DESENVOLVENDO SOFTWARE COM QUALIDADE E AGILIDADE USANDO ICONIX: ESTUDO DE CASO SISTEMA ORÇAMENTÁRIO DO CEFET-RN.

Ana Luisa F. Medeiros
Estudante do Curso Superior de Tecnologia em Desenvolvimento de Software
Gerência de Tecnologia da Informação e Educacional de Telemática – CEFET-RN
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: analuisafdm@gmail.com

Michelle Furtado
Estudante do Curso Superior de Tecnologia em Desenvolvimento de Software
Gerência de Tecnologia da Informação e Educacional de Telemática – CEFET-RN
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: micfurtado@gmail.com

Leonardo Ataide Minora
Gerência de Tecnologia da Informação e Educacional de Telemática – CEFET-RN
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: minora@cefetrn.br

RESUMO

O processo de desenvolvimento de software vem evoluindo sistematicamente ao longo das três últimas décadas, e para ocorrer a melhoria de sua qualidade e gerenciamento, a existência de um modelo é apontada como um dos primeiros passos. Um novo segmento da Engenharia de Software vem defendendo processos simplificados, também conhecidos como "processos ágeis", dentre os quais inclui-se o ICONIX, um processo prático que utiliza notação UML *(Unified Modeling Language)* . Partindo desse pressuposto, o presente artigo apresenta a descrição de uma aplicação prática utilizando ICONIX, no Desenvolvimento do Sistema Orçamentário para o CEFET-RN, e os principais conceitos que envolvem esse processo e a qualidade de processo e de produto de software.

PALAVRAS-CHAVE: engenharia de software; processo de software; processos ágeis; ICONIX; qualidade de processo, qualidade de software.

1 INTRODUÇÃO

A competitividade e a necessidade ascendente por uma construção de softwares de qualidade em prazos curtos tem causado uma mudança na forma de trabalhar das empresas, as quais estão procurando soluções para a organização do processo de desenvolvimento de software, fazendo com que a Engenharia de Software, antes aplicada com mais freqüência no ambiente acadêmico entrasse na área do mercado de desenvolvimento de sistemas.

Nos últimos anos, novos tipos de abordagens utilizadas em processos para o desenvolvimento de software vêm surgindo com o intuito de melhorar a qualidade do software produzido. Dentre os novos processos que utilizam essa abordagem está o ICONIX, uma metodologia simples e prática que utiliza a notação UML, é dirigido por casos de uso e seu processo é iterativo e incremental, não se trata de um processo burocrático, por não gerar muita documentação, além de ser voltado a resultados para o futuro cliente/usuário do software.

O objetivo deste trabalho é mostrar aspectos e conceitos que envolvem a aplicação prática usando o ICONIX, bem como os conceitos de qualidade de produto e processo de software aplicados no desenvolvimento de estudo de caso, o Sistema Orçamentário do CEFET-RN. O Sistema Orçamentário está sendo desenvolvido pelo DATINF/NUDES (Departamento Acadêmico de Tecnologia da Informação/Núcleo de Desenvolvimento de Software) em parceria com o Departamento de Pesquisa do CEFET-RN, sendo financiado pela concessão de bolsa para estudante do Curso de Tecnologia em Desenvolvimento de Software.

2 O PROCESSO ICONIX

Processo de desenvolvimento de software é a especificação de um conjunto ordenado de atividades que utilizam, geram e modificam artefatos de um software (Sommerville, 2003;Scott, 2003). Artefato é qualquer porção significativa de informação sobre o software, e pode ser desde uma lista de requisitos ou um protótipo de interface com o usuário a diagramas de projeto e código fonte (Scott, 2003). Ainda, as atividades podem ser especificadas por um conjunto de técnicas e metodologias (Sommerville, 2003; Scott, 2003).

2.1 Manifesto Ágil

Na década de 90 surgiu um novo grupo de processos de desenvolvimento, insatisfeito com os resultados obtidos com as abordagens existentes de desenvolvimento (Ambler, 2004). No início eram iniciativas independentes, porém com o tempo, os novos processos se juntaram em torno de um conjunto de valores e princípios comuns, dando origem ao desenvolvimento ágil de software (Ambler, 2004).

