Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação
Especialização em Informática. Ênfase: Engenharia de Software

GERSON LUCIO FERREIRA

REUTILIZAÇÃO EM ENGENHARIA DE REQUISITOS

Belo Horizonte
2013

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação
Especialização em Informática. Ênfase: Engenharia de Software

REUTILIZAÇÃO EM ENGENHARIA DE REQUISITOS

Por

Gerson Lucio Ferreira

Monografia de final de Curso
Número da Monografia – fornecido pelo curso

Professor Dr. Eduardo Figueiredo
Orientador

Belo Horizonte
2013

GERSON LUCIO FERREIRA

A REUTILIZAÇÃO DA ENGENHARIA DE REQUISITOS

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do certificado de Especialista em Informática.

Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Eduardo Figueiredo

Belo Horizonte
2013

A Deus,
À minha esposa,
Aos meus filhos,
À toda a equipe de coordenação do curso de pós-graduação Engenharia de Software turma 17 da UFMG
Aos meus familiares
Ao meu orientador Eduardo Figueiredo
dedico este trabalho.

“Graças te rendemos, ó Deus; graças te rendemos, e invocamos o teu nome, e declaramos as tuas maravilhas.
Pois disseste: Hei de aproveitar o tempo determinado; hei de julgar retamente.
Vacilem a terra e todos os seus moradores, ainda assim eu firmarei as suas colunas.
Digo aos soberbos: Não sejais arrogantes; e aos ímpios: Não levantais a foça força.
Não levantais altivamente a vossa força, nem faleis com insolência contra a Rocha.
Porque não é do Oriente, não é do Ocidente, nem do deserto que vem o auxílio.
Deus é o juiz: a um abate e outro exalta.
Porque na mão do SENHOR há um cálice, cujo vinho espuma, cheio de mistura; dele dá a beber; sorvem-no até as escórias, todos os ímpios da terra.
Quanto a mim, exultarei para sempre; salmodiarei louvores ao Deus de Jacó.
Abaterei as forças dos ímpios; mas a força dos justos será exaltada.”
Salmos 75

RESUMO

A análise de requisitos é a parte da Engenharia de Requisitos que se encarrega da fase inicial do desenvolvimento, sua complexidade está em capturar as necessidades e os desejos do cliente e comunicá-los aos demais envolvidos no projeto. Este trabalho tem por objetivo apresentar como pode se dar a reutilização de artefatos produzidos na fase da análise de requisitos de sistemas. Por meio desta reutilização, os Analistas de Sistemas têm a oportunidade de padronizar, customizar e refinar os artefatos produzidos para um sistema. Com isso os analistas podem dedicar mais tempo para atender os interesses de seus clientes e auxiliar os desenvolvedores quanto aos negócios e sistemas sob sua responsabilidade. Neste trabalho foi utilizada a Linguagem de Modelagem Unificada (UML) apoiada pela ferramenta CASE Enterprise Architect de tal forma a possibilitar a reutilização de artefatos da fase de análise. A possibilidade de criação de uma biblioteca de análise para concluir esta fase tem por objetivo reduzir o custo e o esforço das equipes de desenvolvimento de software.

Palavras-chave: Engenharia de Requisitos, Reuso, Análise de Requisitos.

ABSTRACT

Requirements analysis is part of the Requirements Engineering which is in charge of the initial phase of development. This complexity is in capturing the needs and desires of the client and communicate them to others involved in the project. This document aims to present how you to improve reuse of the software artifacts produced during requirements analysis. Through this reuse, the systems analysts have the opportunity to standardize, customize, and refine the artifacts produced for a system. With that analysis, a requirement engineer can devote more time to serve the interests of their clients and assist developers about business and systems under their responsibility. In this work, we used the Unified Modeling Language (UML) supported by the CASE tool Enterprise Architect to enable the reuse of requirements artifacts of the analysis. The possibility of creating a library to complete this analysis phase aims to reduce the cost and effort of software development teams.

Keywords: Requirements Engineering, Reuse, Requirements Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CASE	Computer-Aided Software Engineering
CRUD	Create, Read, Update and Delete
EA	Enterprise Architect
ER	Engenharia de Requisitos
EX	Fluxo de Exceção
FA	Fluxo Alternativo (caso de uso)
FP	Fluxo Principal (caso de uso)
ITE	Interface de sistema
MSG	Mensagens
P	Passos de um fluxo (caso de uso)
RN	Regras de Negócio
RUP	Processo Unificado da Rational (*Rational Unified Process*)
UML	Unified Modeling Language

SUMÁRIO

1. INTRODUÇÃO

A reutilização de artefatos da engenharia de requisitos pode melhorar a produtividade e a qualidade no desenvolvimento de sistemas de software. As características de um software precisam ser determinadas, pensadas e registradas antes que ele se torne um produto. Para tanto, deve-se utilizar algum tipo de categorização destas características. É de grande importância tal categorização a fim de reduzir o risco de que alguns aspectos necessários ao sistema ou, as exigências do cliente, não sejam contemplados no desenvolvimento do software.

Assim, a análise de requisitos é a primeira etapa a ser executada no processo da engenharia de software. Nessa fase os analistas de requisitos devem usar as técnicas e métodos disponíveis para coletar, entender, armazenar, verificar e gerenciar os requisitos. Tais técnicas empregadas na análise de requisitos apoiam os analistas na coleta de necessidades que o sistema de software deverá atender.

Para detalhar aos envolvidos no desenvolvimento de software o que precisa ser o produto, pode ser feito o uso de uma ferramenta CASE como XDE, Visio, ou Enterprise Architect. Essas ferramentas baseadas na UML apoiam a atividade de especificação e comunicação entre os envolvidos na forma de modelos e diagramas de software. Exemplos desses diagramas comumente usados na Engenharia de Requisitos são o Diagrama de Casos de Uso e o Diagrama de Atividades [2].

A ferramenta selecionada para este trabalho é o Enterprise Architect, uma ferramenta de modelagem UML 2.3. Esta ferramenta oferece base para projetar, construir e documentar projetos de sistemas de software. A ferramenta Enterprise Architect proporciona diferentes formas para a prática de reutilização, facilidades para manter a documentação do software bem como para atualizar e criar os modelos de desenvolvimento. Entretanto, é fundamental o emprego de um método de utilização da ferramenta.

A reutilização de sistemas ou de seus componentes é uma proposta que tende a ganhar força a partir da sua aplicação sobre os artefatos produzidos na fase de análise. Os artefatos produzidos para um sistema de software em desenvolvimento

podem ser reaproveitados em análises futuras de outros sistemas similares. Para isso, cabe aos analistas empregar seus conhecimentos no sentido de resolver os problemas que se apresentem e focar em uma solução reutilizável ainda na fase de análise. Tal solução pode ser reutilizada, por exemplo, na especificação de outro sistema de software. Feito isso, a solução pode servir para resolver problemas semelhantes em diferentes sistemas de tal forma a reduzir o custo de desenvolvimento e aumentar a qualidade de software.

O restante desta monografia está organizada a seguinte forma. O Capítulo 2 descreve os conceitos básicos de engenharia de software com o foco na disciplina de análise de requisitos, visando explicar as etapas que esta tarefa envolve e sua complexidade. Passando pelo conceito de modelos de softwares e explicando onde a análise de requisitos se encontra no processo de desenvolvimento selecionado para este trabalho. Ainda no Capítulo 2 é descrita a linguagem selecionada neste trabalho para apoiar a tarefa de criar os modelos de software. O Capítulo 3 apresenta a ferramenta que apoia as tarefas de modelagem de software, nele é descrito as funcionalidades que promovem e apoiam a reutilização. O Capítulo 4 trata da questão central do trabalho, a reutilização dos artefatos na fase da análise de requisitos de um software, nele é explicada a motivação para reusar e os principais focos onde a prática pode ser aplicada. O Capítulo 5 expõe as conclusões do trabalho a partir da experiência adquirida no desenvolvimento de software e nos estudos dedicados a elaboração deste trabalho.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta e discute conceitos relevantes para o contexto deste trabalho, tais como Engenharia de Requisitos (Seção 2.1), Modelagem de Requisitos (Seção 2.2), o Processo Unificado da Rational - RUP (Seção 2.3) e diagramas UML para a Engenharia de Requisitos (Seção 2.4).

