação Delphi. Espera-se com este trabalho evidenciar as vantagens de adotar uma metodologia simples, mas poderosa, aliada a um ambiente de desenvolvimento que permite produzir aplicações completas de forma eficiente, minimizando o esforço de trabalho, otimizando o desenvolvimento e melhorando a qualidade do produto final.

Palavras-Chave: ICONIX, Delphi, Desenvolvimento de Software, UML

1. INTRODUÇÃO

As empresas procuram cada vez mais a informatização de seus processos visando o aumento da produtividade, a otimização de custos e a melhoria na qualidade de serviços e produtos. A informatização tem gerado a necessidade de processamento de grandes quantidades de informações com rapidez e segurança, tornando os softwares cada vez mais complexos. Com isso, é importante estabelecer estratégias eficientes de planejamento e produção de softwares. Isso pode ser feito através de um processo de desenvolvimento de software.

O processo de desenvolvimento de software é um conjunto de atividades e resultados associados que produzem um produto de software (SOMMERVILLE, 1995). São etapas cuidadosamente planejadas onde o sucesso de cada etapa é fundamental para um bom resultado final do produto.

Atualmente, existem metodologias que auxiliam o desenvolvedor a construir softwares de qualidade com rapidez e baixo custo. Cada metodologia se melhor adapta a um determinado tipo de software. Nesse trabalho será apresentada a metodologia ICONIX, associada ao ambiente de desenvolvimento Delphi.

No decorrer do artigo, através do estudo de caso "Um Sistema de Apoio à Análise de Projetos de Combate a Incêndio", abordaremos os conceitos e mecanismos do ICONIX, passando por todas as suas fases, apresentando os diagramas UML básicos utilizados no processo e relacionando seus artefatos com a estrutura organizacional de uma aplicação desenvolvida com o ambiente Delphi.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Processo ICONIX

O ICONIX é um processo de desenvolvimento de software desenvolvido pela ICONIX *Software Engineering*. Trata-se de uma metodologia prática e simples, mas também poderosa e com um componente de análise e representação de problemas sólido e eficaz.

É um processo não tão burocrático quanto o RUP (*Rational Unified Process*), ou seja, não gera tanta documentação, nem tão simples como o XP (*eXtreme Programing*), mas não deixa a desejar na análise e se destaca como um eficiente processo de análise de software (MAIA, 2005).

O ICONIX é um processo que está adaptado ao padrão da UML (OMG®, 2001), possuindo uma característica exclusiva chamada *Traceability of Requiriments* (Rastreabilidade dos Requisitos), que através de seus mecanismos, permite checar em todas as fases se os requisitos estão sendo atendidos (MAIA, 2005).

A abordagem do ICONIX permite que sejam usados outros recursos da UML para complementar os recursos usados nas fases do ICONIX, caso seja necessário.

Os principais diagramas do processo ICONIX são os seguintes:

- Modelo de Domínio
- Modelo de Caso de Uso
- Diagrama de Robustez
- Diagrama de Seqüência
- Diagrama de Classe

Segundo MAIA (2005), o ICONIX é dividido em dois grandes setores, modelo estático e modelo dinâmico, que podem ser desenvolvidos paralelamente e de forma recursiva (ver Figura 1). O modelo estático é formado pelos Diagramas de Domínio e Diagramas de Classe que modelam o funcionamento do sistema sem nenhum dinamismo e interação com o usuário. O modelo dinâmico, por sua vez, sempre mostra a interação entre o usuário e o sistema, através de ações onde o sistema apresenta alguma resposta ao usuário em tempo de execução. O modelo estático é refinado incrementalmente durante iterações sucessivas do modelo dinâmico.