O manifesto ágil[1] surgiu do encontro, no ano de 2001, de 17 líderes representantes de novos processos ágeis, eles pretendiam estabelecer pontos em comum com relação ao novo estilo de desenvolvimento que havia surgido (Ambler, 2004). Este documento pode ser acessado no sítio do manifesto ágil[1]. Nesse encontro foi cunhado o termo "ágil" para representar o estilo de desenvolvimento proposto por esses líderes, porém o principal resultado foi a proclamação de um manifesto, assinado pelos presentes ao encontro – um documento definindo os valores e princípios comuns compartilhados por todo o grupo e suportados por seus processos (Ambler, 2004). Alguns princípios do manifesto são:

- Indivíduos e interações mais que processo e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração do Cliente mais que negociação contratual;
- Responder às mudanças mais que seguir um plano.

Assim, dentre as metodologias ágeis que surgiram, tem-se o ICONIX (Bona, 2002). O ICONIX é um processo simplificado que unifica conjuntos de métodos de orientação a objetos em uma abordagem completa, com o objetivo de dar cobertura ao ciclo de vida (Rosemberg e Scott, 1999; Bona, 2002). Foi elaborado por Rosenberg e Scott a partir da síntese do processo unificado pelos "três amigos" – Booch, Rumbaugh, e Jacobson – o qual tem dado suporte e conhecimento a metodologia ICONIX desde 1993 (Rosenberg e Scott, 1999).

[1] http://www.agilealliance.org/

De acordo com Silva e Videira (2001), o ICONIX é uma metodologia prática, e intermediária entre a complexidade do RUP *(Rational Unified Process)* e a simplicidade do XP *(Extreme Programming)*. O ICONIX está adaptado ao padrão da UML – *Unified Modeling Language* (OMG®, 2001), é dirigido por casos de uso e o seu processo é iterativo e incremental.

O desenvolvimento iterativo é organizado em uma série de miniprojetos, de duração fixa chamada iterações; o produto de cada um é um sistema testado, integrado e executável (Bona, 2002; Craig, 2004; Rosemberg e Scott, 1999). O ciclo iterativo é baseado em refinamentos e incrementos de um sistema por meio de várias iterações, com realimentação *(feedback)* e adaptações cíclicas como principais propulsores para convergir para um sistema adequado (Craig, 2004). Assim, o sistema cresce incrementalmente, iteração por iteração.

O ICONIX é composto pelo Modelo Estático e Modelo dinâmico, ambos podem ser desenvolvidos em paralelo, ou de modo recursivo (Bona, 2002; Rosemberg e Scott, 1999). O Modelo Estático é composto por Diagramas de Domínio, Diagramas de Classe, modelam o funcionamento do sistema, sem dinamismo e interação com o usuário (Bona, 2002; Rosemberg e Scott, 1999). Enquanto o Modelo Dinâmico, mostra o usuário interagindo com o sistema, por meio de eventos onde o usuário recebe resposta em tempo de execução (Bona, 2002; Rosemberg e Scott, 1999).

De acordo com Rosemberg e Scott (1999) as principais características do ICONIX são:

- Iterativo e incremental: várias iterações ocorrem entre o desenvolvimento do modelo de domínio e a identificação dos casos de uso. O modelo estático é incrementalmente refinado pelo modelo dinâmico;
- Rastreabilidade (*traceability*): cada passo referência para os requisitos de alguma forma. Nos permite "obrigatoriamente" por meio de seus mecanismos, verificar no decorrer de todas as fases se os requisitos estão sendo atendidos. De acordo com Silva e Videira (2001), a rastreabilidade é a capacidade de seguir a relação entre os diferentes artefatos produzidos. Assim, o impacto pode ser determinando e qual a alteração de um requisito tem em todos os artefatos restantes;
- Aerodinâmica da UML: a metodologia oferece o uso "aerodinâmico" da UML, como: os diagramas de casos de uso, diagramas de seqüência e colaboração, diagramas de robustez.