2.1. Conceitos da Engenharia de Software

Dentre os conceitos importantes na Engenharia de Software para o escopo deste trabalho, destaca-se o conceito de requisitos de software. Um requisito pode ser definido como uma condição ou uma necessidade do usuário, de um negócio, de um ambiente, ou outro fator externo que o sistema deve satisfazer e estar de acordo [10].

Analisando o software do ponto de vista de produto, ele possui características comuns a qualquer outro tipo de produto. Abaixo são citados alguns desses tipos de requisitos que poderiam ser exigidos por exemplo.

a) Características de uso e instalação

- Para seu funcionamento o software deve ser executado somente em sistemas operacionais Windows.
- O software necessita de no mínimo 2 gigabytes de memória para exibir as imagens.

b) Características de desenvolvimento

- O software deve ser desenvolvido na linguagem C#.
- A base de dados deve ser implementada no MySql.

c) Características de uso (usabilidade)

- O software deve ter ícones representativos para funções salvar e excluir.
- Os botões do sistema devem exibir um texto explicativo sobre a sua função.

Antes de tornar-se um produto, as características do software devem ser descobertas, analisadas, determinadas, e finalmente registradas. Essas tarefas são de responsabilidade da Engenharia de Requisitos (ER).

Um desafio básico do trabalho de determinar os requisitos é encontrar, comunicar e registrar esses requisitos, expressando-os de forma clara para o cliente e os membros da equipe de desenvolvimento [9]. A classificação dos requisitos de software é, tradicionalmente, requisitos funcionais e requisitos não-funcionais. Os requisitos funcionais são aquelas condições que o software deve executar quando sofre uma ação. Por outro lado, os requisitos não-funcionais são aqueles não estão diretamente relacionados às funções especificas do sistema [1].

Outras classificações para os requisitos também são usadas pelo modelo FURPS+ [10]. Usando esse modelo pode se categorizar os requisitos usando o acrônimo que tem o seguinte significado:

- Functionality (Funcionalidade) – características, capacidade, segurança.
- Usability (Usabilidade) – fatores humanos, recursos de ajuda, documentação.
- Reliability (Confiabilidade) – frequência de falhas, capacidade de recuperação, previsibilidade.
- Performance (Desempenho) – tempos de resposta, fluxo de vazão (throughput), precisão, disponibilidade, uso dos recursos.
- Supportability (Suportabilidade) – facilidade de adaptação e de manutenção, internacionalização, configurabilidade.

O “+” indica aspectos auxiliares e subfatores, tais como:

- Implementação – limitações de recursos, linguagens e ferramentas, hardware, etc.
- Interface – restrições impostas pelas interfaces com sistemas externos.
- Operações – gerenciamento do sistema no ambiente operacional.

Como mencionado, é importante utilizar-se de categorizações ou outras formas de organização a fim de reduzir os riscos de que algum aspecto importante do sistema

não seja registrado. A análise de requisitos é a primeira etapa no processo da engenharia de software. Nesta fase, as técnicas e métodos são aplicados para coletar, entender, armazenar, verificar e gerenciar os requisitos. Além disso, durante a análise de requisitos, o que deve ser desenvolvido, é analisado e conhecido [4].

Uma técnica de levantamento e análise de requisitos deve explorar as características específicas do problema. Como as características do problema variam muito, é preciso um repertório de métodos para cada classe de problema [5]. A aplicação de técnicas e métodos no processo de descoberta e levantamento de requisitos é bastante eficaz. Por esta razão, mencionaremos a seguir, algumas das técnicas mais utilizadas pelos especialistas.

Método de Observação: Possibilita um contato pessoal e estreito do analista com o fenômeno pesquisado, o que apresenta uma série de vantagens [6]. Esses métodos, também chamados de estudos etnográficos [1], são úteis para “descobrir” aspectos novos de um problema. Isto se torna crucial nas situações em que não existe uma base teórica sólida que oriente a coleta de dados.

Técnica de Entrevista: Existem dois tipos entrevistas:
a) entrevistas fechadas onde o engenheiro de requisitos procura as perguntas para um conjunto pré-definido de questões;
b) entrevistas abertas onde não há agenda pré-definida, e o engenheiro de requisitos discute, de modo aberto, o que os usuários querem do sistema [7].

Análise de protocolo: A análise de protocolo pede à pessoa se engajar em alguma tarefa e correntemente falar sobre esta tarefa, explicando o seu pensamento do processo. A restrição em estudar protocolos é que as pessoas podem produzir linguagens que oferece um perfil de atividade cognitiva autônoma da pessoa. Tal objeto seria inapropriado para o processo de requisitos porque o cliente não tem qualquer modelo mental pré-existente do sistema desejado.

Método JAD (Joint Application Design): O tema principal do JAD é colocar autoridades representativas e gerenciais juntas dentro de um workshop estruturado para promover decisões. JAD consiste em 5 fases: definição do projeto, pesquisa,

preparação para a sessão JAD, a sessão JAD e o documento final. As fases de definição de projeto e pesquisa no processo JAD lidam com a coleta de informações. JAD promove cooperação, entendimento e trabalho em grupo entre vários grupos de usuários e o pessoal de sistemas de informação [6].

Método QFD: QFD é “um conceito que prevê meios de interpretar requisitos do cliente em requisitos técnicos, apropriados para cada estágio do desenvolvimento e produção do produto” [6]. É um conceito que se aplica bem para a elicitação onde a voz do cliente é o guia para a criação de requisitos.

Método CRC: Em CRC, participantes consistem em não somente de usuários e um facilitador, mas também em outras pessoas envolvidas indiretamente no sistema. CRC é diferente de JAD e QFD, pois ele foca no usuário operativo [3].

Método de Prototipagem: É um método para esclarecer requisitos. Neste método, o projetista precisa manter o protótipo o mais simples possível. O ponto é apresentar alternativas ao usuário antes de iniciar o desenvolvimento. Após a aceitação do protótipo pelos usuários, os desenvolvedores precisam criar um documento de especificação de requisitos paralelo ao protótipo de interface [8].

Método de Cenários: Cenários são exemplos de sessões de interação as quais são concentradas com um tipo único de interação entre o usuário final e o sistema. Usuários finais simulam suas interações. Eles explicam para o time de engenheiros de requisito o que eles estão fazendo e a informação da qual eles precisam do sistema para descrever a tarefa descrita no cenário.

Técnica Brainstorming: Técnica de criatividade para geração de ideias com vistas à resolução de problemas. As ideias são geradas de forma rápida, coletadas e então discutidas e avaliadas pelo grande grupo.

Ao tratar da Engenharia de Software, é importante considerar que a função da Engenharia de Requisitos é capturar as necessidades do cliente e registrar as informações em uma linguagem que todos os envolvidos possam compreender. O registro destas informações deve tornar viável o planejamento, dimensionamento e

desenvolvimento do produto. Assim, as futuras evoluções do sistema que está bem documentado são menos traumáticas.

Para refinar a coleta dos requisitos capturados inicialmente é necessário que a visão em alto-nível do sistema possa ser definida. Um documento de Visão tem o objetivo de fornecer uma perspectiva única do produto entre os envolvidos no projeto. Conforme o processo de desenvolvimento RUP, todo projeto deve ter uma fonte para capturar as expectativas dos envolvidos. Para tanto, o documento de Visão do software é um bom artefato para registrar as expectativas dos participantes do projeto.

O documento de Visão deve ser escrito com base na perspectiva do cliente e seu foco deve ser nas características do sistema. Quando possível, o documento de Visão deve definir os níveis aceitáveis de qualidade. Cabe agora ao analista esclarecer (elicitar) os requisitos. Isto é, ele deve identificar fatos que compõem cada um dos requisitos em particular. Entretanto, alguns requisitos capturados podem não ser relacionados ao sistema em análise. A utilização de técnicas como as citadas anteriormente apoia o analista na coleta dos requisitos do software. A elicitação é a fase na qual será refinado e melhorado o entendimento adquirido na análise inicial.