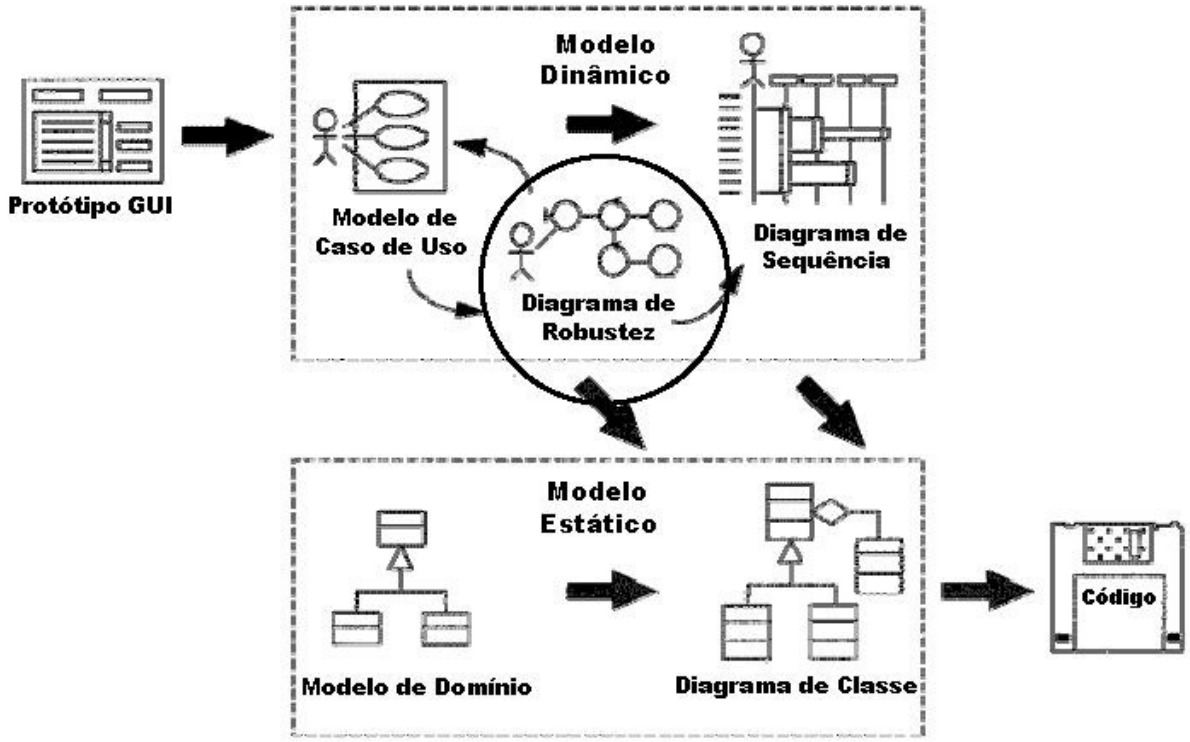

Figura 1 – Visão macro do ICONIX

O processo ICONIX trabalha a partir de um protótipo de interface onde se desenvolvem os diagramas de caso de uso baseados nos requisitos levantados. A partir dos diagramas de caso de uso, se faz a análise robusta para cada caso de uso. Com os resultados obtidos é possível desenvolver o diagrama de seqüência e, posteriormente, complementar o modelo de domínio com novos métodos e atributos descobertos (MAIA, 2005).

O modelo de análise robusta tem por finalidade conectar a parte de análise com a parte de projeto, assegurando que a descrição dos casos de uso esteja correta. Nesta etapa, podem-se descobrir novas classes, que não foram vistas no modelo de domínio. Na análise robusta, os objetos são classificados em três tipos (ver Figura 2): objetos limite ou interface (*Boundary Objects*), objetos controladores (*Control Objects*) e objetos entidade (*Entity Objects*).

Figura 2 – Representação dos objetos interface, controlador e entidade

Objetos interface são usados pelos usuários, denominados atores, para se comunicarem com o sistema. Objetos controladores controlam a lógica de negócio, fazendo a conexão entre objetos interface e objetos entidade. Os objetos entidade são responsáveis por realizar algum tipo de persistência.

2.2. Ambiente de Desenvolvimento Delphi

O Delphi é um Ambiente de Desenvolvimento Integrado (IDE) para o desenvolvimento de softwares. Ele é produzido pela *Borland Software Corporation.*

A linguagem utilizada pelo Delphi é *Object Pascal* (Pascal com extensões orientadas a objetos), que a partir da versão 7 passou a se chamar *Delphi Language*.

O Delphi fornece, em um só programa, a linguagem, a IDE e sua biblioteca de componentes (VCL/CLX), o que contribui para uma boa consistência interna e um pacote mais reconhecível. Alguns destacam como vantagens do Delphi: a existência de uma grande quantidade de componentes prontos em sua biblioteca, facilidade de uso e aprendizado e desenvolvimento rápido de aplicações. Essas características fazem do Delphi uma excelente ferramenta para densenvolvimento de sistemas utilizando a metodologia ICONIX.

Segundo Oliviero (2003), o Delphi é uma ferramenta voltada para a criação de aplicações (*Front-End*) e manipulação do banco de dados (*Back-End*). Para que um banco de dados possa ser ligado a uma aplicação desenvolvida em Delphi, é necessária a utilização de uma camada intermediária (*Middle-End*) que permita fazer com que a aplicação converse com o banco de dados. Essa camada intermediária contém o motor (*engine*) de acesso a dados (ver Figura 3). O Delphi contém uma série dessas *engines*, desenhadas para atender aos mais variados tipos de acesso aos bancos de dados disponíveis no mercado.

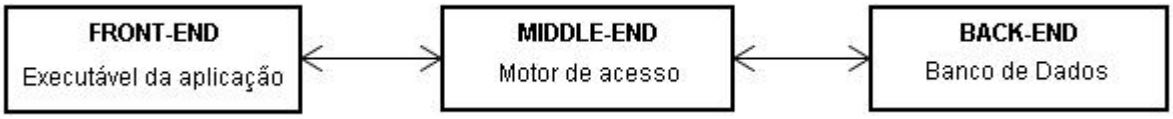

Figura 3 – Arquitetura de acesso a dados

A versão 5 do Delphi introduziu o padrão de conexão IBX (*InterBase eXpress*), que possui um conjunto de componentes especialmente desenvolvido para dar acesso nativo ao banco de dados InterBase. Tais componentes não utilizam nenhuma *engine* de acesso a dados, sendo este acesso feito diretamente através da API do InterBase (OLIVIERO, 2003).

Existem vários componentes no Delphi para representar elementos de banco de dados e possibilitar a conexão entre o aplicativo e o banco. Estes componentes podem ser reunidos em módulos denominados *Data Modules*. Ainda de acordo com Oliviero (2003), o *Data Module* é como um depósito usado para centralizar os componentes "não visuais" necessários para administrar um banco de dados.