2.2 Descrevendo os artefatos gerados

Apesar da abordagem do ICONIX ser flexível e aberta – se for necessário é possível utilizar outro recurso da UML para complementar os recursos usados nas fases do ICONIX (Bona, 2002; Rosemberg e Scott, 1999) – aqui neste trabalho será descrito brevemente somente os recursos da UML. Por questões de espaço, neste trabalho não conterá descrições sobre os recursos da UML. Para maiores detalhes é sugerido os livros de Fowler (2005) e Pender (2004), além da especificação que pode ser adquirida em OMG (2001).

Ainda, neste trabalho será seguindo a didática utilizada por todos os documentos consultados sobre o ICONIX, mostrar a tarefas e os recursos utilizados nesta. Segundo Rosenberg e Videira (1999), as principais tarefas que compõem o ICONIX são: análise de requisitos, análise e projeto preliminar, projeto e implementação. Alguns conceitos resumidos a respeito das tarefas são definidos nas seções seguir.

2.2.1 Análise de requisitos

Na tarefa de análise e projeto preliminar, um modelo de domínio é definido, diagrama de classe da UML de alto nível. Tem como objetivo, procurar os objetos do "mundo real" e identificá-los e todas as relações de agregação de generalização entre eles. Se possível, é realizado uma prototipação rápida de interface com o usuário do sistema, na tentativa de facilitar o entendimento ao cliente do que está sendo realizado (Bona, 2002).

Também, nessa etapa, são identificados os casos de uso do sistema, que são representados através de um modelo de caso de uso, mostrando os atores envolvidos.

2.2.2 Análise e projeto preliminar

Nessa atividade os casos de uso são descritos, com os seguintes itens: pré e pós condições, fluxo principal de suas ações, o fluxo alternativo e o fluxo de exceção. Para maiores detalhes sobre descrição de casos de uso, ver os livros Rosemberg e Scott (1999) e Cockburn (2005).

É realizada uma análise de robustez para cada caso de uso com o objetivo de identificar os objetos do modelo de domínio, além de encontrar e especificar as interações entre a interface com o usuário e o sistema (Rosemberg e Scott, 1999; Bona, 2002). Na literatura é sugerido que esta análise seja registrada em um diagrama de robustez, um diagrama que não faz parte da especificação da UML (Rosemberg e Scott, 1999). Para maiores detalhes sobre diagrama de robustez, podem ser consultados: o trabalho de Bona (2002), e os livros Rosemberg e Scott (1999) e Scott (2003).

2.2.3 Projeto

A tarefa de projetos é composta pela especificação de um diagrama de seqüência para cada caso de uso, com a identificação das mensagens entre os objetos. Se necessário, também pode ser utilizado o diagrama de comunicação[2] para representação de transações. Ambos os diagramas fazem parte da especificação da UML.

Por fim, no diagrama de classes são adicionados detalhes inerentes a esta tarefa, tais como: classes descobertas no projeto, os métodos e atributos (Rosemberg e Scott, 1999; Bona, 2002). O diagrama de classe faz parte da especificação da UML.

2.2.4 Implementação

Segundo Silva e Videira (2001), a atividade de implementação não é foco de interesse do processo do ICONIX, sendo uma atividade meramente de tradução dos diagramas para códigos fontes.

Bona (1999) sugere para essa tarefa algumas atividades: testes unitários, testes de integração e testes de aceitação do usuário.

3 QUALIDADE

A qualidade é hoje o grande motivador para as diversas atividades humanas, não sendo diferente na área de desenvolvimento de software. A Norma ISO 8402 (1993) conceitua qualidade como sendo "a totalidade de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas do cliente".

Bartié (2002), e Falbo e Duarte (2000) definem qualidade está subdivida em 2 (dois) tópicos: qualidade do processo e qualidade do produto, como descrito nas subseções a seguir.

3.1 Qualidade de Processo

A qualidade do processo indica que quanto maior a qualidade do processo de desenvolvimento maior a probabilidade de o software ter melhor qualidade em termo de requisitos do cliente (Bartié, 2002; Falbo e Duarte, 2000; Tsukumo, 1997). Ainda, que o processo de desenvolvimento deve continuamente procurar melhorar seu sub-conjunto de atividades, melhorando suas técnicas e metodologias (Bartié, 2002). Existe alguns selos de qualidade para a qualidade e maturidade do processo de desenvolvimento de uma empresa, dentro os principais temos o CMMi, MPSBr, ISO, Spice (Bartié, 2002).