Uma vez construída a visão comum, elicitados os requisitos funcionais e não funcionais, e com uma lista de requisitos produzida, o analista deve detalhar a equipe de desenvolvimento, o que e como deve ser o software. Os artefatos utilizados para comunicar e detalhar o que se pretende desenvolver aos envolvidos no projeto devem estar de acordo com o modelo de desenvolvimento adotado. Para os processos baseados no RUP, é comum à adoção de casos de uso e diagramas da UML. Por outro lado, métodos ágeis como o Extreme Programming (XP) faz uso de histórias de usuários. De forma semelhante, o SCRUM por sua vez usa o chamado Product Backlog.

É característica dos requisitos de um sistema não permanecer estático, já que normalmente são necessárias alterações, inclusões e manutenções. Cabe à gerência de requisitos o controle, a verificação de sua implementação e efetuar as mudanças nos requisitos, caso sejam necessárias. Também é parte do escopo da

gerência de requisitos a análise dos impactos causados pelas solicitações de alterações durante o desenvolvimento do sistema.

Ferramentas CASE auxiliam a gerência dos requisitos permitindo muitas vezes verificar a rastreabilidade entre os componentes da análise (casos de uso, regras de negócio, mensagens do sistema, atores, interfaces, classes, dentre outros). Este rastreamento tem como objetivo reduzir o esforço para controle e execução da gerência dos requisitos.

Controlar as mudanças de requisitos significa saber os motivos e os impactos que uma alteração em um requisito irá ter sobre o produto desenvolvido e/ou em desenvolvimento. Por outro lado, o controle de versão está intimamente ligado à gerência de configuração e deve ser capaz de recuperar um requisito em sua versão de acordo com a versão do produto. O acompanhamento de versões se dá durante o desenvolvimento do software. É nessa fase que se torna possível verificar a implementação de um requisito. Ou seja, se o requisito está conforme foi originalmente solicitado, se foi desenvolvido e se está funcional.

A rastreabilidade é a parte responsável por verificar os relacionamentos dos requisitos entre si e entre os demais artefatos. Durante o processo de desenvolvimento de software, o entendimento dos stakeholders sobre o problema muda constantemente e o processo de gerenciamento de requisitos é um processo para compreender e controlar as mudanças dos requisitos de sistema [1].

2.2. Modelagem de Requisitos

É inevitável a evolução dos requisitos. A partir deste pressuposto, os requisitos dividem-se em permanentes e voláteis. Os primeiros são relativamente estáveis e derivados da atividade central da organização. Por outro lado, os requisitos voláteis tendem a mudar durante ou depois que o sistema estiver em produção. Os requisitos devem ser expostos através de uma linguagem natural, pois eles devem ser compreendidos por todos os stakeholders. Entretanto, a equipe de desenvolvimento geralmente necessita de mais detalhes técnicos.

A competência de comunicar os requisitos de sistema está diretamente ligada capacidade de o analista de requisitos interagir com os desenvolvedores e usuários (cliente). Para auxiliar os analistas, existem padrões de diagramas e notações que facilitam essa comunicação e agregam valores ao processo de desenvolvimento. Um aspecto importante de um modelo é o fato de os detalhes ficarem de fora e serem uma abstração em estudo. Ou seja, um modelo é uma representação alternativa do sistema [1].

As ferramentas CASE baseadas na UML, apoiam também a atividade de especificação e comunicação entre os envolvidos no desenvolvimento de sistemas com a criação de modelos e diagramas. Com isso, o processo de desenvolvimento ganha em desempenho, e padronização dos produtos gerados na fase de análise. Ou seja, modelos representativos do sistema são mais facilmente alcançados. Neste estudo será dada ênfase à modelagem baseada no processo de desenvolvimento RUP usando as notações da UML apoiada pela ferramenta CASE Enterprise Architect.

2.3. Processo Unificado da Rational (RUP)

O RUP é um processo de desenvolvimento de software baseado em disciplinas que visa definir tarefas e responsabilidades dentro de uma organização de desenvolvimento. A figura 1 apresenta uma visão macro do processo de desenvolvimento RUP.

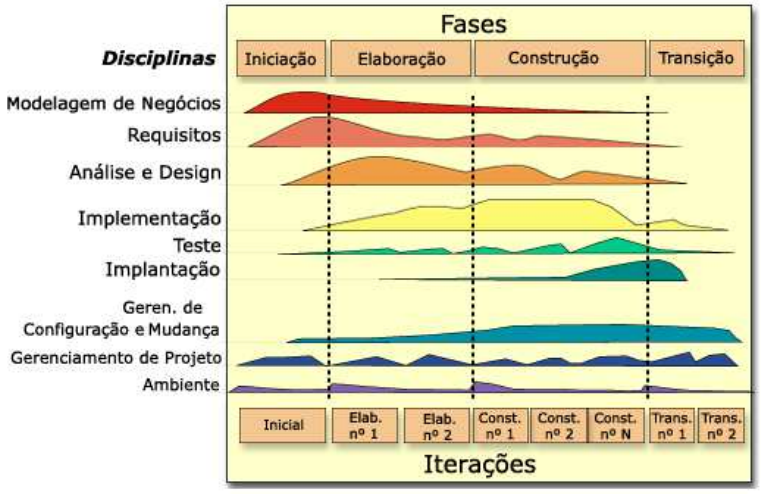

Figura 1 - Fases do Processo de Desenvolvimento RUP

A análise de requisitos é impactante em todo o processo do desenvolvimento. Ela se encontra dentro do modelo como uma segunda etapa representada pela segunda onda do desenvolvimento na figura 1. Embora a análise de requisitos seja uma etapa de preparação para se definir o que será necessário no desenvolvimento, sua presença é forte durante todo o processo de desenvolvimento.

O RUP é geralmente descrito a partir de três perspectivas: dinâmica, estática e prática. A dinâmica mostra as fases ao longo do tempo. Por outro lado, a perspectiva estática mostra as atividades do processo. Finalmente, a perspectiva prática sugere as boas práticas a serem usadas durante o processo [1].

Conforme a figura 1, as fases do modelo de desenvolvimento se integram entre si e retrata a ligação da disciplina de requisitos que se estende por todo o ciclo de vida do desenvolvimento. Utilizando-se da notação UML, suas fases são consideradas pesadas e normalmente é aplicável em projetos com grandes equipes técnicas.

2.4. Modelos UML para ER

A UML é uma linguagem de modelagem para especificar, visualizar, construir e documentar artefatos de sistemas de software [2]. A UML também é usada para modelar negócios e outros sistemas que não sejam de software. Mantida pela OMG desde 1989, a UML foi adota como padrão em 1997. Em sua versão atual (2.4.1), a OMG destaca que além das classes e objetos modelados nas versões 1.X, esta versão procura adicionar a capacidade de representar modelos arquitetônicos, processos de negócios e regras até mesmo de disciplinas não relacionadas à computação [11].

Os diagramas UML podem ser classificados em diagramas de estrutura, diagramas de comportamento e diagramas de interação. Os diagramas de estrutura incluem o Diagrama de Classes, Diagrama de Objetos, Diagrama de Componentes, Diagrama de Estrutura de Composição, Diagrama de Pacote e Diagrama de implantação. Por outro lado, diagramas de comportamento incluem o Diagrama de Caso de Uso (usado por algumas metodologias durante o levantamento de requisitos); Diagrama de Atividades e Diagrama de Máquina de Estado. Além disso, os diagramas de

interação são todos os derivados do diagrama de comportamento mais geral e incluem o Diagrama de Sequência, Diagrama de Comunicação, Diagrama de Tempo e Diagrama de Visão Geral da Interação.