3. ASSOCIANDO O PROCESSO ICONIX AO DELPHI

3.1. Mapeamento entre os componentes do Delphi e os diagramas do ICONIX

Além dos diagramas de caso de uso e de classes, a associação entre o processo ICONIX e o ambiente Delphi pode ser fortalecida através do diagrama de robustez. Os objetos utilizados neste diagrama (ver Figura 2) podem ser associados diretamente a componentes do ambiente de desenvolvimento Delphi, tornando mais objetiva a realização do projeto do software para equipe de desenvolvimento a partir de tais diagramas.

Um caso de uso simples, como um cadastro, pode ser implementado no Delphi com apenas uma tela (*Form*) e um *Data Module*. No Sistema de Apoio à Análise de Projetos de Combate a Incêndio, por exemplo, cada projeto tem um imóvel, que deve ser cadastrado no sistema quando é dada entrada em um projeto.

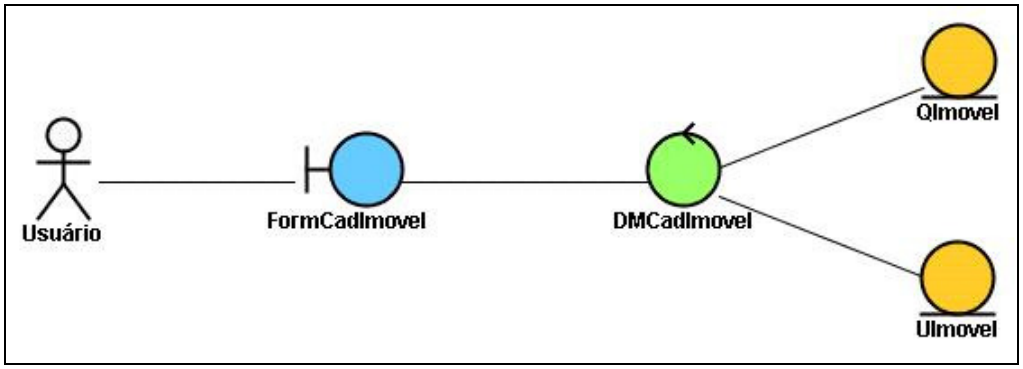

Figura 4 – Esboço do diagrama de robustez para o caso de uso "Cadastro de Imóvel"

O esboço do diagrama de robustez do caso de uso "Cadastro de Imóvel" (ver Figura 4) mostra o formulário de cadastro FormCadImovel como sendo o objeto interface, que possibilita a comunicação entre o usuário e o sistema. O *Data Module* DMCadImovel representa o objeto controlador, que faz a conexão entre o objeto interface e o objeto entidade. Por fim, os componentes QImovel e UImovel correspondem aos objetos entidade, que permitem a manipulação das tabelas no banco de dados.

O objeto QImovel (ver Figura 5), do tipo *TIBQuery*, é um *DataSet*, ou seja, ele carrega os dados da tabela para serem exibidos no formulário. O objeto UImovel, do tipo *TIBUpdateSQL*, contém o código SQL para inserir, atualizar e remover registros da tabela. E o objeto DSImovel serve como canal entre o *DataSet* e os componentes de manipulação e visualização de dados do formulário.

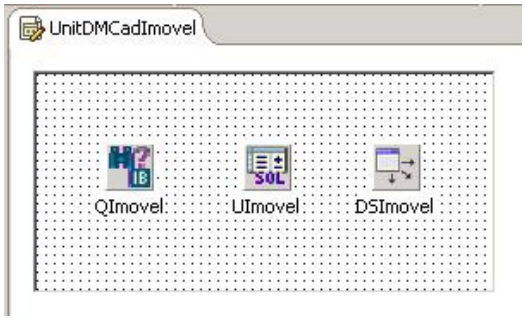

Figura 5 – Representação visual do Data Module DMCadImovel

3.2. Etapas do processo de desenvolvimento ICONIX

No ICONIX, o desenvolvimento começa a partir de um protótipo de interface validado pelo usuário de acordo com os requisitos do sistema. O uso dessa interface possibilita ter maiores garantias de que os requisitos estão sendo atendidos, além de dar uma visão maior do sistema e, conseqüentemente, maior controle (MAIA, 2005).

Para ilustrar as etapas da metodologia ICONIX, usaremos o estudo de caso do Sistema de Apoio à Análise de Projetos de Combate a Incêndio, como já foi dito anteriormente. Apresentaremos a parte de cadastro de projetos para análise, exibindo os diagramas e componentes do processo ICONIX e explicando, sempre que necessário, como implementá-los no Delphi.