O ICONIX é considerado um processo de qualidade por ser simples e, principalmente, por permitir a rastreabilidade dos requisitos do cliente. Contudo, ele não especifica nenhuma atividade para avaliar o processo continuamente.

---

[2] Diagrama de comunicação para a versão UML 2, para versões anteriores da especificação considerar diagrama de colaboração.

3.2. Qualidade de Produto

Enquanto a qualidade do processo procura garantir a qualidade dos requisitos do cliente, a qualidade do produto procura garantir a qualidade tecnológica (Bartié, 2002; Falbo e Duarte, 2000). Entenda como qualidade tecnológica a verificação da inexistência de erros de implementação (*bug's*), o desempenho do software, a usabilidade do software, entre outras (Bartié, 2002).

O ICONIX, como afirmado por Bona (2002), não possui atividades para garantia de qualidade do produto. Contudo Bona (2002) sugere a adição de testes para este propósito.

4 O SISTEMA ORÇAMENTÁRIO DO CEFET/RN

Este trabalho esta sendo desenvolvido por 2 alunas do Curso Tecnologia em Desenvolvimento de Software, do Departamento Acadêmico de Tecnologia da Informação do CEFET/RN, com o apoio de bolsas do Departamento de Pesquisa do CEFET/RN.

Teve seu início em março de 2006 e tem previsão de término em dezembro de 2006. Este projeto é um projeto piloto que tem a proposta de servir como base à futuros trabalhos de implementação de outros softwares do próprio CEFET/RN.

Ainda, o projeto tem como ferramenta de apoio um sítio criado para centralizar os documentos e códigos gerados que pode ser acessado em http://www.nudes.cefetrn.br/~orcamentario/

4.1 Descrevendo o estudo de caso

O sistema orçamentário receberá informações referentes a programas, elementos de despesa e fontes das unidades administrativas do CEFET/RN dentro de um ano orçamentário, e através desses dados gerar formulários para automatização do processo de controle de gastos e orçamentário. Ainda tem como proposta, fazer o controle da especificação do planejamento anual das unidades orçamentária.

Atualmente, está sendo implementado o caso de uso Planejar Orçamento, anteriormente foi realizado o caso de uso Lançar Despesas, sendo representado nesse trabalho. Nas seções abaixo, serão descritos os artefatos gerados no desenvolvimento do trabalho.

As ferramentas e tecnologias que estão sendo utilizadas no projeto são:

- JUDE Community Free[3] para desenho dos diagramas UML;
- Eclipse 3.2[4] para implementação dos códigos fontes;
- HTML e navegador Mozilla Firefox[5] para gerar protótipos e testar as interfaces com o usuário;
- JSP e JSTL para montar as páginas web do sistema;
- JVM (Java Virtual Machine) da Sun Microsystem[6] em sua versão 5;
- JUnit[7] em sua versão 3.8 como ferramenta de automação de testes unitários;
- CVS[8] como sistema gerenciador de código, mantido no sítio www.nudes.cefetrn.br.

4.2 Modelo de Domínio

---

[3] https://jude.change-vision.com/
[4] http://www.eclipse.org/
[5] http://www.mozilla.org/
[6] http://www.sun.com/
[7] http://www.junit.org/
[8] http://www.nongnu.org/cvs/

Atualmente o modelo de domínio encontra-se como descrito no diagrama de classe da figura 1. Este esta continuamente sendo incrementado por novas informações a cada ciclo de desenvolvimento.