Na ER, alguns dos diagramas UML comumente utilizados são os de comportamento e de interação porque, nesta fase do desenvolvimento, o objetivo é compreender e capturar justamente os comportamentos funcionais e não-funcionais para desenvolver o sistema. A fase de análise de requisitos pode incluir ainda uma descrição dos processos do domínio e escritos como casos de uso. Nesse caso, os requisitos funcionais são registrados e explorados em um modelo de casos de uso. Um caso de uso é uma coleção de cenários relacionados ao sucesso ou fracasso de uma interação. A essência do caso de uso é descobrir e registrar os requisitos funcionais escrevendo narrativas de uso de um sistema que ajude a satisfazer os objetivos de vários interessados [9]. As subseções a seguir descrevem os diagramas UML mais usados na engenharia de requisitos.

2.4.1. Diagrama de Casos de Uso

Os Diagramas de Casos de Uso são representações de funções completas de um sistema. Eles buscam exibir um aspecto maior da funcionalidade do produto. Um conjunto de casos de uso representa toda a funcionalidade de um produto de software. A figura 2 abaixo mostra um exemplo de caso de uso para parte de um sistema de avaliação de desempenho.

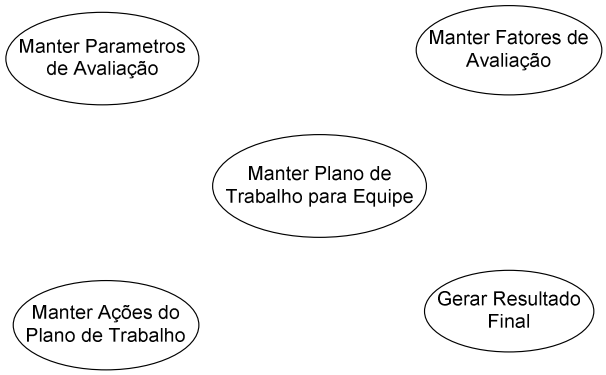

Figura 2 - Casos de Uso do sistema

Os atores na UML e ER são representações dos envolvidos com o sistema. Eles podem ser ou não humanos. Mesmo outros sistemas podem ser identificados como

atores, ou seja, participar das funcionalidades de um produto. Quando o número de atores é muito grande e/ou existem vários grupos com interesses comuns nas funcionalidades de um software, é recomendado aplicar o conceito de herança ou generalização de atores. Por exemplo, a figura 3 mostra os usuários levantados para um sistema e a generalização do ator funcionário.

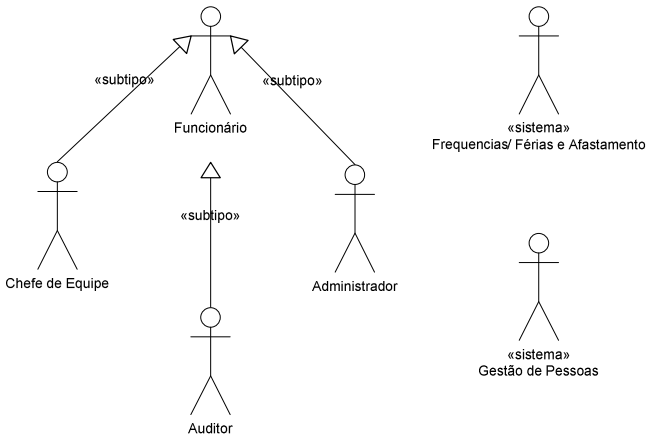

Figura 3 - Generalização de Atores

Um ator é qualquer coisa com um comportamento, inclusive o próprio sistema [9]. Para a ER o estudo dos atores é de fundamental importância para prever as funcionalidades que um sistema deve ter. A ligação entre os atores e os casos de uso visa obter um contexto do sistema. No conjunto deste diagrama, é possível visualizar e prever as fronteiras do sistema e os atores que atuam ou participam com interações ou comunicações entre os casos de uso e os atores. Na UML, as associações são descritas como “um relacionamento semântico entre dois ou mais classificadores que envolvem conexão entre suas instâncias” [9].

A figura 4 mostra os casos de uso e os atores envolvidos no sistema de avaliação de desempenho. Um diagrama de caso de uso é uma excelente imagem do contexto do sistema. Ele é um bom diagrama de contexto. Ou seja, mostra a fronteira de um sistema, o que está fora dele e como o sistema será usado [9]. Um diagrama que mostre o contexto deve exibir as interfaces do produto com seu ambiente de aplicação. Além disso, os diversos tipos de usuários e outros sistemas com os quais o produto deva interagir são representados por atores situados fora de um retângulo que marca a fronteira do produto.

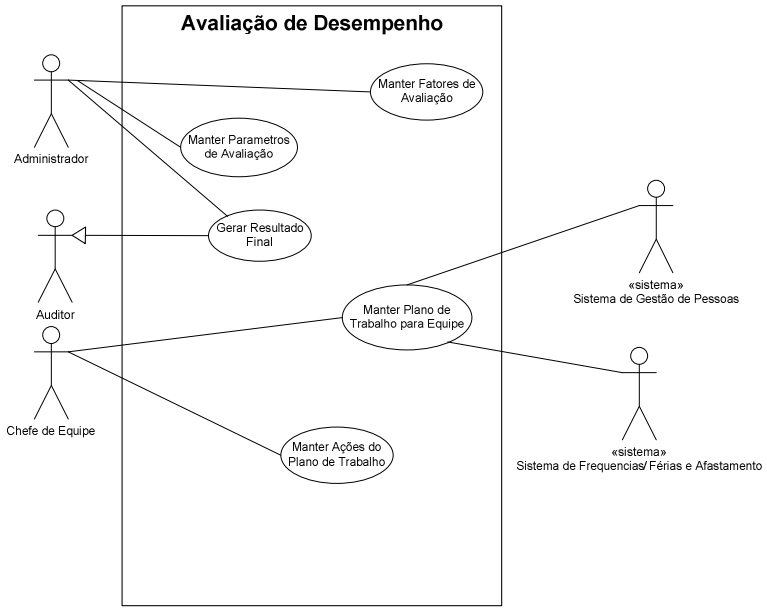

Figura 4 - Diagrama de Caso de uso

2.4.2. Diagrama de Atividades

O Diagrama de Atividades é considerado um caso especial do antigo Diagrama de Gráfico de Estados. Porém, a partir da UML 2.0, este diagrama se tornou independente, deixando inclusive de se basear em máquinas de estados e passando a se basear em redes de petri.

O Diagrama de Atividades é uma variante de fluxogramas, sendo geralmente usados para descrever processos de negócio ou outros fluxos onde o paralelismo de atividades seja importante. Ele se concentra na representação do fluxo de controle e no fluxo de objeto de uma atividade. A figura 5 mostra um diagrama de atividades para identificar os passos necessários para um ator concluir o cadastro em um sistema seu objetivo é identificar as regras que precisam ser impostas ao ator e as opções que o sistema deve oferecer durante a interação com o sistema. O Diagrama de Atividades será aprimorado à medida que o entendimento das necessidades é ampliado.

