UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

UM MÉTODO PARA ELICITAÇÃO E MODELAGEM DE REQUISITOS BASEADO EM OBJETIVOS

MÁRCIA CRISTINA FERRO CARVALHO

São Luis

2001

UM MÉTODO PARA ELICITAÇÃO E MODELAGEM DE REQUISITOS BASEADO EM OBJETIVOS

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de mestre em Ciência da Computação.

Por

MÁRCIA CRISTINA FERRO CARVALHO

Novembro, 2001

UM MÉTODO PARA ELICITAÇÃO E MODELAGEM DE REQUISITOS BASEADO EM OBJETIVOS

MÁRCIA CRISTINA FERRO CARVALHO

DISSERTAÇÃO APPROVADA EM __ / __ / 01

_____

Prof. Dr. Zair Abdelouahab
(Orientador)

_____

Prof. Dr. Luiz Marcio Cysneiros
(Membro da Banca Examinadora)

_____

Profª. Dra. Maria del Rosário Girardi
(Membro da Banca Examinadora)

UM MÉTODO PARA ELICITAÇÃO E MODELAGEM DE REQUISITOS BASEADO EM OBJETIVOS

MESTRADO

Área de Concentração: CIÊNCIA DA COMPUTAÇÃO

MÁRCIA CRISTINA FERRO CARVALHO

Orientador: Dr. Zair Abdelouahab

Programa de Pós-Graduação

Em Engenharia de Eletricidade da

Universidade Federal do Maranhão

DEDICATÓRIA

Às pessoas interessadas na área de Engenharia de Requisitos.

AGRADECIMENTOS

A DEUS, pela superação dos momentos de desânimo e angústia.

A minha mãe, pela confiança, estímulo e apoio.

A minha irmã, pelas orações, auxílio e amor.

Ao Prof. Dr. Zair Abdelouahab, pela orientação durante a realização desse trabalho.

Ao mestrando Luís Carlos Fonseca, pela colaboração.

“O mais importante que o conhecimento é a imaginação”

Albert Einstein

RESUMO

Este trabalho propõe uma integração da abordagem CREWS L'ecritoire baseado em cenários com a abordagem de casos de usos descrita por Regnell et al e o Método GBRAM baseado em objetivos.. Dessa forma são adicionados ao trabalho de Regnell et al a noção de pedaço de requerimento (RC), as estratégias de descoberta do objetivo através dos relacionamentos AND, OR e de refinamento entre RCs, além de extender o modelo com um nível físico onde são descritas as ações internas do sistema.

Em contrapartida, é adicionado à abordagem CREWS- L' Ecritoire o acoplamento objetivo-caso de uso onde a obtenção dos cenários origina-se da aplicação de uma estratégia de refinamento sobre os casos de uso, a integração de cenários normais e excepcionais em um modelo de uso sintetizado, com uma visão única por ator.

No que concerne ao Método GBRAM, foram extendidas as relações de dependência de contrato entre objetivos para dependência alternativa, composicional, dependência condicional exclusiva e inclusiva, assim como foi introduzida a técnica de caso de uso, conforme descreve Regnell et al extendida a noção de cenários para caracterizar situações não só excepcionais, mas também as normais e variacionais.

Com a integração desses métodos, o método resultante irá representar um adicionamento de cada um isoladamente. Esse método se propõe a acoplar objetivos e casos de uso numa decomposição top-down em diferentes níveis de abstração. Ele combina a utilização destes dois conceitos em um pedaço de requerimento (RC). Um RC é um par <objetivo, caso de uso>, relacionados através de relacionamentos AND, OR e de refinamento. Esses três tipos de relacionamentos entre RCs conduzem a uma organização hierárquica de RCs em três níveis de abstração: o funcional, o comportamental e o físico.

Palavras-chaves: Casos de uso, Cenários, Objetivos, Requisitos, Elicitação, Modelagem

ABSTRACT

This work proposes an integration of the approach CREWS L'ecritoire based on scenarios with the approach of use cases described by Regnell & al and the Method GBRAM. In this case, the work of Regnell is extended with the notion of Requirement Chunk (RC), strategies for discovering objectives through the relationships AND, OR, and refinement between RCs, as well as extending the model with a physical level where the internal actions of the system are described.

On the other side, the approach CREWS – L'ecritoire is extended with coupling of objective and use case where the scenarios are obtained by applying a strategy of refining use cases, and also with an integration of normal and exceptional scenarios in one synthetic model of use with one unique vision for each actor.

The Method GBRAM is extended with the alternative contract dependency, composicional contract dependency, inclusive and exclusive contract dependency, as well it introduzed the use case technique, described by Regnell et al and extended with the notion of scenarios for caracterize exceptionals, variationals and normal situations.

With the integration of threes methods, the new method which benefit from the advantages of the approaches of CREWS-L'ecritoire, use cases.and goals. The new method proposes a coupling between objective and Use Cases in a top-down decomposition of different levels of abstraction. The method combines the two concepts into a Requirement Chunk (RC). An RC is a pair of <objective, Use Case> related with AND, OR and refinement. These three types of relations between RCs lead to a hierarchic organisation of RCs in three levels of abstractions: functional, behaviour, and physical.

Keywords: Use Cases, Scenarios, Requirements, Elicitation, Modelling

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIATURAS E SÍMBOLOS

UCDA – Análise Dirigida a Caso de Uso.
UORE – Engenharia de requisitos orientada a uso.
OAI – Objetos Abstratos de Interface.
UCS – Especificação de casos de uso.
CAU – Cenário Abstrato de uso.
SUM – Modelo de uso sintetizado.
PUCT – Tabela de caso de uso parcial.
UCT – Árvores de casos de uso.
RC – Pedaço de requisito.
LEL – Léxico Extendido da Linguagem.
ER – Entidade Relacionamento.
BMV - Visão do Modelo Básico.
GBRAM - Método de Análise de Requisitos baseados em objetivos.
DRS – Documento de Requisitos de software.

1. INTRODUÇÃO

1.1 DESCRIÇÃO DO PROBLEMA

A Engenharia de Software é o processo de desenvolvimento de um sistema baseado em computador. Seu principal objetivo é a construção de um sistema que reflita o negócio da empresa e que atenda às necessidades e expectativas de seus usuários finais.

Um modelo de desenvolvimento de software, de acordo com [28] é composto de um conjunto de atividades: obtenção de requisitos, análise, projeto, construção, teste, desenvolvimento e manutenção. A direção e a qualidade do sistema em computador resultante dependerão da ênfase que é oferecida a cada fase.

Entre as diversas fases do ciclo de vida do desenvolvimento de um software, a engenharia de requisitos é uma das fases mais importantes e objetiva sistematizar o processo de definição de requisitos, gerando especificações que descrevam de forma não ambígua, consistente e completa o comportamento do universo do domínio de um problema.

Para isso, é necessário o correto entendimento do contexto do sistema, ou seja, delimitar onde realizar essa tarefa, quais os recursos disponíveis e quais os limites do sistema, assim como captar as necessidades dos vários usuários de um sistema e traduzir essas necessidades na forma de requisitos.

Os requisitos representam algo que uma aplicação de computador deve fazer para seus usuários, constituindo-se em um escopo de um projeto de desenvolvimento de software. Os requisitos podem ser classificados em funcionais e não funcionais. Os requisitos funcionais, de acordo com [47,48] descrevem os aspectos comportamentais de um sistema. Os requisitos não funcionais descrevem os aspectos não comportamentais do sistema, capturando as propriedades e restrições sob as quais um sistema deve operar. Como tipos de requisitos não funcionais estão a segurança, garantia, confiabilidade e usabilidade.

Entre os diversos desafios que a obtenção de requisitos enfrenta, um deles consiste em encontrar o que os usuários precisam, já que na maioria das vezes, eles não sabem o que eles querem ou mesmo o que esperar da potencialidade das máquinas. Os usuários são

cientes de suas atividades e responsabilidades diárias, mas não sabem até que ponto a máquina irá auxiliá-los.

A obtenção de requisitos e restrições dos sistemas a serem desenvolvidos advém de entrevistas com usuários e especialistas, leitura da documentação já existente e de observações, entre outras técnicas [23]. Dessa forma, o perfil dos usuários e dos desenvolvedores são de fundamental importância para um processo bem sucedido. No entanto, o que se verifica é a sua inabilitação para articular seus requisitos, sem conhecimento do que é tecnologicamente capaz de ser realizado. Observa-se, ainda que os mesmos freqüentemente não querem comunicar os requisitos, dificultando dessa forma o trabalho do engenheiro de requisitos.