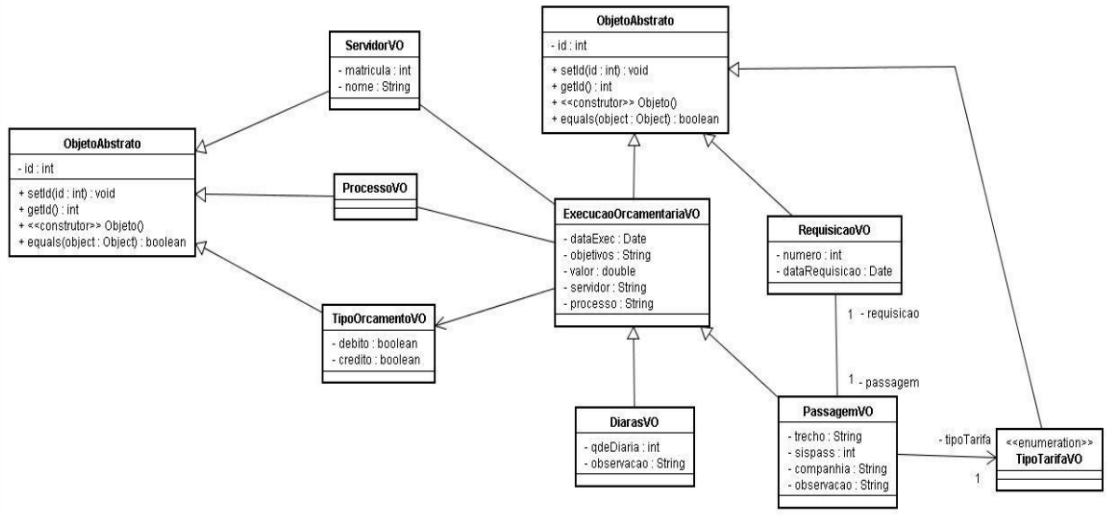

Figura 1 – Diagrama de Domínio do Caso de Uso Lançar Despesas

4.3 Prototipagem de GUI

A figura 2 demonstra o protótipo de interface com o usuário gerado para o caso de uso Executar despesa.

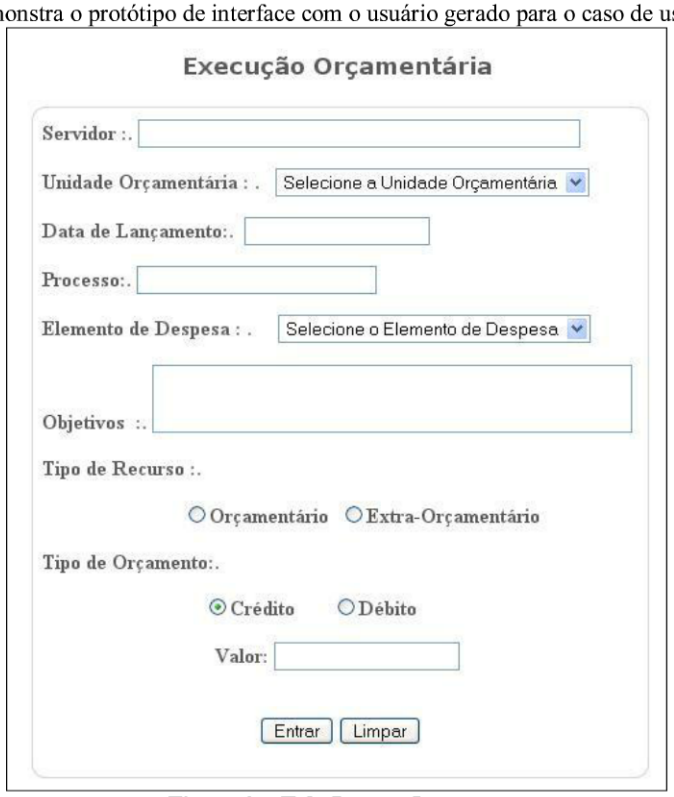

Figura 2 – Tela Lançar Despesas

4.4 Modelo de Caso de Uso

O sistema orçamentário está especificado pelo diagrama de casos de uso da figura 3. Neste são descritos todos os atores do sistema, bem como todas as interações entre os atores e o sistema orçamentário.