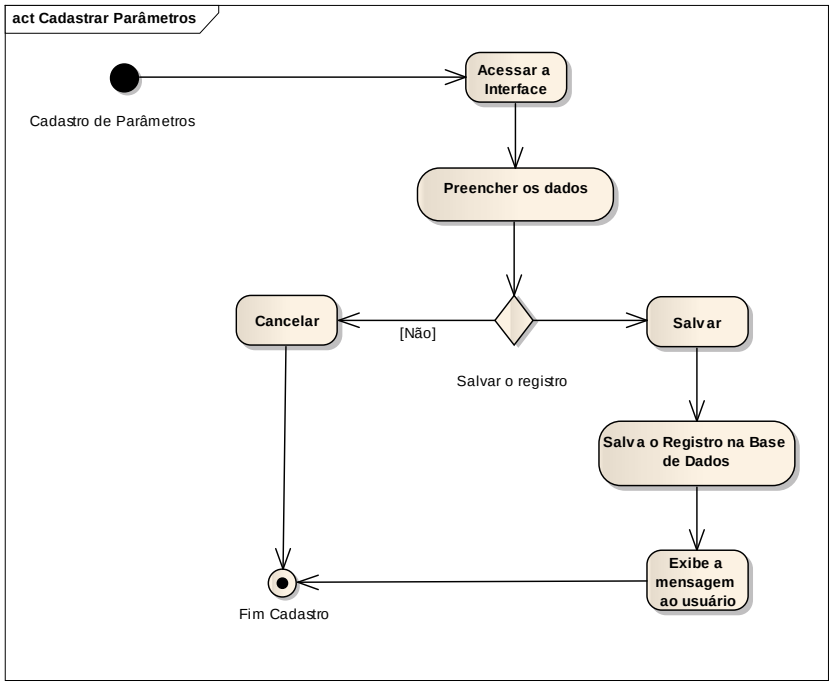

Figura 5 - Diagrama de Atividades

2.4.3. Diagrama de Sequência

O Diagrama de Sequência se preocupa com a ordem temporal em que as mensagens são trocadas entre os atores e o sistema. Um diagrama de sequência é uma figura que mostra os eventos que os atores externos geram a sua ordem e os eventos entre sistemas [9]. A figura 6 mostra um diagrama de sequência para o processo de compras que um sistema pretende apoiar, nele é retratada uma possível sequencia de ações necessárias para descrever as interações dos atores e as respostas que o sistema deve ter em relação ao processo que se deseja apoiar.

O diagrama de sequência busca representar uma série de passos sequenciais no decorrer do tempo e como os elementos interagem ao longo do tempo. Na UML, o uso dos estereótipos é um recurso importante para poder comunicar de forma padronizada as verdadeiras representações desejadas. Os estereótipos são usados para classificar um elemento.

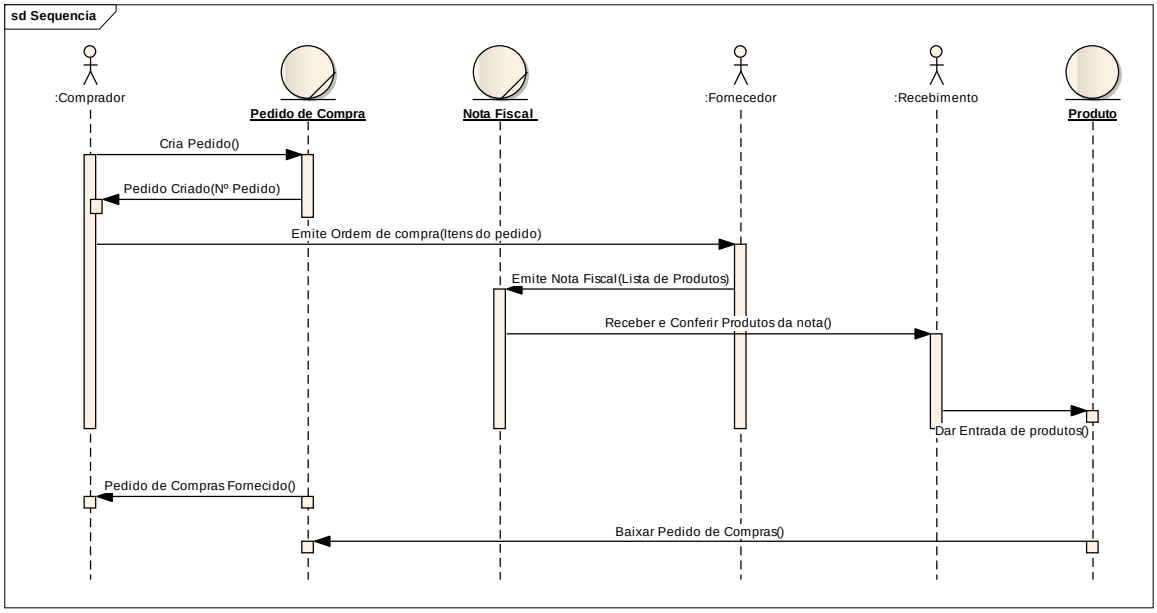

Figura 6 - Diagrama de Sequência

3. ENTERPRISE ARCHITECT (EA)

A ferramenta Enterprise Architect é uma ferramenta de modelagem e serve para projetar, construir e documentar projetos de sistemas de software. Ela pode ser usada tanto para a modelagem de processos de negócios quanto para fins de modelagem generalizada.

A versão usada da plataforma de modelagem do Enterprise Architect é baseada na UML 2.3. A notação fornece um rico conjunto de elementos gráficos para a modelagem de sistemas orientados a objetos. Esta ferramenta é capaz de suportar o controle de versão dos pacotes e seus componentes para um repositório central de controle de versão. Você pode colocar todos os pacotes individuais, nós visualização ou nós raiz do modelo sobre o controle de versão.

Alguns dos componentes presentes na ferramenta são de extrema importância para o emprego da reutilização. Por exemplo, com os estereótipos da UML é possível configurar e incluir na modelagem aspectos importantes para o negócio, definindo como será sua utilização.

A capacidade de exportação e importação dos itens da modelagem permite a reutilização de vários artefatos do produto em análise: regras de negócio, mensagens, protótipos, modelos de dados e podendo, inclusive alcançar o nível textual de casos de uso. Além disso, a ferramenta disponibiliza o uso da engenharia reversa e permite integração com vários sistemas de gerenciamento de banco de dados como SQL Server, MySql, Oracle, Informix, Postgre SQL entre outros. Ela permite ainda a geração de código de artefatos como classes, interfaces e enumeradores, em diferentes linguagens como: Java, C##, Delphi, PHP e outras.

Outro aspecto positivo é a capacidade de criar, manipular e trabalhar os tipos de métricas. A ferramenta permite reusar através da importação e exportação e também atribuir os tipos de métrica criados aos artefatos do modelo. Esta funcionalidade possibilita dimensionar o esforço para elaboração ou desenvolvimento do produto.

O gerenciamento do custo e avanço do projeto ou produto permite definir os valores por hora e usar estimativas com base no nível de dificuldade dos casos de uso, fatores ambientais e fatores técnicos de complexidade. Não menos relevantes, as funcionalidades que promovem a rastreabilidade exploram rapidamente a cadeia de relacionamento dos elementos da modelagem. Tais funcionalidades fornecem aspecto decisivo para verificar a viabilidade das alterações propostas ou impostas a um produto de software. Por exemplo, quando é necessário alterar um item da modelagem, o projeto deve prever o impacto da mudança nas demais áreas do projeto. A figura 7 exemplifica o benefício de utilização da ferramenta CASE. Nessa figura, o analista pode verificar que uma alteração na mensagem “MSG010” ou nas “RN001” e “RN002” terão impacto em apenas um dos casos de uso.