O protótipo de interface para o cadastro de projetos pode ser visto na Figura 6, abaixo:

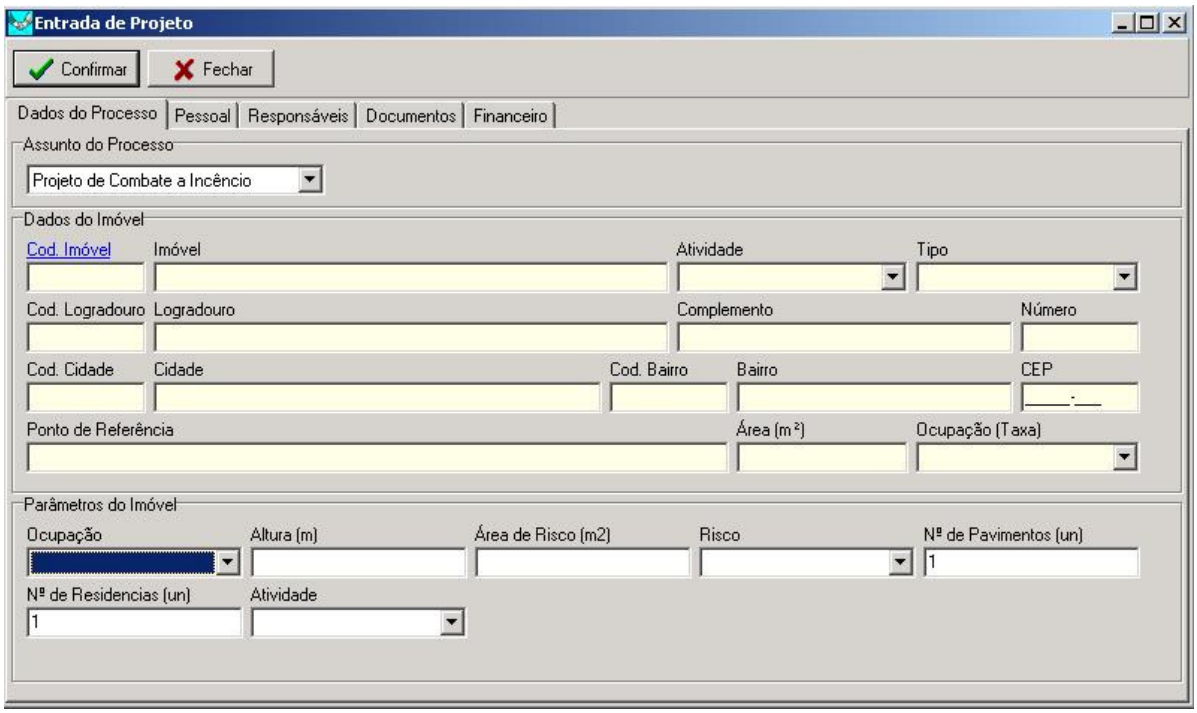

Figura 6 – Protótipo de interface do cadastro de projeto desenvolvido no Delphi

3.3. Modelo de Domínio

A partir dos requisitos do sistema, criamos o modelo de domínio, uma parte essencial do processo. Basicamente, ele consiste em achar classes, substantivos, frases de substantivos, verbos e frases de verbos, que se tornarão objetos, atributos e associações em um diagrama de domínio.

No sistema proposto, um dos requisitos é o cadastro de projetos, no qual projetos são cadastrados para ser analisados quanto à conformidade com a Norma Estadual de Combate a Incêndio. Um projeto é um processo que tramita pelo sistema. Ele tem, dentre outras características, um imóvel, um responsável, uma lista de documentos e vários parâmetros de projeto. Alguns desses parâmetros de projeto possuem faixas de valores, que definem os equipamentos de segurança e de combate a incêndio necessários para aquele imóvel. Outros parâmetros têm taxas associadas, que definem o valor do projeto. O valor do projeto pode ser dividido em parcelas.

De posse destas informações, é possível criar um modelo de domínio, como o da Figura 7:

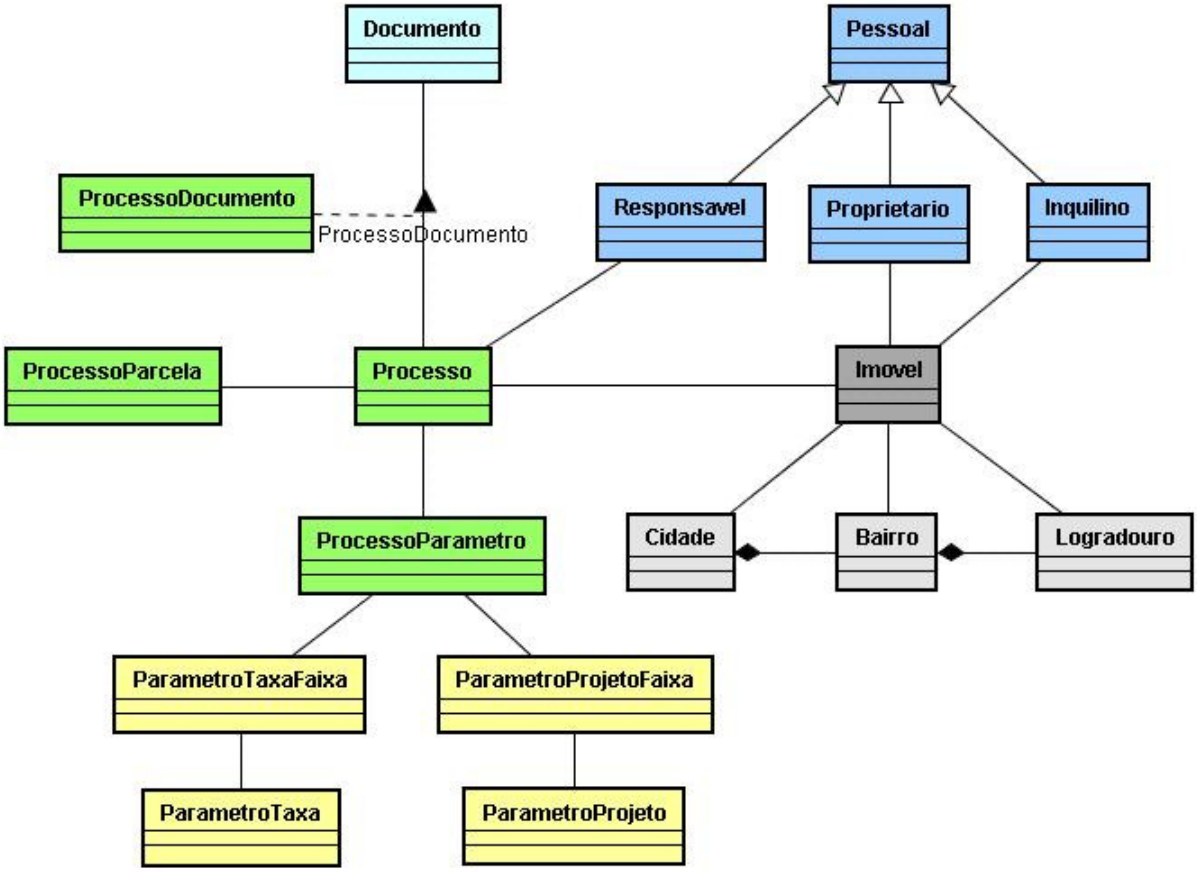

Figura 7 – Modelo de domínio: Cadastrar Projeto

É importante frisar que na construção do modelo de domínio, não é levada em consideração a multiplicidade das associações.

3.4. Modelo de Caso de Uso

Este modelo é usado para representar a funcionalidade do sistema destacando os seus usuários. Ele deve detalhar de forma clara todos os cenários que os usuários atuarão para realizar alguma tarefa (MAIA, 2005).

Na descrição do caso de uso, deve-se ter o curso normal do caso de uso e todos os possíveis cursos alternativos. A descrição do caso de uso ‘Cadastrar Projeto’ encontra-se a seguir:

Curso Básico:

O usuário entra com as informações necessárias do projeto no sistema.
O sistema valida o cadastro do projeto.
O sistema calcula o valor do projeto e gera as parcelas.
O sistema armazena os dados do projeto.

Curso Alternativo:
Se os dados estiverem incompletos o cadastro é bloqueado.
O sistema exibe uma mensagem para que o usuário forneça as informações completas.

O diagrama de caso de uso gerado a partir da descrição pode ser visto na Figura 8, abaixo:

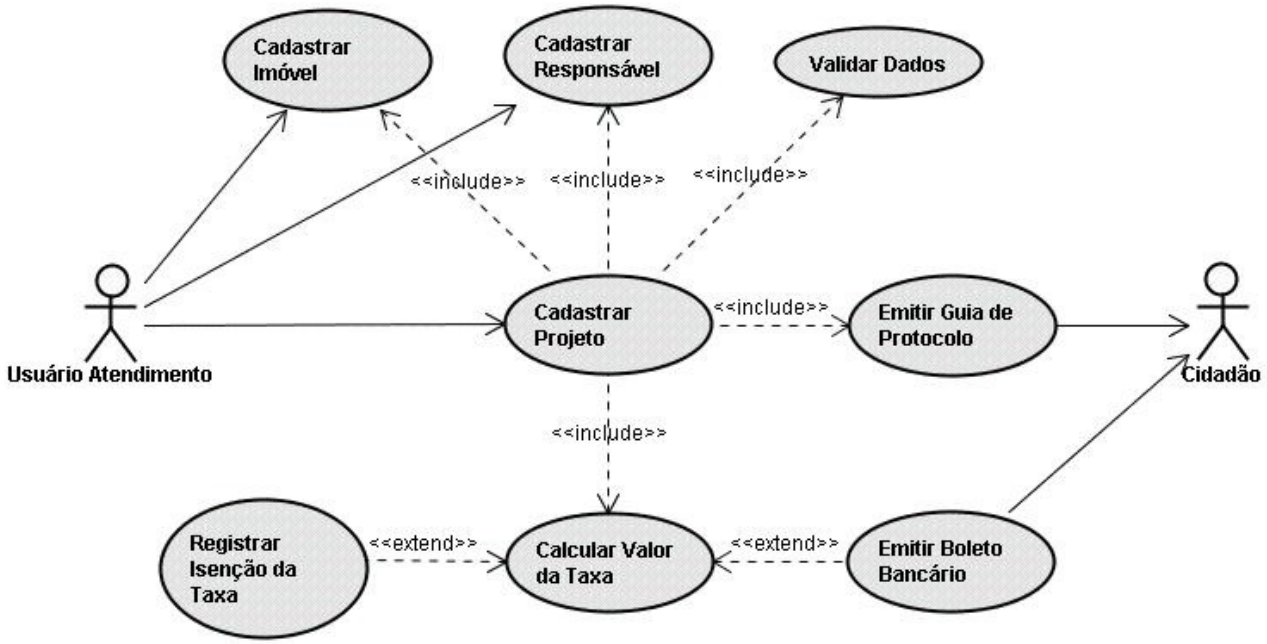

Figura 8 – Diagrama de caso de uso: Cadastrar Projeto

3.5. Diagrama de Robustez

Conforme citado, o diagrama de rodustez tem por finalidade conectar a parte de análise com a parte de projeto, assegurando que a descrição dos casos de uso esteja correta e, dentro da idéia proposta, realizando o mapeamento dos objetos do diagrama para os componentes do Delphi.