Clientes e engenheiros de software, por vezes, não falam a mesma linguagem, surgindo mal-entendidos entre os mesmos, oriundos da comunicação difícil, onde o cliente não consegue expressar adequadamente conhecimento sobre a aplicação e, por outro lado, o engenheiro não conhece o domínio da aplicação e, muitas vezes não possui as habilidades necessárias para obter os requisitos dos usuários ou são inabilitados para desenvolver um entendimento das necessidades dos usuários, induzindo, por vezes os usuários a concordarem com os seus requisitos, originando assim em omissão de informações importantes, ambigüidade na definição de requisitos do sistema e interpretações errôneas [40].

De modo a obter requisitos consistentes e completos, faz-se necessário que sejam resolvidos os conflitos e eliminada a redundância, ocasionando assim uma redução do volume de especificação de requisitos. Durante a definição de requisitos, um dos processos difíceis consiste em documentar as necessidades dos usuários e realizar sua confirmação, verificando se esses requisitos identificados não já foram documentados de uma maneira diferente ou de uma maneira conflitante, bem como identificar o que está faltando.

Devido à não participação dos usuários em revisões da definição de requisitos para confirmarem um conjunto de requisitos escritos, estes mesmos usuários, posteriormente insistem em novos requisitos após o custo e o escalonamento terem sido fixados.

Outro aspecto a ser destacado concerne à dificuldade de acomodação das mudanças surgidas durante e após a análise onde o engenheiro é incumbido de efetuar uma atualização constante do documento de requisitos diante da evolução dos requisitos do

sistema. Entretanto não só os requisitos evoluem naturalmente, mas também os cenários. Em [8] é mostrado um modelo e uma framework para evolução de cenários.

Destaca-se ainda no processo de Engenharia de requisitos a consideração de diversos pontos de vista [48], a partir de vários agentes ou usuários, uma vez que nenhum usuário possui todas as respostas, que advém de muitas fontes de informação.

A Engenharia de requisitos, embora gaste apenas 6% do esforço total durante todo o ciclo de vida, quando realizada de forma ineficiente pode vir a gerar inúmeros problemas durante o desenvolvimento de um software. Entre os problemas mais comuns que ocorrem nesta fase destacam-se, de acordo com [44]:

- A maioria dos projetos tem requisitos adicionais descobertos após a aprovação do documento final gerado nesta fase pelo engenheiro de requisitos.
- Muitos projetos acabam por repetir a fase de engenharia de requisitos.
- Os erros encontrados depois da fase de engenharia de requisitos necessitam de muito mais esforço para correção.

As conseqüências de erros em requisitos são severas e de alto custo. Em sistemas complexos, 95% do código tem que ser reescrito de maneira a satisfazer mudanças nos requisitos dos usuários.

Em [44], reporta-se que 12% dos erros descobertos em um sistema de software durante um período de três anos foram devido a erros nos requisitos de sistemas originais. O custo da correção pode ser 1/3 do custo da produção total. Se um erro nos requisitos de software é detectado e corrigido durante o estágio de análise do processo de desenvolvimento, o erro é relativamente simples na hora de corrigir, uma vez que ele envolve apenas uma correção na especificação de requisitos.

Todavia, se um erro no requisito não é corrigido até o estágio de manutenção, um inventário muito maior de artefatos é afetado, como por exemplo, especificações, código, assim como manuais de usuários e manutenção.

Existem estimativas de que 47% do tempo devotado às atividades de manutenção se deve às tarefas de correção, enquanto 62% é dedicado às atividades de compreensão [44].

Dessa forma, os custos de manutenção podem ser significativamente reduzidos se esforços são devotados ao aumento do nível de compreensão dentro de especificações do sistema. Assim, é crítico garantir que requisitos sejam identificados tão cedo quanto possível e que medidas sejam tomadas para prevenir quaisquer requisitos de serem negligenciados.

À medida que o ciclo de vida progride, o custo de reparar erros feitos na especificação de requisitos aumenta significantemente. Por conseguinte, é de fundamental importância um entendimento mais claro dos requisitos mais brevemente nos estágios de planejamento do processo de desenvolvimento.

Com o propósito de combinar algumas características das abordagens baseadas em objetivos, casos de uso e cenários, um estudo comparativo será realizado de maneira a abstrair as principais vantagens e desvantagens destas várias abordagens.

Dentre os vários métodos que adotam casos de uso como técnica de elicitação e modelagem de requisitos [14,17,27, 42,43] destacamos o trabalho de Regnell et al, descrito em [42,43] que apresenta o Método UORE (Engenharia de requisitos baseada em uso) e o modelo hierárquico de caso de uso. Por outro lado, entre os métodos que utilizam cenários [25, 35, 39, 45] a abordagem CREWS- L´Ecritoire [45] emprega o acoplamento cenário-objetivo na descoberta de objetivos e obtenção de cenários. E, dentre as abordagens que utilizam objetivos [4,6,18,24, 49] destacamos o Método GBRAM [4].

Este trabalho propõe uma integração da abordagem CREWS-L´Ecritoire baseado em cenários com a abordagem de casos de uso descritas por Regnell et al e o Método GBRAM baseado em objetivos. Dessa forma são adicionados ao trabalho de Regnell et al a noção de pedaço de requerimento (RC), as estratégias de descoberta do objetivo através dos relacionamentos AND, OR e de refinamento entre RCs, além de extender o modelo com um nível físico onde são descritas as ações internas do sistema.

Em contrapartida, é adicionado à abordagem CREWS- L' Ecritoire o acoplamento objetivo-caso de uso onde a obtenção dos cenários origina-se da aplicação de

uma estratégia de refinamento sobre os casos de uso, a integração de cenários normais e excepcionais em um modelo de uso sintetizado, com uma visão única por ator.

No que concerne ao Método GBRAM, foram extendidas as relações de dependência de contrato entre objetivos para dependência alternativa e dependência condicional exclusiva, assim como foi introduzida a técnica de caso de uso, conforme descreve Regnell et al em [42,43] e extendida a noção de cenários para caracterizar situações não só excepcionais, mas também as normais e variacionais.

Além disso, de maneira a facilitar e sistematizar o processo de obtenção e representação desses requisitos, será desenvolvido um método para elicitação e modelagem de requisitos, no qual procura-se mesclar casos de uso, cenários e objetivos, que se constituem nas principais técnicas utilizadas para essa tarefa.

De forma a agilizar a aplicação do método proposto, será desenvolvido também o protótipo de uma ferramenta, onde serão automatizadas algumas fases do processo.

1.2 OBJETIVOS

Este trabalho procura:

- ✓ Investigar as abordagens existentes baseadas em casos de uso, cenários e objetivos.
- ✓ Realizar um estudo comparativo entre os diversos métodos existentes no que concerne às fases de elicitação e modelagem de requisitos.
- ✓ Propor um método para elicitação e modelagem de requisitos que combine essas técnicas e aplicá-lo em um estudo de caso.
- ✓ Implementar o protótipo da ferramenta que suporte as diversas fases do método proposto.

1.3 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está organizado em seis capítulos. O primeiro capítulo apresenta a introdução ao tema, descrevendo o problema a ser estudado, os objetivos e a estrutura do trabalho.

O segundo capítulo ilustra o estado da arte, considerando a fase de engenharia de requisitos de vários métodos que utilizam casos de uso, cenários e objetivos para elicitação e modelagem dos requisitos.

Com o propósito de facilitar e sistematizar o processo de obtenção e representação dos requisitos, foi criado no capítulo três, um método para elicitação e modelagem de requisitos, onde são integradas as técnicas de casos de uso, cenários e objetivos, as predominantemente utilizadas nessa tarefa.

O quarto capítulo se destina à aplicação do método proposto em um estudo de caso para elicitar os requisitos de um restaurante.

O quinto capítulo apresenta a implementação do protótipo de uma ferramenta de suporte à aplicação do método proposto.

Por último, o sexto capítulo faz as considerações finais, relatando as contribuições desse trabalho e as perspectivas de trabalhos futuros.