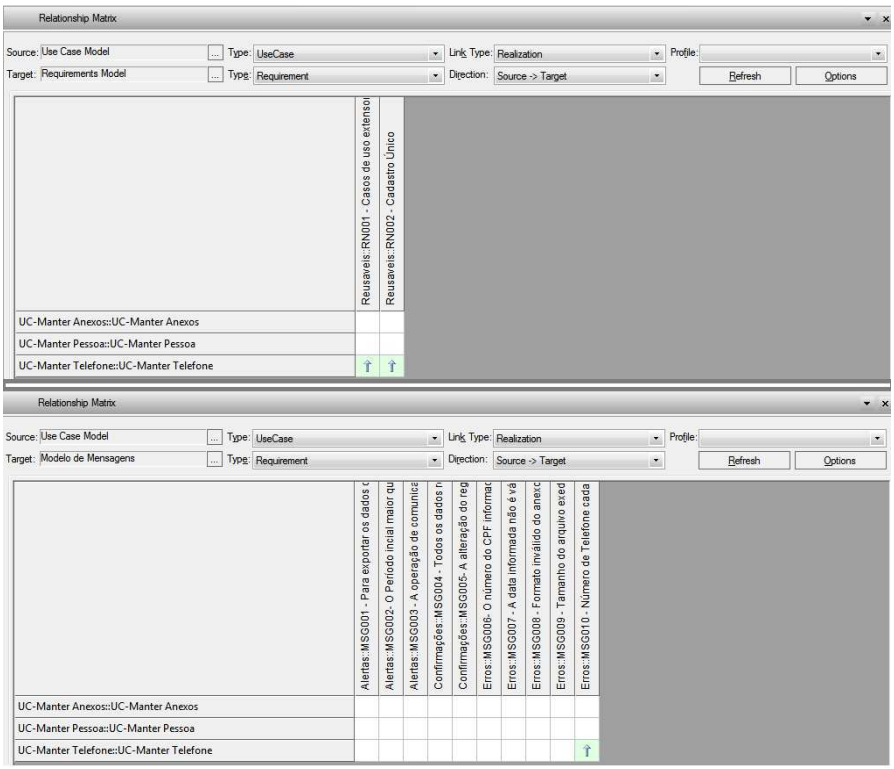

Figura 7 - Matrix de rastreabilidade

A rastreabilidade em um modelo simples pode ser implementada por meio de planilha ou com anotações. Porém, à medida que a complexidade e quantidade de artefatos crescem o esforço para mapear e manter o relacionamento entre os artefatos fica muito caro ao projeto.

Embora uma ferramenta CASE possa proporcionar várias formas para prática de reutilização e facilidades em manter e atualizar os modelos de desenvolvimento, é necessário que seja criada uma metodologia de utilização e implementação dos modelos. Além disso, políticas bem definidas devem ser criadas visando à utilização de forma eficaz e sustentável. Para atender esta necessidade, uma suíte de análise pode ser criada como template para utilização de todos os membros da equipe de desenvolvimento. Essa suíte deve ser colocada sobre a gerência de configuração permitindo o crescimento e ampliação da capacidade de reutilização e, naturalmente, criando uma biblioteca de análise de sistemas.

4. REUTILIZAÇÃO NA ENGENHARIA DE REQUISITOS

4.1. Motivação para reusar

A tabela abaixo exibe um benefício e um problema para reusar que está diretamente ligado à engenharia de requisitos [1].

Benefício	Explicação
Desenvolvimento acelerado	A apresentação de um sistema para o mercado tão cedo quanto possível é muitas vezes mais importante do que os custos totais de desenvolvimento. O reuso de software pode tornar rápida a produção do sistema porque tanto o tempo de desenvolvimento quanto o de validação devem ser reduzidos
Problema	Explicação
Falta de apoio de ferramenta	O conjunto de ferramentas CASE podem não apoiar o desenvolvimento com reuso. Pode ser difícil ou impossível integrar essas ferramentas a um sistema de biblioteca de componentes. O processo de software suposto por essas ferramentas pode não levar em conta o reuso

Tabela 1- Beneficio e problema de reusar.

Para que seja possível o desenvolvimento de um software, a equipe deve ter os devidos artefatos para produzir a solução aderente e adequada à solicitação do cliente. Para tanto, a análise e especificação dos requisitos inevitavelmente têm de ser realizada de forma a não perder os detalhes indispensáveis ao desenvolvimento. Essa análise tem como objetivo atender as necessidades do cliente.

O processo de especificação pode ser acelerado através da criação de bibliotecas de requisitos, onde os aspectos de um software estejam em um nível de abstração que permita, por exemplo, a sua reutilização e padronização. Dentre as possibilidades de reusar podemos citar: (i) a integração com webservices pode ser especificada uma única vez e (ii) com pouca alteração, um modelo pode ser reutilizado para diferentes sistemas. Da mesma forma, a funcionalidade de anexar

arquivo pode ser padronizada e reusada para várias soluções. Outra padronização pode ser obtida com a especificação da funcionalidade de inserir números de telefones vinculados a um cadastro, quando esta for definida uma única vez e reusada para outros tantos sistemas. Em alguns casos, devem ser executadas alterações especificas conforme a análise de necessidades.

A reutilização deve explorar o fato de que sistemas em um mesmo domínio de aplicação são similares [1]. Este conceito pode ser ampliado se analisada também as características de determinadas funcionalidades comuns nas diferentes aplicações. Assim, para exemplificar, mencionamos as funcionalidades CRUD[1] que são comumente utilizadas em todos os sistemas de informação. Assinalando que os comportamentos de um CRUD são similares, eles podem ser reusados com pequenas ou nenhuma adequação.

Ao se utilizar a metodologia de desenvolvimento RUP, o artefato denominado caso de uso descreve o que foi descoberto, analisado e definido para produção e implementação do sistema. Confeccionar o caso de uso demanda tempo e, no que diz respeito ao quesito qualidade, implica em alto risco para o projeto uma vez que ele está ligado às demais disciplinas e à fase do desenvolvimento. Assim, a utilização de uma ferramenta CASE, como por exemplo, o Enterprise Architect, estruturada para reusar os artefatos de análise pode acelerar a produção e garantir a melhoria da qualidade dos artefatos da fase de análise.

4.2. Reutilização de Cenários

Utilizam-se cenários nos casos de uso para descrever as circunstâncias e as variações de uso de uma função completa do software. Quando o foco é reusar, torna-se importante maximizar as possibilidades. É de suma importância padronizar a escrita e alinhar o entendimento em relação aos cenários, tanto do cliente quanto da equipe de desenvolvimento.

[1] Sigla do inglês para Create, Read, Update and Delete.

Um caso de uso descreve do início ao fim o cenário de uso para executar uma funcionalidade do sistema. Nos cenários, são descritas opções desta funcionalidade que o sistema disponibiliza para o usuário. Os fluxos ou cenários de exceção tratam as condições adversas ao funcionamento do sistema. Eles indicam também quais ações corretivas e/ou alternativas o sistema deve contornar para continuar o seu funcionamento.

O caso de uso, sobre o qual trataremos a seguir, é estruturado em um formato cujo objetivo é reduzir o esforço de descrever os cenários de uso sem alterar a capacidade de entendimento dos desenvolvedores e dos usuários. Estes usuários irão interagir com o sistema em especificação e produção. Segue abaixo um exemplo de cenário para inclusão de parâmetros de avaliação.

FP - Inclusão de Parâmetros da Avaliação
P1. O Sistema exibe a interface para o usuário. [ITE001]
P2. O Ator informa os dados e aciona o comando para salvar as informações. [RN01] [RN..N] [EX01] [EX02]
P3. O Sistema exibe a mensagem [MSG001] ao usuário.
P4. O Ator confirma a ação de cadastro.
P5. O Sistema salva as informações na base de dados e exibe a mensagem [MSG002] ao usuário. [RN03]
P6. Fim do fluxo de inclusão.

A figura 8 abaixo especifica a interface ITE001 de cadastro dos parâmetros de avaliação.

Figura 8 - ITE001 – Cadastro de Parâmetros de Avaliação

A RN01 que deve ser implementada no passo dois, contêm a seguinte descrição textual:
- O sistema deve garantir que no período, a data de início deve ser válida e igual ou maior que a data atual do sistema. Além disso, a data de fim deve ser válida e maior ou igual à data inicial.

A exceção EX01 que representa os campos obrigatórios não preenchidos pode ser levantada no passo dois do fluxo. Ela tem a seguinte estrutura:

EX01 – Campos obrigatórios não preenchidos
P1. O Sistema identifica que campo(s) obrigatório(s) não foi preenchido e exibe a mensagem [MSG03].
P2. O Ator confirma a opção da mensagem
P3. O Sistema fecha a mensagem e retorna ao fluxo acionador sem executar a ação iniciada.

A mensagem “MSG03” será exibida ao usuário no caso da exceção “EX01” ser acionada. Esta mensagem deve ter a especificação Nº003 definida na tabela 2.