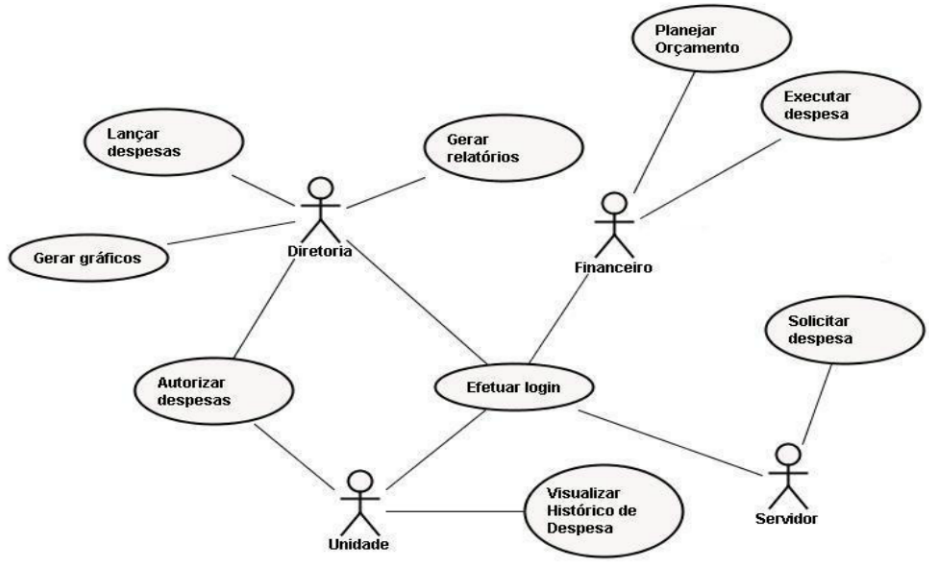

Figura 3 – Diagrama de Caso de Uso

4.4.1 Descrição dos casos de uso

A seguir são descritos de forma resumida os casos de uso mostrados na figura 3:

- Lançar Despesas: O Funcionário irá lançar no Sistema as despesas geradas pelas Unidades que compõem o CEFET/RN.
- Planejar Orçamento: Este caso de uso será responsável pelo planejamento de recursos pelo Financeiro
- Efetuar Login: Os atores terão acesso ao Sistema através de um login e senha.
- Autorizar Despesas: A Unidade ou Diretoria autorizarão as despesas através do Sistema.
- Visualizar Histórico de Despesas:
- Solicitar Despesas: O Servidor solicitará pelo Sistema o elemento de despesa.
- Gerar Gráficos: O Sistema irar gerar gráficos relacionados ao orçamento,solicitados pela Diretoria
- Gerar Relatórios: O Sistema irar gerar relatórios solicitados pela Diretoria.
- Executar Despesas: O Financeiro executará a despesa, caso seja autorizada.
- Planejar Orçamento: O Financeiro distribuirá os recursos para cada unidade orçamentária.

4.4.2 Descrição dos atores

Os atores, e suas respectivas descrições, estão listadas a seguir:

- Servidor: O Servidor será o funcionário do CEFET/RN que irá solicitar via web através do Sistema Orçamentário o elemento de despesa , essa solicitação irar gerar um processo.
- Unidade: A Unidade Orçamentária, será uma das unidades que compõem o CEFET/RN, como também seus departamentos, ela irá receber a solicitação feita pelo servidor através do Sistema, irá autorizar ou não o elemento de despesa requerido, dependendo dos recursos disponíveis.
- Diretoria: A Diretoria Geral do CEFET/RN lançará as despesas provenientes das Unidades Orçamentárias no Sistema, emitirá relatórios e gráficos.

- Financeiro: O Departamento Financeiro executará as despesas depois que autorizadas pela Diretoria, como também, poderá autorizar despesas, dependendo da situação e do processo gerado pela solicitação do servidor.

4.5 Análise de Robustez

Abaixo, no diagrama de robustez da figura 4, está o resultado da análise de robustez para o caso de uso Executar Despesa.