2. ESTADO DA ARTE

Este capítulo apresenta diversas abordagens que utilizam casos de uso, cenários e objetivos na fase de engenharia de requisitos. Na seção 2.1 são apresentadas algumas abordagens que utilizam casos de uso como técnica de elicitação e modelagem de requisitos [14,17, 27, 42,43]. Em [26] pode ser encontrado um resumo das principais abordagens que utilizam casos de uso. Na seção 2.2 são ilustradas algumas abordagens que se baseiam em cenários para elicitar e modelar os requisitos [25, 34, 30, 36]. Na seção 2.3 apresentam-se abordagens que utilizam objetivos para a fase de engenharia de requisitos [4,6,18,24, 29,30,49]. No final de cada seção é realizado um estudo comparativo entre as abordagens levantadas.

2.1 Abordagens baseadas em casos de uso como técnica de elicitação e modelagem de requisitos

2.1.1 Engenharia de requisitos do Objectory

O Objectory é um método orientado a objeto proposto por Jacobson para desenvolvimento de sistemas, que consiste na especificação de requisitos, análise, projeto, implementação e testes [27].

O modelo de requisitos ajuda a delimitar o sistema e a definir a funcionalidade que este deve oferecer. Ele irá governar o desenvolvimento de todos os outros modelos, sendo estruturado pelo modelo de análise, realizado pelo modelo de projeto, implementado pelo modelo de implementação e testado pelo modelo de testes.

O modelo de requisitos consiste em três partes: o modelo de caso de uso, o modelo de objeto de domínio do problema e as descrições de interface do usuário. A seguir será descrita cada uma dessas partes em mais detalhes.

1 – Modelo de caso de uso

O modelo de caso de uso utiliza atores e casos de uso visando definir o que existe externamente ao sistema (atores) e o que deve ser executado pelo sistema (casos de uso). Atores interagem com o sistema e representam um tipo de usuário ou categoria,

definindo os papéis que os usuários podem ocupar, sendo que estes podem ser usuários humanos ou outros sistemas.

Os atores são divididos em dois grupos: atores primários e atores secundários. Os atores primários são usuários diretos do sistema projetado, e governam a estrutura do sistema, executando uma ou algumas de suas principais tarefas. Atores secundários supervisionam e mantém o sistema. A razão desta divisão é que a estrutura do sistema deve ser decidida em termos da funcionalidade principal.

Um caso de uso é uma maneira específica de usar o sistema executando alguma parte da funcionalidade. Ele consiste em um diálogo (ou seja, uma seqüência de transações) entre o sistema e um usuário. Normalmente um caso de uso possui um curso básico de eventos e alguns cursos alternativos para representar erros ou variantes desse curso básico.

Outro conceito referenciado em [27] é a noção de caso de uso concreto e caso de uso abstrato. Casos de uso abstratos descrevem partes que são comuns aos outros casos de uso e que não podem ser instanciados. Por outro lado, casos de uso concretos referem-se aos casos de uso que podem ser instanciados.

Para relacioná-los, são definidos os construtores *extends* e *uses*. O primeiro é utilizado para modelar partes opcionais dos casos de uso, partes complexas e alternativas, e alguns subcursos que são executados apenas em determinados casos. Este construtor permite ao modelador inserir um caso de uso em outro caso de uso, extendendo, dessa forma o segundo caso de uso. Já o construtor *uses* refere-se à utilização de um caso de uso de forma completa por outro caso de uso. Ele é utilizado quando dois ou mais casos de uso tem comportamento comum.

2 - Modelo de objetos do domínio do problema

Serve como um suporte ao desenvolvimento do modelo de requisitos, oferecendo uma visão lógica do sistema utilizando objetos do domínio do problema. Esses objetos têm uma influência direta no ambiente da aplicação e o sistema deve ter conhecimento, permitindo seu melhor entendimento, além de apoiar na formulação de descrições de casos de uso.

O modelo de domínio do problema pode ser utilizado para diferentes propósitos. Ele oferece suporte à formulação de descrições de casos de uso, e no projeto de

interface homem-máquina, sendo também direcionado à obtenção de um melhor entendimento do sistema, focalizando mais nos objetos do domínio do problema.

O modelo é aplicado quando há dificuldade em se definir a tarefa do sistema especialmente os seus limites, em virtude da especificação de requisitos ser obtida de fontes muito vagas.

Este modelo apóia o desenvolvimento de uma lista de substantivos que serve de suporte quando se especificam os casos de uso. Deste modelo podem ser definidos conceitos que o sistema deve trabalhar, constituindo um glossário que pode ser utilizado para formular a funcionalidade dos casos de uso.

3 -Descrições de interface

Após a descrição dos casos de uso, é importante que elas sejam acompanhadas por descrições de interface, de modo a especificar em detalhes o que a interface do usuário parecerá quando os casos de uso são executados. Outrossim, é relevante o envolvimento do usuário, refletindo sua visão lógica sobre o sistema.

Na descrição das interfaces, procura-se apresentar aos usuários potenciais uma simulação do que ele deverá visualizar na tela, reduzindo-se a possibilidade de má interpretação de requisitos, além de procurar manter a consistência entre a imagem conceitual do usuário sobre o sistema e o seu comportamento real.

A Análise Dirigida a Caso de Uso proposta por Jacobson apresenta como vantagens a simplicidade e a participação ativa dos clientes e usuários finais na análise dos requisitos, uma vez que a descrição dos casos de uso são baseadas em conceitos naturais que podem ser encontrados no domínio do problema. Além disso, diferentes casos de uso podem ser identificados e analisados independentemente focalizando em aspectos limitados do uso do sistema por vez.

Como desvantagens enumeraremos inicialmente a falta de síntese da técnica, na forma proposta por Jacobson, que produz uma coleção solta de casos de uso. Além disso, o modelo de caso de uso oferece uma aplicabilidade limitada como modelo de referência para validação e verificação, uma vez que não pode ser utilizado para geração automática de casos de teste. Outro aspecto confuso refere-se à falta de uma definição clara da semântica de casos de uso e à ausência de diretrizes de como casos de uso devem ser descritos. Também não é

claro quais tipos de eventos devem ser tratados na descrição dos casos de uso: estímulos-resposta externos ou atividades do sistema interno também.

Embora cada caso de uso seja associado a um ator específico, as descrições de casos de uso permitem que muitos atores estejam envolvidos. Outrossim, em sistemas complexos podem ocorrer inconsistências e contradições entre os casos de uso, em virtude de sua produção ser feita de forma independente e expressar objetivos de diferentes atores.

De forma geral, a Análise Dirigida a Caso de Uso (UCDA) não endereça completamente os seguintes aspectos:

- Os Casos de uso não são independentes. Eles podem se sobrepor, ocorrer simultaneamente ou influenciar um ao outro.
- Os Casos de uso ocorrem sob condições específicas. Eles têm contextos específicos para serem iniciados e finalizados.
- O nível de abstração dos casos de uso e seu tamanho são de escolha arbitrária;
- Os casos de uso podem, na prática, garantir apenas cobertura parcial de todas as possibilidades de uso do sistema.

2.1.2 Método UORE (Engenharia de requisitos orientada a uso)

Em [42] é proposto um método de Engenharia de requisitos (UORE - Usage-Oriented Requirements Engineering) como uma extensão da Análise Dirigida a Caso de Uso [27], tendo como resultado um modelo de uso sintetizado.

O intuito desse método é melhorar Análise Dirigida a Caso de Uso original, extendo-o com uma fase de síntese onde casos de uso separados são integrados em um Modelo de Uso Sintetizado (SUM), que captura requisitos funcionais e aspectos de uso do sistema.

O processo do Método UORE consiste de duas fases: análise e síntese, como mostrado na figura 2.1, executadas iterativamente até a criação do modelo de casos de uso do UORE. A fase de análise do UORE consiste de duas atividades interrelacionadas, a saber: identificação de atores e casos de uso e unificação da terminologia.