Lista de mensagens			
Nº	Texto da Mensagem	Opções de Ação	Observações
001	Confirma o cadastro?	Botão de confirmação "Sim" e botão de cancelamento da ação "Não"	N/A
002	O Período informado não é um periodo valido! A data inicio deve ser me nor ou igual a data fim.	Botão de confirmação de ciencia da mensagem	N/A
003	Campo(s) Obrigatório não preenchido! Os campo marcados com "*" (Asterisco) são de preenchimento obrigatório!	Botão de confirmação de ciencia da mensagem	O Sistema deve marcar os campos obrigatórios não preenchidos para facilitar a identificação do usuário

Tabela 2 - Lista de mensagens do caso de uso

O fluxo de inclusão de outro CRUD do sistema é descrito a seguir. Neste fluxo, o que precisa ser alterado são as referências para a interface, regras, mensagens e outras partes que são especificas entre colchetes.

Fluxo de inclusão dos Fatores de Avaliação

P1. O Sistema exibe a interface para o usuário. [ITE002]

P2. O Ator informa os dados e aciona o comando para salvar as informações. [RN01] [EX01] [EX02]

P3. O Sistema exibe a mensagem [MSG001] ao usuário.

P4. O Ator confirma a ação de cadastro.

P5. O Sistema salva as informações na base de dados e exibe a mensagem [MSG002] ao usuário.

P6. Fim do fluxo de inclusão.

A figura 9 abaixo especifica a interface do cadastro de fatores de avaliação.

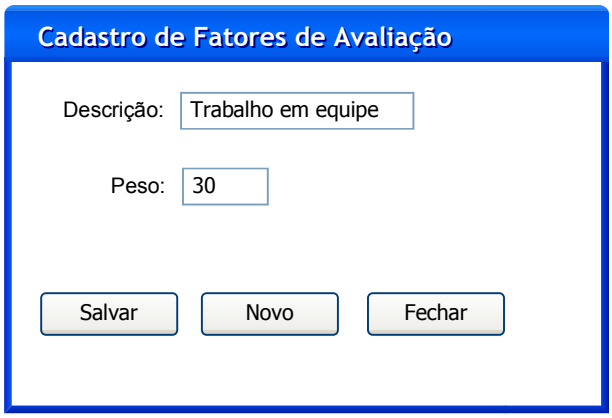

Figura 9 - ITE002 – Cadastro de Fatores de Avaliação

A regra de negócio RN01 que deve ser implementada no passo dois contêm a seguinte descrição textual:

- O sistema só deve permitir cadastro de pesos com valor acima de 20.

Assim, a exceção EX01 que pode ser levantada no passo dois do fluxo de inclusão dos Fatores de Avaliação tem a mesma estrutura da exceção descrita no fluxo de Inclusão de Parâmetros da Avaliação. Da mesma forma, a mensagem MSG003 que a exceção deve exibir pode ser reusada bem como todo o cenário de EX01. Essa análise leva a mover as mensagens MSG001 e MSG003 da lista de mensagem do caso de uso para uma lista de mensagens globais. Ação que reduzirá o esforço de

manutenção dos casos de uso e promoverá a reutilização até no nível de desenvolvimento.

4.3. Reutilização da especificação de mensagens

Normalmente, as mensagens de um sistema são exibidas em ocorrências similares. Por exemplo, elas são exibidas quando ações sobre o sistema não são suportadas ou derivadas de problemas de processamento. Também podem ser acionadas mensagens que trazem informações de erros e alertas, ou ainda mensagens de integração com recursos indisponíveis. A figura 10 mostra uma pequena suíte de mensagens na biblioteca de análise criada para ser reutilizada no decorrer desta monografia.

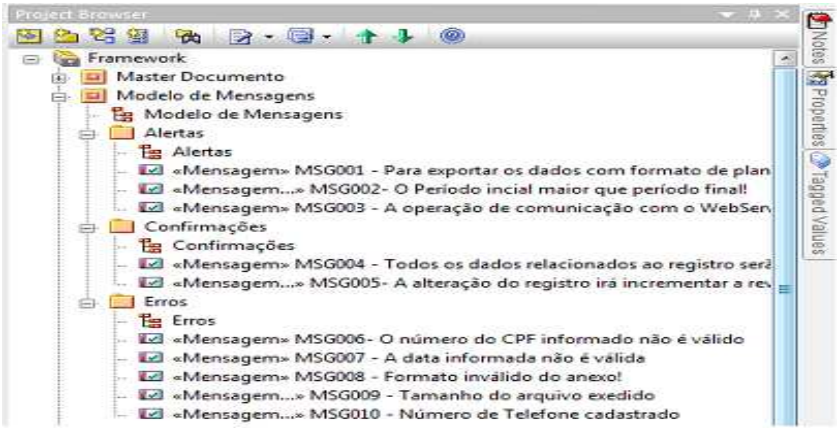

Figura 10 - Suite de mensagens do framework de análise

A estrutura da suíte deve proporcionar a localização rápida e o crescimento escalável dos artefatos em análise. Para isso, um diagrama central agrupa as mensagens na biblioteca de análise possibilitando a identificação de uma mensagem e evitando sua duplicidade. No caso das mensagens, foi criado um pacote agrupando-as por tipos.

A partir da biblioteca de análise as mensagens podem ser reusadas através da funcionalidade de exportação da suíte para outros projetos com um mínimo de esforço. A ferramenta EA permite a importação completa ou parcial de mensagens permitindo assim que a especificação das mensagens seja reusada sem a necessidade de reescrever para outros sistemas.

4.4. Reutilização de Casos de Uso

A UML possibilita, através do conceito de *extends* e *includes,* empregar o conceito de herança e sobreposição da programação orientada a objetos. Por este conceito, uma classe pode herdar os atributos e métodos de outra além de poder estender ou suprimir suas características. A partir da utilização correta de uma ferramenta CASE pode-se empregar a reutilização da análise e especificação dos casos de uso e acelerar esta fase.

Exemplificando, temos que estejam sendo levantados os requisitos dos cadastros básicos em um sistema. Na primeira análise os casos de uso identificados são mostrados no diagrama da figura 11. Após uma análise mais elaborada, é possível identificar que os casos de uso UC001, UC002 e UC003 têm uma funcionalidade comum. Isto é, todos eles podem ter uma lista com seus números de telefone. Após essa análise, o diagrama da figura 11 passa a ser apresentado como na figura 12.

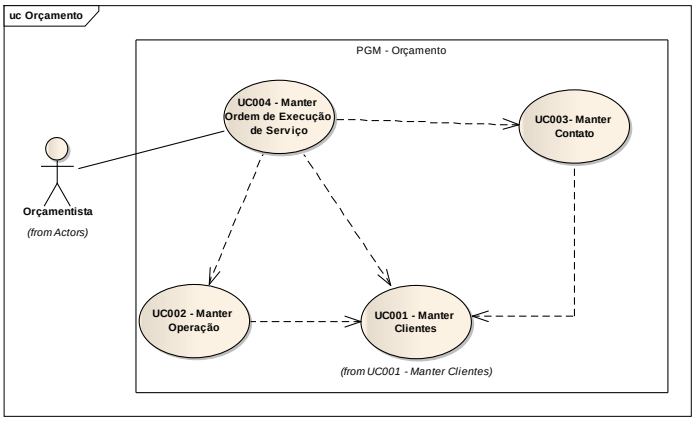

Figura 11 - Contexto de modulo

Desta forma, com uma análise voltada para a reusabilidade, podem ser identificadas funcionalidades que serão utilizadas não somente para esse módulo como também em outras partes deste sistema ou até mesmo em outros sistemas. Ao isolar as particularidades de uso da funcionalidade do lado do caso de uso extensor e agrupando-as no caso de uso estendido, pode-se elevar o caso de uso para um alto nível de abstração e reutilização.