O modelo de análise robusta para o caso de uso proposto encontra-se na Figura 9, a seguir. No diagrama, objetos interface representam formulários do Delphi (FormCadProjeto, FormConsImovel, FormConsPessoal). Os objetos entidade representam componentes TIBQuery e TIBUpdateSQL (QProcesso, UProcesso, QImovel, dentre outros). Já os objetos controladores podem representar Data Modules (DMTProcesso, DMConsImovel, DMLkpAnalise, dentre outros), funções de chamada a outros objetos (localizarImovel, localizarResponsavel), ou classes de controle (TProcesso, TProcessoDocumento, dentre outras).

As classes de controle mapeiam entidades do sistema. Cada classe tem um Data Module próprio. Essas classes estão representadas no diagrama como objetos controladores porque não interagem diretamente com as tabelas do banco e são acessadas pelos formulários. Sua função é reunir os métodos necessários para criar, alterar e remover registros do banco. Uma das vantagens de utilizar essas classes é que assim torna-se possível instanciar vários objetos e colocá-los dentro de uma coleção, além de poder modificá-los sem necessariamente alterar os dados no banco.

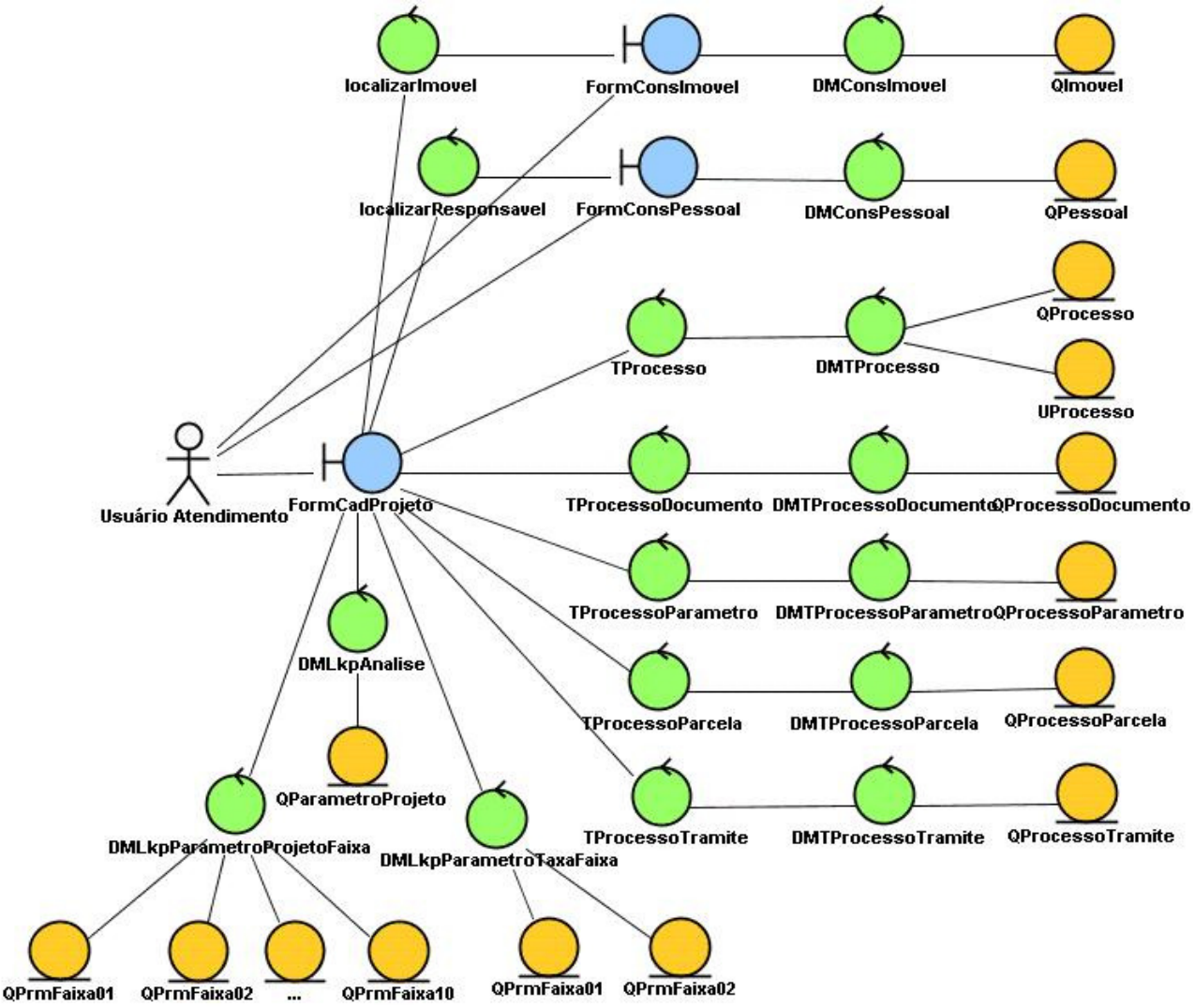

Figura 9 – Modelo de análise robusta: Cadastrar Projeto

3.6. Diagrama de Seqüência

Depois de criado o modelo de análise robusta, é hora de projetar como o software realmente irá funcionar. O Diagrama de Seqüência é então realizado com o objetivo é obter um modelo dinâmico entre o usuário e o sistema. Para isso, devemos utilizar os objetos e suas interações identificadas na análise robusta, só que agora, é detalhado o fluxo de cada ação (MAIA, 2005).