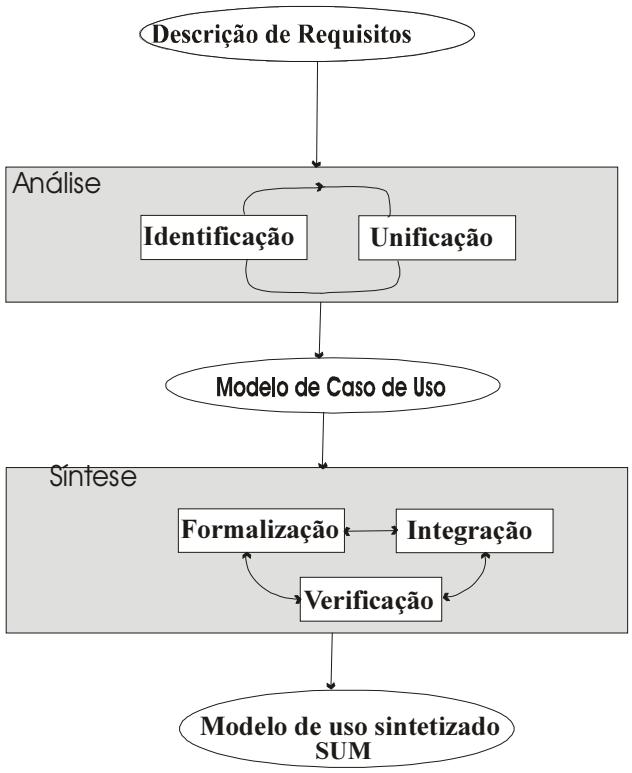

Figura 2.1: Processo UORE

A primeira atividade objetiva identificar e descrever atores e casos de uso, conforme ilustra um exemplo na figura 2.2. Já a segunda atividade unifica a terminologia dessas descrições. Para isso, os objetos do domínio do problema e seus atributos são identificados e descritos em um dicionário de dados, focalizando entidades manipuladas pelos atores, operações externas ao sistema e elementos da interface com o usuário.

A fase de análise do UORE assemelha-se à versão UCDA (Análise Dirigida a Caso de Uso). No entanto, suas principais diferenças referem-se a semântica modificada de atores e casos de uso, a identificação de contextos de casos de uso, aplicação de uma visão única para cada ator, a unificação de terminologia e a realização de descrições estruturadas dos casos de uso.

A figura. 2.2 apresenta um exemplo da descrição de atores e casos de uso de uma máquina ATM, utilizando uma descrição textual estruturada. Inicialmente são ilustrados os atores, as condições de invocação, o fluxo de eventos e as condições de término dos casos de uso “Efetuar saque”, caso normal, “Efetuar saque”, quantidade inválida e “Controle de conta”, caso normal.

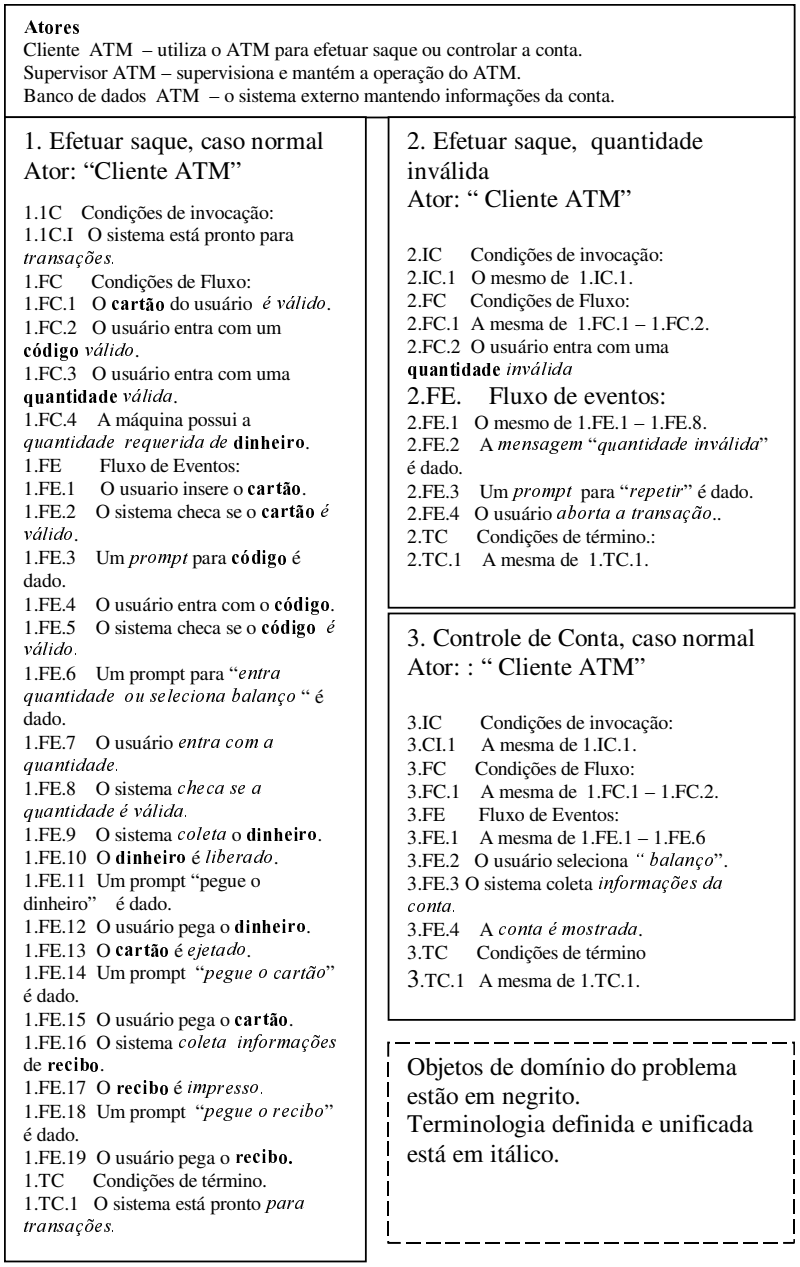

Figura. 2.2. Exemplo de Descrição do caso de uso.

A fase de síntese formaliza os casos de uso, integra-os e cria o Modelo de Uso Sintetizado, consistindo de três atividades, executadas de maneira iterativa:

1 - Formalização de casos de uso;

2 - Integração de casos de uso;

3 - Verificação.

- **Atividade de formalização**

A atividade de formalização destina-se a produzir especificações de casos de uso formais para cada caso de uso identificado na fase de análise. O produto desta atividade é uma coleção de especificações de casos de uso (UCS) representados em uma linguagem gráfica que expressa a ordem temporal de estímulos aos usuários, respostas do sistema e operações atômicas. Essa atividade engloba os seguintes passos:

1 - Identificação dos objetos abstratos de interface

O usuário nunca se comunica diretamente com o sistema. Uma interface está sempre envolvida nessa comunicação. As entidades que formam o ambiente da comunicação são chamadas de objetos abstratos de interface (OAI). A identificação das entidades que tomam parte na comunicação ator-sistema é conseguida pela análise de todos os casos de usos e da terminologia do domínio do problema.

2 - Identificação de operações atômicas:

São operações executadas pelo sistema e tem efeito nos usuários. Uma operação do sistema é atômica do ponto de vista de um ator, se ela não requer nenhuma comunicação com esse ator durante sua execução. Estas são identificadas focalizando-se nas operações do sistema que não requerem interação com o ator envolvido no caso de uso.

3 - Criação de uma especificação de caso de uso (UCS) para cada caso de uso.

Após a identificação de todos os objetos abstratos de interface e das operações atômicas, o fluxo de eventos de cada caso de uso é transformado em uma especificação que modela as relações temporais entre eles.

- **Atividade de Integração**

A atividade de integração se propõe a realizar a união de diferentes especificações de casos de uso e produzir um modelo de uso sintetizado (SUM). O SUM consiste em uma coleção de visões de uso, uma para cada ator. Esta atividade consiste nos seguintes passos:

1 - Identificação de ações do usuário e do sistema

Ao representar formalmente a utilização do sistema, faz-se necessário ter um mecanismo de abstração para representar o protocolo detalhado de interação durante as partes controladas pelo usuário e as partes controladas pelo sistema. Os termos "ações do usuário" são utilizados para protocolos onde o usuário está no controle, e “ações do sistema” para protocolos onde o sistema está no controle.

2 - Criação de cenários abstratos de uso

Cada especificação de caso de uso é transformada em um cenário abstrato de uso (CAU), desenhado como uma seqüência de ações do usuário e ações do sistema, interconectados por transições que representam as mensagens resultantes de cada ação, como demonstra um exemplo na figura 2.3.

O contexto de invocação e de término de um cenário abstrato de uso (CAU) é indicado por *labels*, que denota o estado interno do sistema, ou seja, um subconjunto do produto cartesiano de todos os estados dos objetos de interface abstrata.

3- Integração de cenários abstratos de uso.

O Modelo de Uso Sintetizado (SUM) resulta da união de todos os cenários abstratos de uso produzidos para um ator específico. O SUM consiste em uma visão de uso do sistema para um ator, e também contém descrições de atores, visões de uso, especificação de caso de uso, objetos abstratos de interface, ações do usuário e do sistema e, um dicionário de dados com objetos de domínio do problema. Um exemplo pode ser visto na figura 2.4.