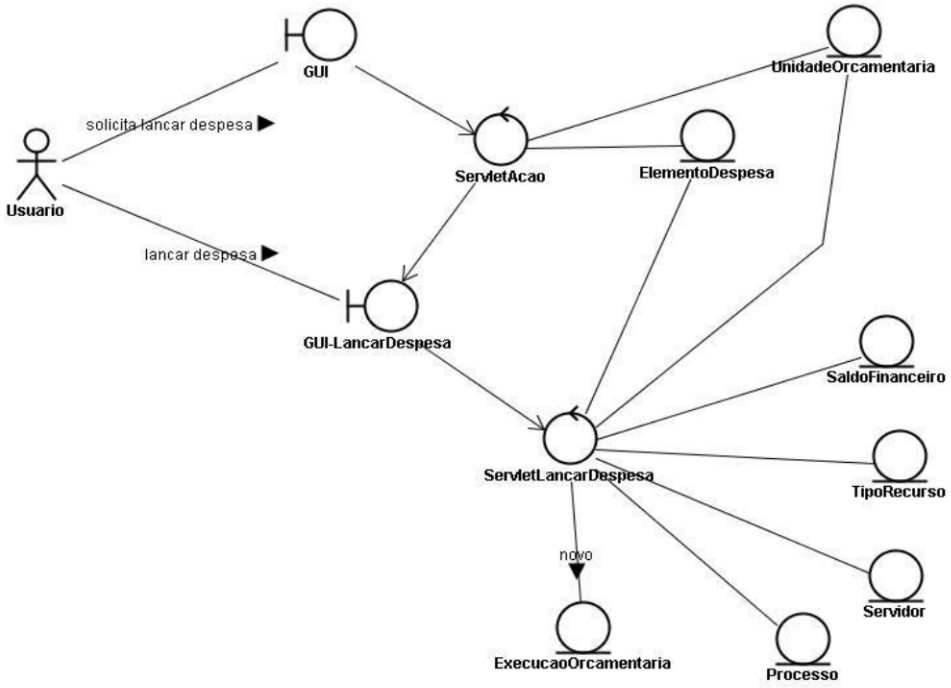

Figura 4– Representação da Análise de Robustez

4.6 Diagrama de Seqüência

Para o diagrama de robustez da figura 4, foi construído o diagrama de seqüência da figura 5.

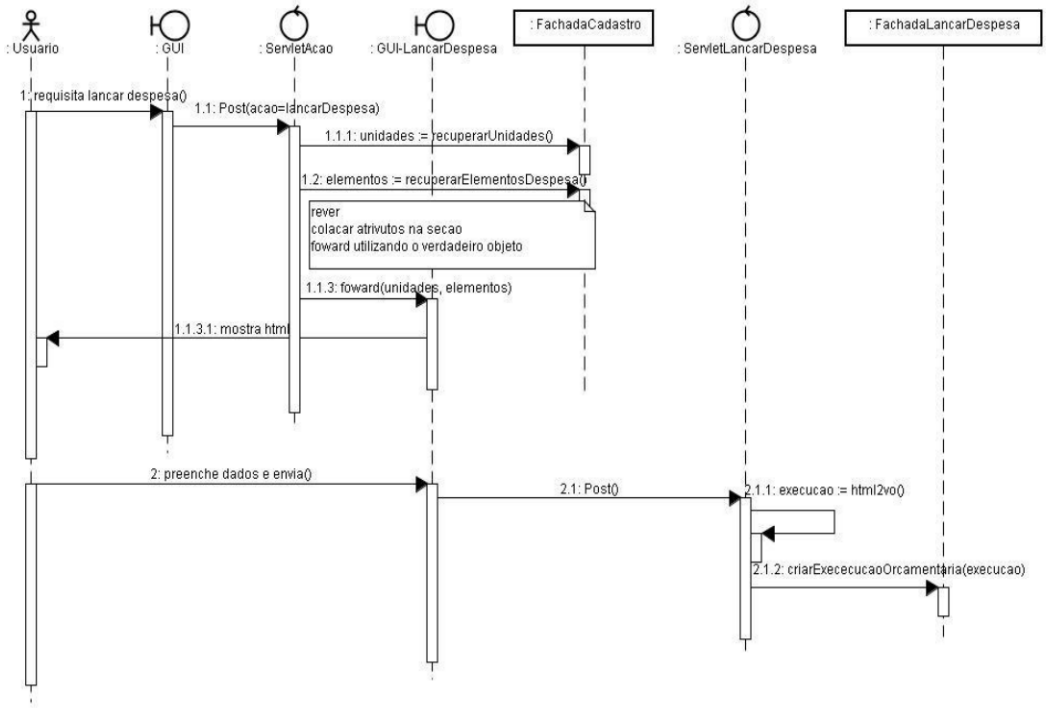

Figura 5 – Diagrama de Sequência do caso de uso Lançar Despesas

4.7 Implementação

De acordo com o diagrama de seqüência da figura 5, foram implementados códigos fontes (JSP e Java) que estão guardados no repositório do projeto. Estes não estão aqui dispostos por limitações de tamanho do documento, mas num futuro este será liberado para acesso a comunidade por acesso anônimo ao repositório do projeto.