O diagrama de seqüência tem como papel principal mostrar o funcionamento do sistema em tempo de execução. Ele mostra a troca de mensagens entre os objetos, de acordo com a descrição do caso de uso.

O diagrama de seqüência parcial do caso de uso proposto (ver Figura 10) mostra a rotina de cadastro de um projeto, focando na consulta de imóvel e nos parâmetros de projeto. Os demais objetos e associações foram omitidos para não complicar a visualização do diagrama.

De acordo com a descrição do caso de uso e os modelos elaborados até agora, quando o formulário é aberto, os parâmetros e suas faixas são carregados na tela. O usuário faz uma consulta de imóvel para trazer o imóvel (já cadastrado) daquele projeto, e preenche os campos de parâmetros de projeto com as faixas apropriadas. Quando o usuário confirma o cadastro, uma rotina valida os dados e cria o processo no banco. Basicamente, esse é o roteiro de implementação do cadastro de projeto.

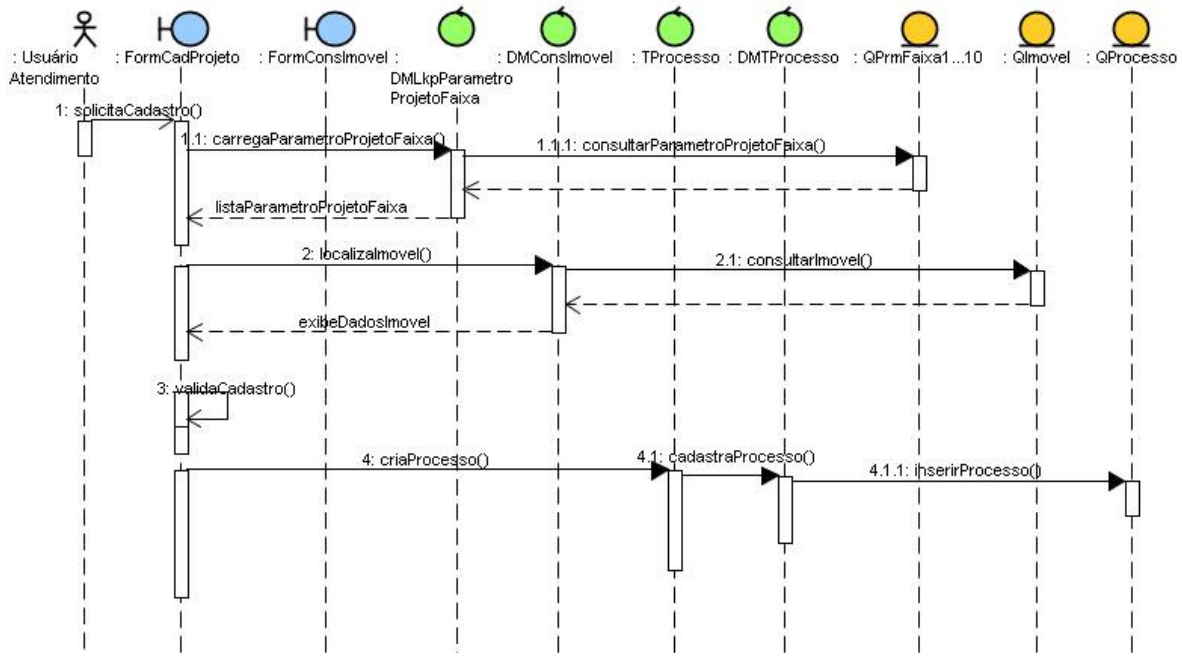

Figura 10 – Esboço do Diagrama de Seqüência: Cadastrar Projeto

3.7. Diagrama de Classes

Finalmente é construído o diagrama de classe a partir de uma evolução do modelo de domínio que foi atualizado ao longo das fases do ICONIX (ver Figura 11).