Na figura. 2.3 é ilustrado um cenário abstrato de uso da máquina ATM, onde o usuário efetua saque. As ações do usuário são representadas por elipses e as ações do sistema, por retângulos. Este cenário inicia com o usuário inserindo cartão até à retirada do recibo pelo usuário.

Na figura. 2.4 é apresentado um exemplo do produto final da aplicação do Método UORE, o modelo de uso sintetizado (SUM) para o ator cliente com a unificação dos vários cenários abstratos de uso, para o mesmo exemplo do ATM.

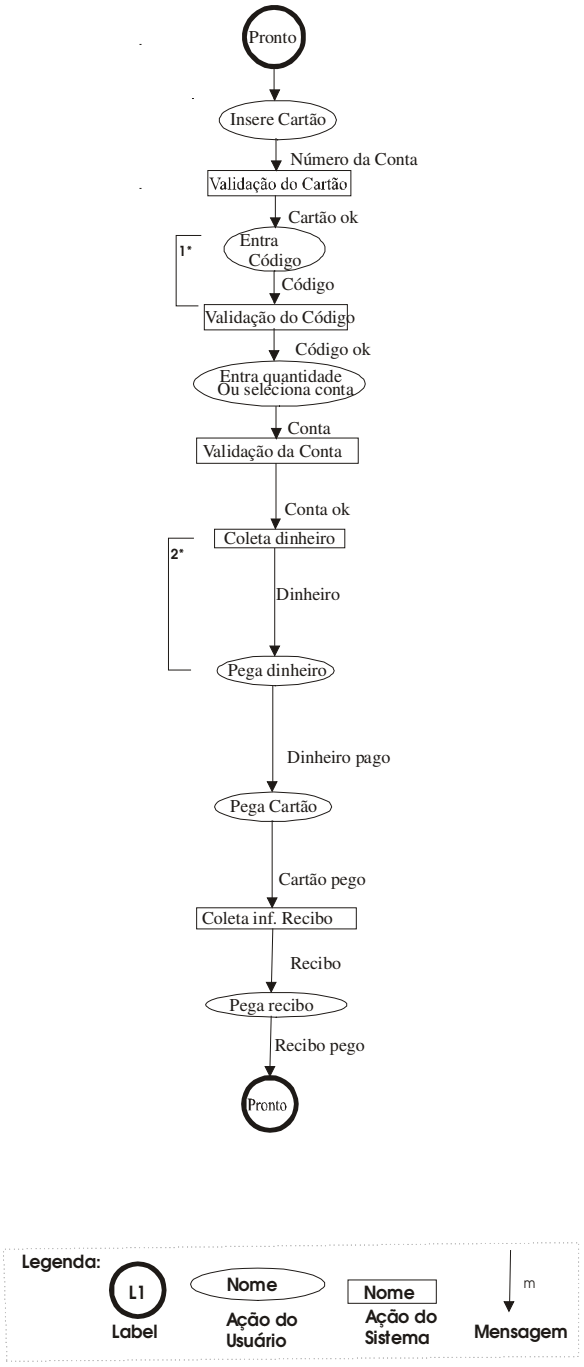

Figura 2.3: Exemplo de cenário abstrato de uso.

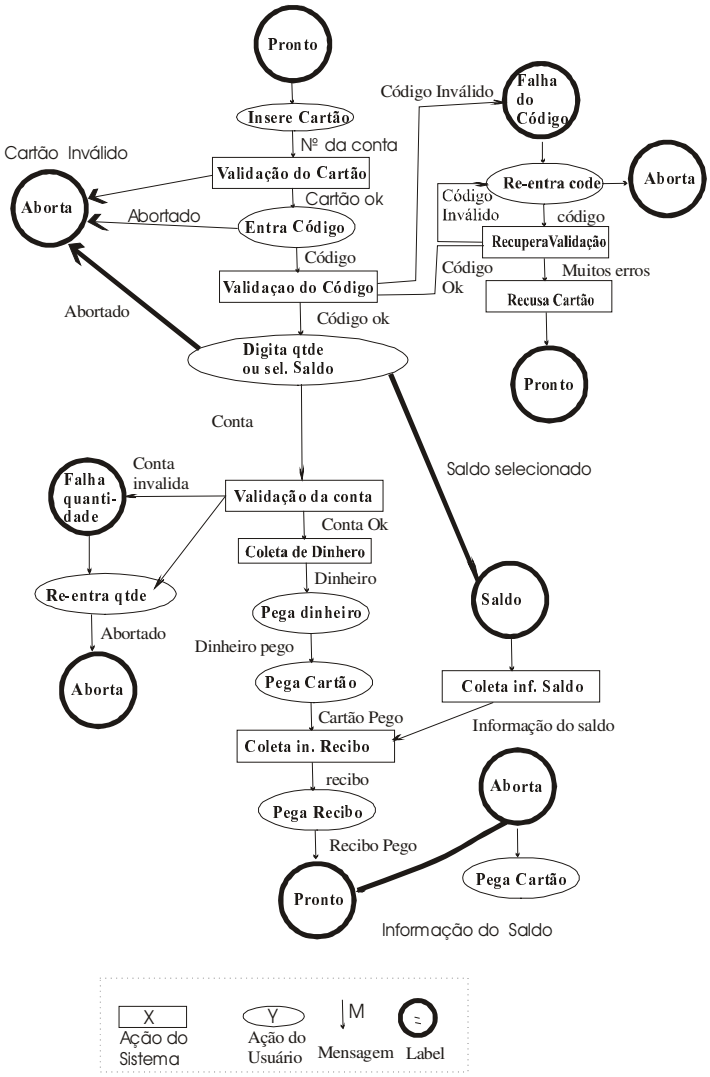

Figura 2.4: Exemplo de modelo de uso sintetizado.

- **Atividade de verificação**

Esta atividade destina-se à obtenção de um SUM completo e consistente. Existem dois passos de verificação relacionados à atividade de formalização e de integração respectivamente.

1 -Verificação da Especificação de caso de uso

Nesta verificação ocorre uma comparação entre a especificação de caso de uso (UCS) e o caso de uso descrito no modelo de caso de uso (UCM) criado durante a fase de análise do UORE.

2 - Verificação do SUM

Esta verificação visa garantir que o SUM cubra todo UCS (Especificação de caso de uso). Isto pode ser realizado por meio de uma ferramenta que poderia checar se todo cenário abstrato de uso é um caminho possível na visão de uso correspondente.

O Modelo SUM foi projetado para ser utilizado como um modelo de referência para as fases restantes do desenvolvimento do sistema. Ele captura aspectos funcionais e comportamentais do sistema que são importantes para seu projeto.

O conjunto de operações atômicas do sistema e seus contextos de uso são uma informação valorosa para o projeto interno. Deve-se considerar o fato de que algumas operações do sistema podem ser atômicas para um ator, mas não para outros, o que pode ser útil para encontrar uma estrutura do objeto do sistema, e para alocar funcionalidade para objetos.

UORE é um melhoramento significante da Análise Dirigida a Caso de uso (UCDA), oferecendo como contribuições principais: o melhoramento de conceitos de casos de uso e atores; a formalização de descrições de casos de uso, a idéia de síntese de casos de uso, o modelo de uso sintetizado e o processo de engenharia de requisitos orientada a uso.

2.1.3 Modelo de caso de uso hierárquico

Regnell et al [43] propõe um modelo de caso de uso hierárquico com representação gráfica, organizado em três níveis: ambiente, estrutura e evento, conforme mostra a figura 2.6. Antes de detalharmos os níveis, veremos alguns conceitos básicos.

Os principais conceitos utilizados, mostrados no modelo conceitual (vide figura. 2.5), são sumarizados como segue. Um caso de uso descreve uma situação de uso do sistema. Um contexto é utilizado para expressar as pré e pós condições que restringem o escopo do caso de uso. Uma pré-condição é definida como propriedades do sistema e seu ambiente que precisam ser satisfeitas para invocar o caso de uso. Uma pós-condição descreve o estado do sistema após o término do caso de uso. Um ator representa um conjunto de usuários que tem algumas características comuns com respeito a porque e como é utilizado o sistema. Um serviço é descrito por um conjunto de casos de uso. Os objetivos são utilizados para categorizar usuários em atores.