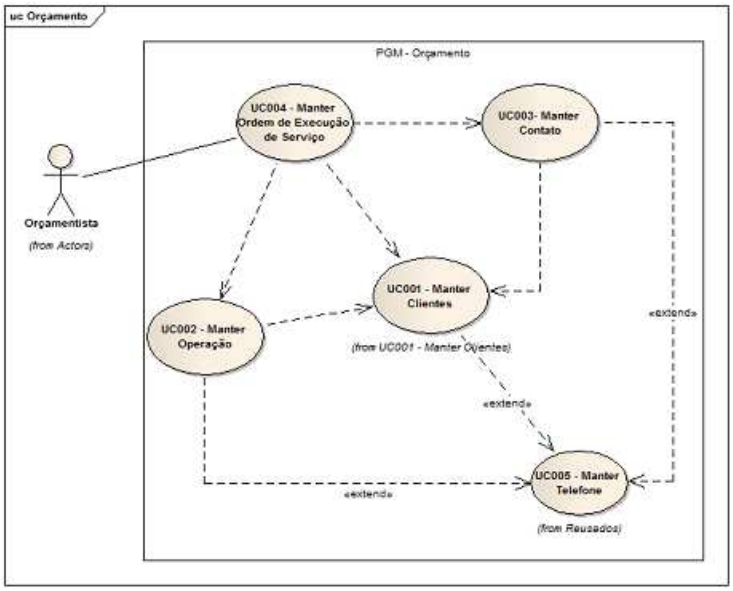

Figura 12- Análise remodelada

Uma vez identificada a possibilidade de reutilizar, o passo seguinte deve ser o de descrever o caso de uso e confirmar a sua capacidade de atender a diferentes sistemas com um baixo grau de adequação. Uma vez cumprido tal requisito, deve-se disponibilizá-lo na suíte de central ou biblioteca de análise. Isto é facilmente realizado através da funcionalidade de importação e exportação do EA.

Seguindo uma lógica para que a suíte se torne realmente utilizável, outros artefatos devem ser inseridos na biblioteca de análise. Exemplos desses artefatos são as interfaces que o caso de uso implementa, as suas regras e as mensagens relacionadas. O caso de uso reusado deve ser revisado e calibrado, permitindo aos analistas e desenvolvedores aumentar a capacidade de produzi-lo com qualidade e assertividade, o que concorre de forma positiva à capacidade de reusar as métricas deste caso de uso.

4.5. Reutilização das Estimativas

Um dos maiores problemas encontrados no processo de desenvolvimento ainda é a diferença encontrada na estimativa inicial do projeto e o valor final do produto de software. Com a prática de reutilização da análise dos requisitos pode ser reduzida esta diferença proporcionando ajustes no valor do desenvolvimento de artefatos já

utilizados. Ao ser estimado o desenvolvimento de um caso de uso e depois aferido o esforço real, pode ser calibrado os valores na biblioteca de análise deixando a métrica de desenvolvimento mais próxima da realidade possível, tornando a biblioteca de análise cada vez mais completa.

Um caso de uso já implementado e calibrado deve ter o seu custo de desenvolvimento mais assertivo do que um caso de uso em análise, mesmo tendo novas funcionalidades desenvolvidas. De forma geral, os artefatos da biblioteca de análise já estão metrificados e com seus prazos para implementação mais próximo do real. Com isso, todo o processo de desenvolvimento é beneficiado permitindo a equipe focar na qualidade e produtividade.

4.6. Reutilização da Inteligência do Negócio

Dentro das fábricas de desenvolvimento de software, as dificuldades no desenvolvimento são acentuadas com o fato de que as equipes normalmente são distribuídas e muitas vezes geograficamente separadas. Como o negócio dos clientes está em constante mudança e normalmente não é bem documentada, a manutenção dos artefatos da análise fica na maioria das vezes desatualizada. Assim, a maior parte do conhecimento sobre o negócio fica concentrada com os analistas e no produto.

O uso da UML, de ferramentas CASE e a implementação de políticas para promover a reutilização tende a reduzir a manutenção e ao mesmo tempo registrar os objetivos de negócio. Esta prática permite também o registro das particularidades das corporações. A dificuldade de abstrair os aspectos e necessidades comerciais dos clientes, mesmo com as similaridades de negócio, envolve aspectos como: diversidade de linguagens de programação, arquiteturas legadas e variadas, necessidade de integração entre sistemas.

Para os negócios muitas vezes similares, pode ser reduzido o tempo da análise, manutenção, adaptação e acompanhamento com o uso corporativo da UML apoiada por uma ferramenta CASE. Tal observação é válida tanto para sistemas antigos como para novas demandas. Definir os requisitos não funcionais para a corporação,

ao invés de defini-los para os sistemas, pode proporcionar que os outros sistemas sejam contemplados com tais requisitos. Reusando os requisitos adicionados a biblioteca de análise à medida que sejam necessários.

5. CONCLUSÕES

Com base no estudo feito para a elaboração deste trabalho e considerando o conhecimento pessoal adquirido por meio de experiências de trabalho e pesquisas aleatórias, tornou-se possível concluir a relevância do tema e a importância da reutilização na fase de análise de requisitos. Em particular, este trabalho discute as particularidades e vantagens da implementação de uma biblioteca de reutilizável nessa fase preliminar de análise.

Outro aspecto que se destaca neste estudo é o de que uma das formas de permitir que a análise cubra às particularidades do negócio de um cliente é reduzir o esforço necessário para entender e transmitir a equipe de desenvolvimento o que precisa ser feito. Isso pode ser alcançado implementando a biblioteca de análise como apresentada neste trabalho.

Observa-se, ainda, que a construção de uma biblioteca de análise dá como resultado à equipe de desenvolvimento, uma ferramenta capaz impulsionar o projeto de software e permitir uma maior atenção aos aspectos relevantes do negócio que esse software irá apoiar. Em particular, os erros da fase de análise de requisitos, são caros quando encontrados tardiamente no projeto. Entretanto, são os artefatos produzidos nesta fase que guiará as demais etapas do desenvolvimento.

No que diz respeito à metodologia para implantar a biblioteca de análise, há que ser bem planejada a fim de obter o efeito satisfatório e incremental. Assim, o ideal é que a metodologia seja guiada por politicas e normas institucionais e controlada pela gerência de configuração.

Enfim, está claro que a reutilização de sistemas pode ganhar um impulso determinante com a reutilização da fase de análise, permitindo o uso em outros artefatos do produto seja no desenvolvimento, integração ou testes. Na prática, os problemas terão uma solução no nível da análise e irão permitir a reutilização da especificação de sistemas. A tendência é que, pela reutilização na engenharia de requisitos, seja demandado menor esforço para desenvolver as soluções necessárias para atender os clientes.

6. REFERÊNCIAS

[1] SOMMERVILLE IAN. Engenharia de Software. 8ª ed. São Paulo: Addison Wesley, 2007.

[2] G. BOOCH, J. RUMBAUGH, I. JACOBSON. UML, Guia do Usuário. Editora Campus, 2000.

[3] BEZERRA, Eduardo. Princípios de Análise e Projetos de Sistemas 2ª ed.: Campus/Elsevier,2006.

[4] THAYER, R. H. e DORFMAN, M.; “Introduction to Tutorial Software Requirements Enginnering” in Software Requirements Engineering, IEEE-CS Press, Second Edition, 1997.

[5] SIDDIQI, Jawed. “Requirements Engineering: the emerging wisdow” in Software Requirements Engineering, IEEE-CS Press, Second Edition, 1997.

[6] DAMIAN, Adrian, et al. “Joint Application Development and Participatory Design” 1997.<http://www.cpsc.ucalgary.ca/~pand/seng/613/report.html>

[7] KOTONYA, Gerald e SOMMERVILLE, Ian. Requirements Engineering Processes e Techniques. John Wiley and Sons, 1998

[8] McCONNEL, Steve. “Software Project Survival Guide: How to Be Sure Your First Important Project isn’t Your Last. 1998.

[9] LARMAN, C. Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos e ao Processo Unificado. Trad. Luiz Augusto Meirelles Salgado e João Tortello. 2 ed. Porto Alegre: Bookman, 2004.

[10] GRADY, Robert. Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall, 1992.

[11] "Introduction To OMG's Unified Modeling Language" Last updated on 12/04/2012 Disponível em:<http://www.omg.org/gettingstarted/what_is_uml.htm> acesso em:20 out. 2012.