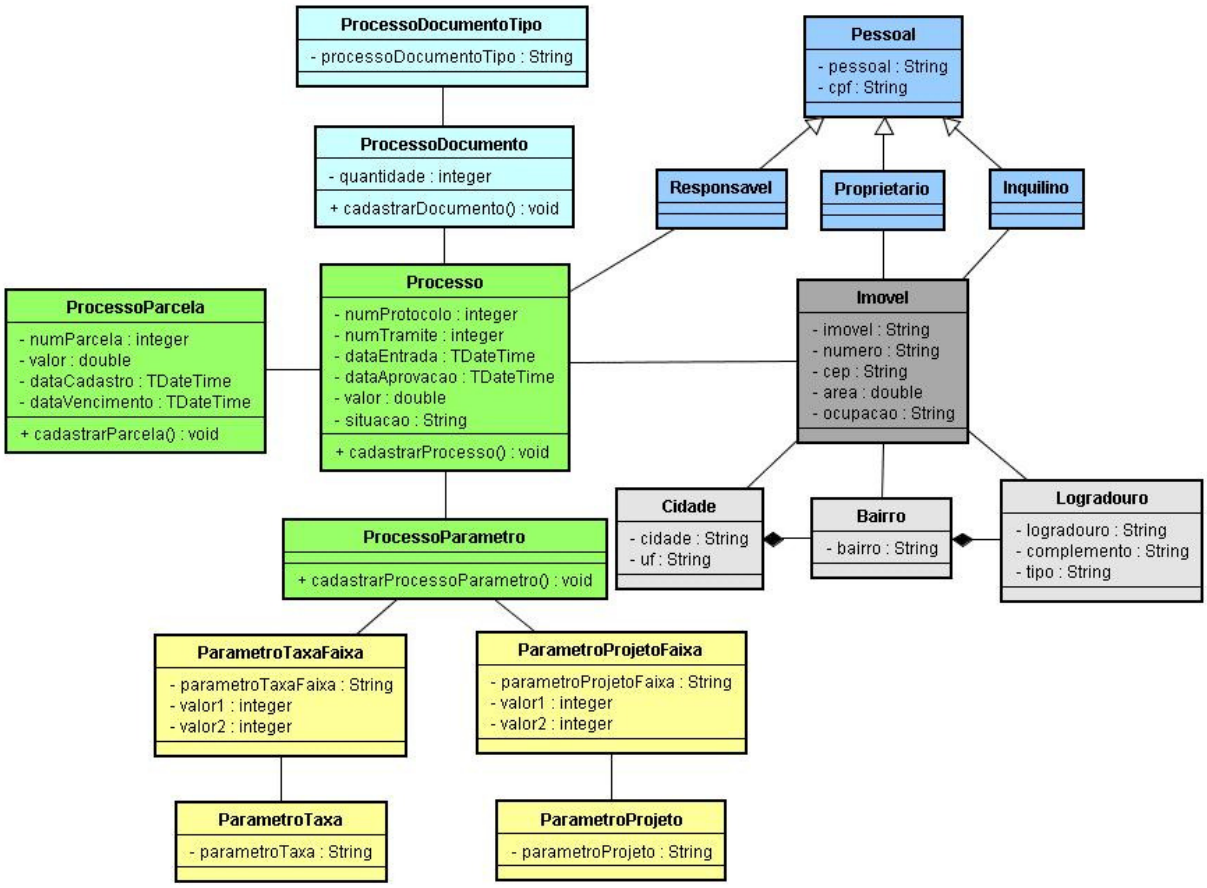

Figura 11 – Diagrama de classe: Cadastrar Projeto

4. CONCLUSÃO

Neste trabalho foram apresentados alguns conceitos de Engenharia de Software, dando destaque à metodologia ICONIX. Também foram feitas reflexões relativas à importância do processo de software como base para o planejamento de projeto e apoio ao desenvolvimento.

Como proposto, foi apresentado um estudo da metodologia ICONIX aliada ao ambiente Delphi com o objetivo de desenvolver aplicações robustas de forma ágil e eficiente. Usando um estudo de caso – o projeto do Sistema de Apoio à Análise de Projetos de Combate a Incêndio – foram abordadas sucintamente as etapas do ICONIX, focando no mapeamento entre os modelos que fazem parte do processo e os componentes existentes no Delphi.

Foi verificado que o ICONIX, com sua abordagem prática e flexível, pode funcionar em plena harmonia com o Delphi. Ambos adotam a idéia de programação tendo a interface como ponto de partida, e procuram simplificar ao máximo a tarefa de projetar e codificar. Além disso, por ser o ICONIX um processo menos burocrático que os demais, permite uma maior adequação dos diagramas ao modelo de programação do Delphi.

A metodologia apresentada é utilizada pelo Núcleo de Desenvolvimento de Software do CEFET-RN na implementação do Sistema de Apoio à Análise de Projetos de Combate a Incêndio do Corpo de Bombeiros Militar do Rio Grande do Norte. Tal metodologia tem apresentado resultados satisfatórios, otimizando o tempo de desenvolvimento do sistema e melhorando a qualidade do produto.

REFERÊNCIAS

MAIA, José Anízio. **Construindo softwares com qualidade e rapidez usando ICONIX**, 2005. Disponível em: <http://www.guj.com.br> Acesso em: 10 jun 2007.

OMG®. Object Management Group **- Unified Modeling Language Specification (2.0)**, 2001. Disponível em: <http://www.omg.org> Acesso em: 27 ago 2007.

OLIVIERO, Carlos A. J. **Sistema Comercial Integrado com Delphi 7: Relatórios & Ferramentas**. 1. ed. São Paulo: Érica, 2003. – (Série Faça um Aplicativo)

SOMMERVILLE, Ian. **Engenharia de Software.** 6 ed. São Paulo: Pearson Education do Brasil, 2003.

AGRADECIMENTOS

Ao Corpo de Bombeiros Militar do Rio Grande do Norte pela parceria e financiamento indispensáveis ao desenvolvimento deste trabalho.