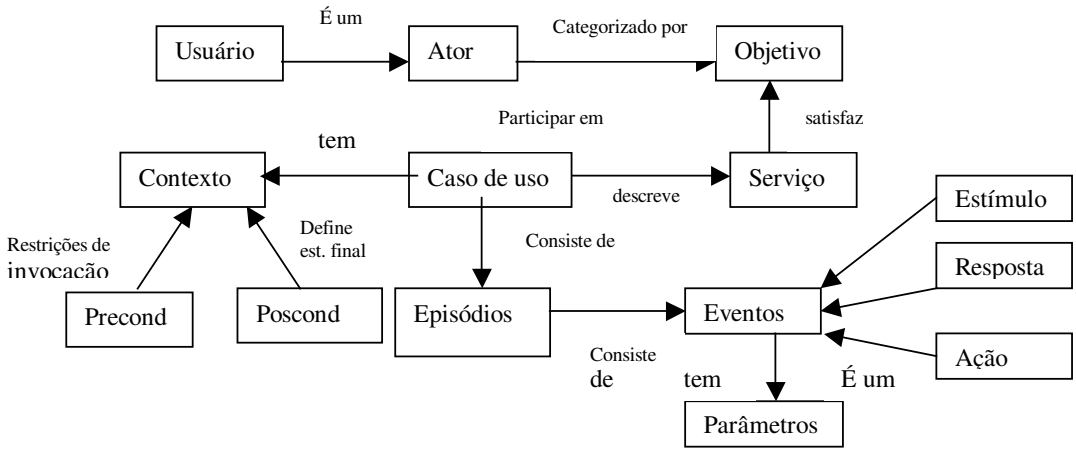

Figura 2.5: Modelo conceitual da abordagem de Regnell et al.

Um caso de uso pode ser dividido em partes coerentes chamadas episódios. Cada episódio consiste de eventos de três tipos: estímulos (mensagens dos usuários para o sistema alvo), respostas (mensagens do sistema alvo para o usuário) e ações (eventos intrínsecos sem nenhuma comunicação entre o sistema alvo e os usuários que participam no caso de uso).

No nível ambiente, o ambiente de cada caso de uso é descrito associando-o com atores relacionados, serviços e objetivos. O nível estrutura descreve cada caso de uso como um grafo de episódios, representando um fluxo coerente e demarcado de eventos.

Um episódio pode ser expandido em novas estruturas de episódios de uma maneira hierárquica. Um episódio folha nesta hierarquia é definida no nível evento como um quadro de seqüência de mensagens. Neste modelo também pode ser utilizada decomposição de episódios de maneira a torná-los mais fáceis de entender e de tamanho gerenciável, assim como para permitir reutilização de episódios entre muitos casos de uso. Neste nível também são definidos operadores de seqüência, alternativa, repetição, exceção e interrupção, assim como as pré e pós condições que restringem o escopo do caso de uso.

No nível evento, os episódios são descritos em detalhes adicionais em termos dos eventos que ocorrem em cada episódio. O nível evento ordena os eventos da mesma

forma como os episódios são ordenados no nível estrutura. Existem três tipos de eventos: estímulo, resposta e ações do sistema. Operadores para alternativa, repetição, exceção e interrupção são também úteis neste nível. Os operadores podem ser aninhados, e atores são instanciados de modo a permitir múltiplas instâncias do mesmo ator participarem no mesmo fluxo de eventos.

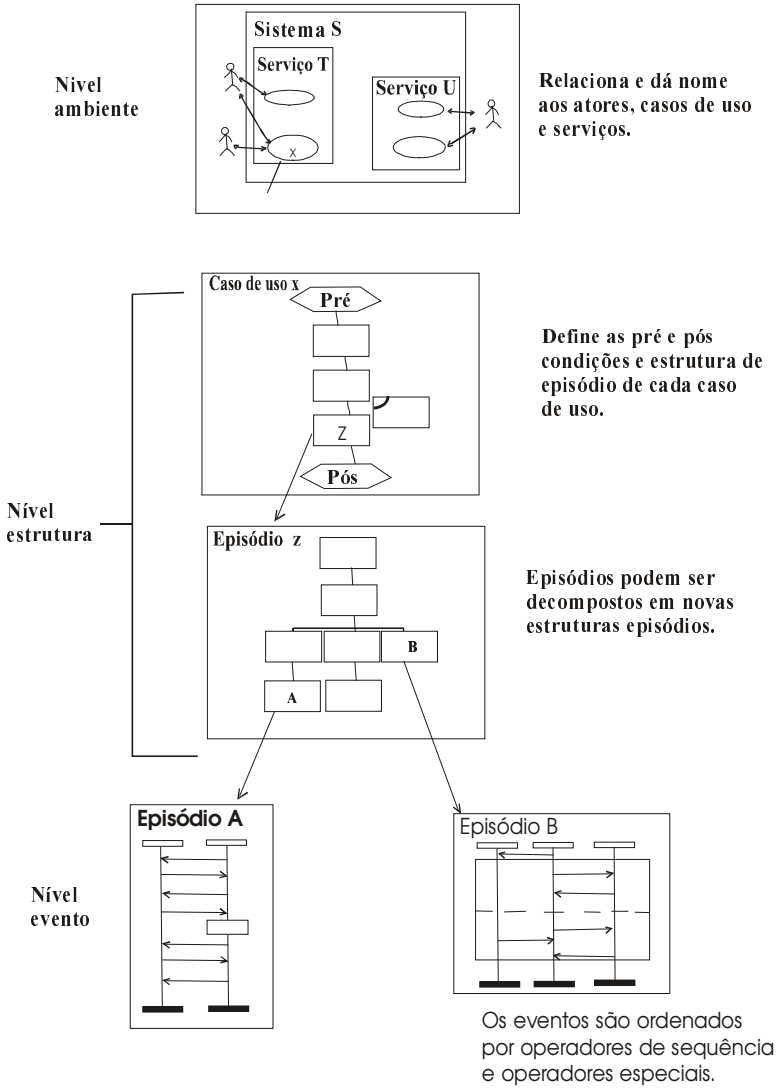

Figura. 2.6: Modelo de caso de uso hierárquico.

O modelo de caso de uso hierárquico possui uma representação gráfica de casos de uso, auxiliando a visualizar os requisitos funcionais, e oferecendo um modelo mais estruturado e menos ambíguo, quando comparado a representações em linguagem natural.

O modelo explanado não oferece uma integração entre modelagem de casos de uso e objetivos. Esse relacionamento será mostrado na próxima abordagem a ser apresentada: Abordagem de Cockburn.

2.1.4 Abordagem de Cockburn

Cockburn [14] ilustra um pequeno modelo de comunicação e interação, distinguindo "objetivos" como um elemento chave de casos de uso, onde estes se assemelham em muito à abordagem de casos de uso de Jacobson [27].

Uma seqüência de problemas é apresentada e colocada quando um grupo de projetos começa a utilizar casos de uso, seguida por uma seqüência de oportunidades que são descobertos à medida em que a estrutura de caso de uso auxilia no projeto de outras maneiras.

O primeiro ponto de conflito discutido na Abordagem de Cockburn consiste na própria definição de casos de uso e sua diferença com relação a cenários. Inicialmente, o conceito de caso de uso consiste em "uma coleção de possíveis seqüências de iterações entre o sistema sob discussão e seus atores externos, relacionados a um objetivo particular". No entanto, são necessárias mais algumas definições para complementar a definição anterior.

"Uma ação conecta o objetivo de um ator com a responsabilidade de outro". Nesta definição são incluídos os atores, que podem ser pessoas ou sistemas de computador. A seguir será mostrado um modelo de comunicação ator-ator por meio da interação. Um ator primário é aquele que possui um objetivo requerendo a assistência do sistema. Um ator secundário é aquele do qual o sistema precisa de assistência para satisfazer seus objetivos.

Cada ator possui um conjunto de responsabilidades. Para realizar determinada tarefa são apresentados alguns objetivos. Para alcançar um objetivo são executadas algumas ações. Uma ação é o acionamento de uma interação com outro ator, chamando uma das responsabilidades de outro ator.

O Diagrama Entidade-Relacionamento do modelo de comunicação apresentado pela abordagem de Cockburn é mostrado na figura. 2.7 e explicado a seguir. Existem atores internos e externos. Um ator externo pode ser uma pessoa, um grupo de pessoas ou um sistema de qualquer tipo. O ator interno pode ser o sistema em projeto, um subsistema ou um projeto. O sistema em projeto consiste de subsistemas, e estes de objetos. Os atores têm

comportamentos. O comportamento do nível de topo é uma responsabilidade. Ele contém objetivos, e estes contêm ações. Uma ação aciona uma interação, onde uma interação é o objetivo de um ator sobre a responsabilidade de outros atores.