5 CONSIDERAÇÕES FINAIS

As atividades especificadas no ICONIX serviram como guia para os alunos no desenvolvimento de parte do software, o projeto ainda não terminou, o que resultou nos artefatos acima apresentados neste documento (menos os códigos fontes, por limitação de tamanho do presente documento).

Algumas dificuldades foram encontradas durante o desenvolvimento de software com o ICONIX, a principal foi a falta de um documento que especifique a arquitetura de software. Este documento é importante para orientar os desenvolvedores nas tecnologias utilizadas para construção do software (Bass, Clements e Kazman, 2003).

Além da falta da arquitetura de software, também faltou a descrição de como realizar os testes unitários que ao final não foram executados. Estes foram executados de forma artesanal e não foi realizado nenhum registro de sua execução e armazenamento.

Como sugestão para outros projetos, é interessante especificar um documento de arquitetura de software bem como a técnica desejada da execução dos testes a serem realizados.

6 REFERENCIAL BIBLIOGRÁFICO

Bona, Cristina. **Avaliação de Processo de Software: um Estudo de Caso em XP e ICONIX.** Dissertação (Mestrado em Engenharia de Produção) – UFSC/Florianópolis, 2002.

_____ **Gestão da Qualidade e Garantia da Qualidade/Terminologia**. Associação Brasileira de Normas Técnicas, NBR ISO 8402. Rio de Janeiro, 1993.

Craig, Larman. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado.** Porto Alegre: Bookman, 2004.

_____ **Unified Modeling Language Specification (2.0)**. OMG® Object Management Group, 2001. Disponível em: <http://www.omg.org>. Acesso em: 10 out. 2006.

Rosenberg, Doug; Scott, Kendall. **Use Case Driven Object Modeling with UML**: A Practical approach. Massachusetts: Addison-Wesley Longman, 1999.

Sommerville, Ian. **Engenharia de Software.** São Paulo: Addison Wesley, 2003.

Silva, Alberto M. R.; Videira, Carlos A. E. **UML, Metodologias e Ferramentas Case**. Lisboa: Centro Atlântico, 2001.

Tsukumo, Alfredo N. Rêgo, Claudete, et al. **Qualidade de Software: Visões de Produto e Processo de Software**. II Escola Regional de Informática da Sociedade Brasileira de Computação Regional de São Paulo, Piracicaba-SP,1997.

Ambler, Scott W. **Modelagem Ágil: Práticas eficazes para a programação eXtrema e o processo unificado**. Ed Bookman, 2004.

Fowler, Martin. **UML Essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. 3ª ed, Ed Bookman, 2005.

Pender, Tom. **UML – A Bíblia**. Ed Campus, 2004.

Cockburn, Alistair**. Escrevendo Casos de Uso Eficazes: um guia prático para desenvolvedores de software**. Ed Bookman, 2005.

Scott, Kendall. **O Processo Unificado Explicado – UML**. Ed Bookman, 2003.

Bartié, Alexandre. **Garantia da Qualidade de Software: as melhores práticas de engenharia de software aplicadas À sua empresa**. Ed Campus, 2002.

Falbo, Ricardo de Almeida; Duarte, Katia Cristina. **Uma ontologia de qualidade de software**. Anais do VII Workshop de Qualidade de Software, XIV Simpósio Brasileiro de Engenharia de Software, pp. 275-285, João Pessoa, Paraíba, Brasil, Outubro 2000.

Bass, Len; Clements, Paul; Kazman, Rick. **Software Architecture in Practice**. 2ª ed, Ed Addison-Wesley Professional, 2003.