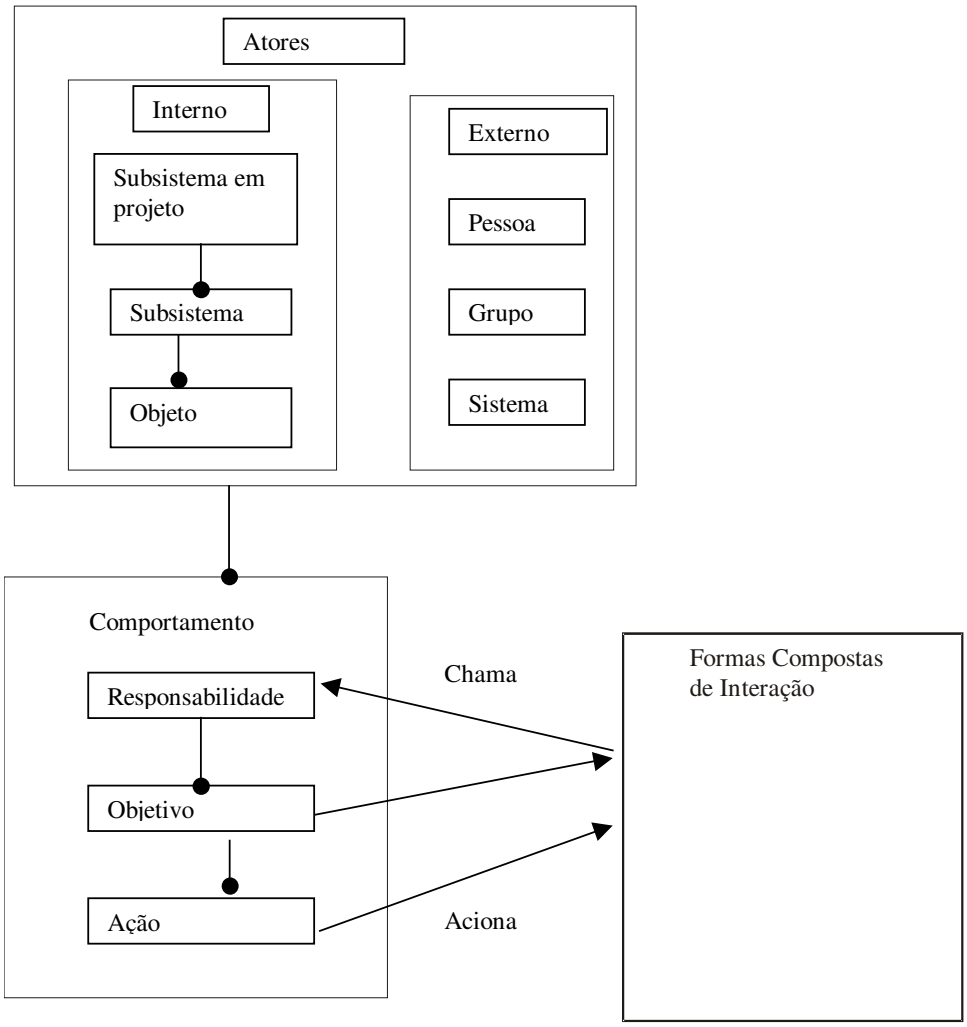

Figura. 2.7: DER do Modelo de Comunicação

Uma interação é simples ou composta. No caso da composta, esta pode ser o envio de uma mensagem ou uma seqüência de interações. Para delimitar o término de um caso de uso ou um cenário são necessárias as seguintes considerações:

- Todas as interações relacionam-se ao mesmo objetivo

- Interações começam com o evento de acionamento e terminam quando o objetivo é realizado ou abandonado, e o sistema completa suas responsabilidades com respeito à interação.

A seguir são definidos os termos casos de uso e cenários, bem como as informações que os caracterizam. Os casos de uso caracterizam-se pelos atores, objetivos e cenários utilizados. No caso dos cenários devem ser colocados como atributos: ator primário, objetivos, condição sob a qual o cenário ocorre e resultado dos cenários.

O segundo ponto enfocado por Cockburn refere-se a como controlar a explosão de cenários. Explosão de cenário é evitada utilizando três técnicas: casos de uso subordinados, extensões e variações.

Variações é um local do texto do caso de uso onde são colocadas as possíveis diferenças na entrada dentro de um caso de uso. Essa estrutura de organização evita a explosão de cenários que ocorreria caso fosse feito a tentativa de listar todas as possíveis maneiras de interagir com o sistema.

As falhas de um passo são manipuladas por outros cenários, ou um cenário de extensão, sendo utilizadas muitas técnicas para combinar os cenários principais e os de extensão. Cada linha em um cenário é um objetivo de um caso de uso subordinado ou uma ação primitiva. Dessa forma, a expressão de cenário é prevenida porque um caso de uso subordinado contém e oculta um grande número de caminhos alternativos.

Distingue-se curso principal a partir de cursos alternativos. O curso principal é o curso mais simples, em que o objetivo é realizado sem dificuldade e sem necessidade de nenhuma recuperação. Os cenários alternativos são compostos pelos cenários de recuperação e de falha, que conduzem ao término do objetivo.

Nesta abordagem, casos de uso e cenários são dispostos em vários níveis: nível de escopo ou limite do sistema, nível de especificidade do objetivo e o nível de detalhe da interação. No nível de escopo do sistema encontram-se o objetivo e o caso de uso. A segunda dimensão de refinamento é o detalhamento do objetivo. Ela consiste de objetivos sumários, objetivos do usuário e subfunções.

O nível de maior interesse é o objetivo do usuário, já que constitui o objetivo que o ator primário está tentando executar. Coleções de objetivos do usuário formam os

objetivos sumários. Um objetivo sumário pode coletar todos os objetivos do usuário relacionados a um determinado assunto. Abaixo dos objetivos do usuário estão as subfunções. Uma subfunção é um subobjetivo ou um passo em um cenário externo, abaixo do nível principal de interesse para os usuários.

O terceiro tipo de nível é o de detalhe de interação. São visualizados dois níveis: nível de interface do diálogo e interface semântica. A maioria das pessoas prefere trabalhar no nível de interface semântica, uma vez que preserva a máxima latitude na implementação da interação em diferentes tecnologias e na aceitação de variação dos dados (aceitando, por exemplo, diferentes formatos de endereço para vários países). O nível de detalhe de interação é consistente entre projetos, pelo menos para objetivos do usuário.

O quarto ponto de enfoque relaciona-se à interpretação de casos de uso apenas como descrições de interface do usuário, correspondendo ao nível de interação do diálogo. Entretanto esta interpretação escrita de casos de uso como descrições de interface do usuário não os valoriza adequadamente dentro de um projeto. Isso se deve à probabilidade de mudança freqüente do projeto da interface do usuário, não sendo possível utilizá-la como requisito contratual e como requisito do sistema.

Por esta razão, é aconselhável trabalhar no nível de interação semântica descrevendo qual informação deve passar o limite do sistema, sem descrever a seqüência ou natureza da interação (sem descrever o diálogo).

O outro ponto abordado concerne à escrita do texto. Os casos de uso descritos têm conteúdo e estrutura semi-formal, permitindo que seja incluída uma seção de variação. Geralmente, a forma geral da prosa é a seguinte:

<Tempo ou fator de seqüência>...<ação>...<ação>...<restrições>.

A fim de manter a narrativa clara, melhorar rastreabilidade a partir de requisitos para projeto ou teste, os casos de uso são especificados com linhas de referência necessários na seção de extensão.

Os termos chaves encontrados na literatura são casos de uso, cenário, *script*, curso alternativo e principal, assim como os construtores *use*, *extends* e de subordinação. A abordagem de Cockburn mantém a definição de casos de uso de Jacobson como um conjunto de possíveis seqüências que sucede ou falha para executar o objetivo.

A abordagem de Cockburn provê a extensão da abordagem de Jacobson com o adicionamento do conceito de objetivo de uma forma explícita, observando-se a separação de cursos de acordo com sucesso ou falha.

Com o objetivo de não repetir as partes comuns de um cenário previamente descrito, deve-se coletar todos os cursos alternativos e colocá-los na seção extensão, que é uma seção onde ocorre separação dos cursos de acordo com o sucesso ou falha.

A relação "use" de Jacobson encontra similaridade com a abordagem de Cockburn, uma vez que toda linha em um cenário nomeia também uma ação primitiva ou um caso de uso subordinado. Já a relação "extend" de Jacobson encontra-se na abordagem de Cockburn na seção de extensão, onde as partes comuns devem ser colocadas juntamente com a sua origem na narrativa principal e em qual condição elas se diferenciam (contexto de ocorrência), com o propósito de evitar sua repetição.

Nesta seção de extensão são encontradas algumas linhas, assim como a recorrência para a narrativa principal, bem como para alguma execução específica. Para efeitos de comparação, o construtor *use* corresponde a um item de linha do cenário, referenciando outro caso de uso (chamada de subrotina), enquanto a "extensão" é outro cenário a que se refere em algum ponto dentro do cenário.

Entre os problemas que permanecem, existe a contínua confusão sobre os níveis de casos de uso, embora a introdução de termos identificando os três níveis explanados auxilie ao engenheiro de requisitos. As variações de dados, ainda requerem execução informal seguindo o nível apropriado de refinamento. Além disso, a trilhagem encontra dificuldade, uma vez que os aspectos e funcionalidades do sistema envolvem muitos casos de uso.

2.1.5 Processo de Engenharia de Requisitos dirigida a caso de uso

Em [17] é proposta uma abordagem baseada em casos de uso para auxiliar ao analista durante as atividades de aquisição e conceitualização de requisitos, com o propósito final de produzir especificações orientadas a objeto. A abordagem é orientada a especialistas do domínio no sentido de que eles podem ativamente participar durante a atividade de aquisição de requisitos identificando e descrevendo os casos de uso.

Dessa forma, dentro desse processo de Engenharia de requisitos orientada a objeto, podem ser focalizadas duas atividades: a atividade de aquisição de requisitos e a de conceitualização de requisitos, descritas a seguir.

A atividade de aquisição de requisitos é a atividade durante a qual casos de uso são coletados e descritos por meio de duas técnicas. A primeira técnica utiliza tabelas para facilitar a comunicação entre o analista e o *expert* do domínio. A segunda técnica utiliza redes de Petri e traz o formalismo requerido para o analista. Algumas regras de mapeamento são apresentadas, que auxiliam ao analista na elaboração de redes de Petri a partir de tabelas.

A atividade de aquisição de requisitos consiste em obter elementos comportamentais relevantes para o sistema capturando as necessidades do *expert* do domínio de uma maneira sistemática, utilizando uma "abordagem dirigida a caso de uso" que naturalmente facilita a comunicação entre o analista e o expert do domínio. Duas técnicas são utilizadas para adquirir e descrever casos de uso, incluindo uma notação tabular direcionada ao *expert* do domínio e uma técnica de modelagem precisa mais direcionada ao analista.

Nesta abordagem, um caso de uso é definido como um "objetivo" que um ator deve executar utilizando uma funcionalidade completa do sistema. Para a descrição de casos de uso foram utilizadas tabelas com o intuito de facilitar a comunicação entre o analista e o especialista do domínio. Estas tabelas são definidas com ações, os estados dos tipos de objetos durante a execução de cada ação, bem como as condições e suposições. Estes elementos coletados em uma tabela descrevem apenas parte de um caso de uso, sendo chamado portanto de "Tabela de caso de uso parcial" (PUCT). As tabelas contêm os estados dos tipos de objetos envolvidos na execução de cada ação. O conjunto de estados de cada tipo de objeto definido durante a execução de uma ação constitui a configuração dessa ação.

Para uma ação, pode existir mais de uma configuração, sendo distinguidas pelas condições ou hipóteses associadas àquela ação. Todas as tabelas que descrevem um caso de uso podem ser representadas por algumas árvores chamadas Árvores de caso de usos (UCT), onde a raiz da árvore é uma tabela PUCT, observando-se que nós são os PUCTs que contém condições e as folhas são PUCTs que não contém condições.

Embora a descrição de casos de uso por meio de tabelas faça o envolvimento do *expert* do domínio durante a atividade de aquisição de requisitos mais fácil, faz-se necessário utilizar uma técnica mais rigorosa como as redes de Petri, de modo a oferecer ao

analista um nível de formalidade para garantir que nenhum requisito seja mal-entendido, incompleto ou inconsistente. Esta técnica é uma das mais comuns para modelar aspectos dinâmicos e permite tratar aspectos como concorrência, não determinismo e conexões causais entre eventos.

As regras são propostas de modo a elaborar redes de Petri a partir de elementos contidos nas tabelas de caso de uso parciais (PUCTS). Estas regras permitem a identificação de alguns locais, algumas transições e alguns arcos da rede. A primeira regra identifica e define locais a partir de estados contidos em uma configuração de uma ação dada, aplicando-se a cada configuração das ações definidas em um PUCT. Esta regra consiste em agrupar juntos os estados contidos em uma configuração em um ou alguns conjuntos de estados, cada um formando um local.

A regra dois identifica alguns locais a partir de suposições associadas com uma dada ação. Esta regra aplica-se a uma dada ação para o qual existe uma suposição associada. A terceira regra consiste em unificar locais identificados da aplicação da regra um e alguns locais identificados da aplicação da regra dois em um local.

A quarta regra tem o propósito de identificar algumas transições e alguns arcos da rede. Esta regra conecta os locais que tenham sido identificados da aplicação das regras um, dois e três com transições e arcos. Finalmente a regra cinco visa identificar algumas transições e alguns arcos da rede, conectando os locais identificados da aplicação das regras um, dois e três. Mais especificamente, sejam pk e pk' um conjunto de locais identificados na aplicação da regra um, regra dois e três para outra configuração da ação ak, e o conjunto de locais identificados na aplicação da regra um para outra configuração da ação ak. No caso destas duas configurações serem separadas por uma linha tracejada na PUCT, então isso indica uma transição e um arco daquela transição para cada local em pk'.

Em virtude da descrição de casos de uso não ser suficiente para se obter uma visão unificada do sistema, são introduzidos novos tipos de *links* entre casos de uso chamados "links temporais", expressando como os casos de uso são dependentes do tempo. Diferentemente dos *links* de composição *uses*, *extends* e *adds* entre casos de uso, estes são *links* de escalonamento.

A segunda fase dessa abordagem é a fase de conceitualização que consiste em mapear os elementos obtidos para conceitos comportamentais objetos (incluindo os conceitos de estado, evento e transição).

Durante esta segunda fase, outras regras são também propostas para auxiliar ao analista a produzir especificações orientadas a objeto dinâmicas a partir de redes de Petri. A aplicação dessas regras permite produzir um Diagrama de Transição de Tipos de Objetos a partir de uma dada árvore de caso de uso parcial (PUCT).

A utilização de tabelas na fase de aquisição de requisitos não permite expressar uma descrição complexa de forma completa. Durante essa fase, devem ser extendidas as tabelas, com o adicionamento de novos elementos para expressar ações concorrentes e iterativas, e não apenas de seleção.

A essa abordagem falta uma melhor definição das regras para prover ao analista um melhor suporte durante as atividades de aquisição e conceitualização na elaboração de especificações orientadas a objeto.

2.1.6 Estudo comparativo entre as abordagens de casos de uso selecionadas.

Um estudo comparativo entre algumas abordagens que utilizam casos de uso explicadas nas seções anteriores é realizado nesta seção e sumarizado na tabela 2.1. Foram descritos a fase de Engenharia de requisitos do Método Objectory, o Método UORE, o modelo de caso de uso hierárquico, a abordagem de Cockburn e um modelo de caso de uso baseado em redes de Petri.

Como critérios de comparação foram escolhidos os produtos obtidos como resultado do aplicação do método, as fases desse processo, os principais conceitos em que se baseiam os métodos escolhidos, as particularidades e características de cada método e, por fim, as falhas existentes em cada uma das abordagens que utilizam casos de uso.

2.2 Abordagens que utilizam cenários para elicitação e modelagem de requisitos

2.2.1 Abordagem formal para Análise de Cenários

A análise de cenários é um dos métodos possíveis de simular o sistema para acompanhar alguma função que o usuário deseja, gerando especificações que descrevam o comportamento do sistema sem ambigüidade, consistentemente e completamente.

Em [25] é apresentado um método formal, sistemático, que provê uma base matemática formal, e gera automaticamente cenários precisos, não ambíguos, consistentes e internamente completos endereçando mudanças nos requisitos e envolvimento do usuário na identificação do cenário.

O modelo de processo desse método formal apresenta seis estágios, conforme é mostrado na figura 2.8 e comentado a seguir.

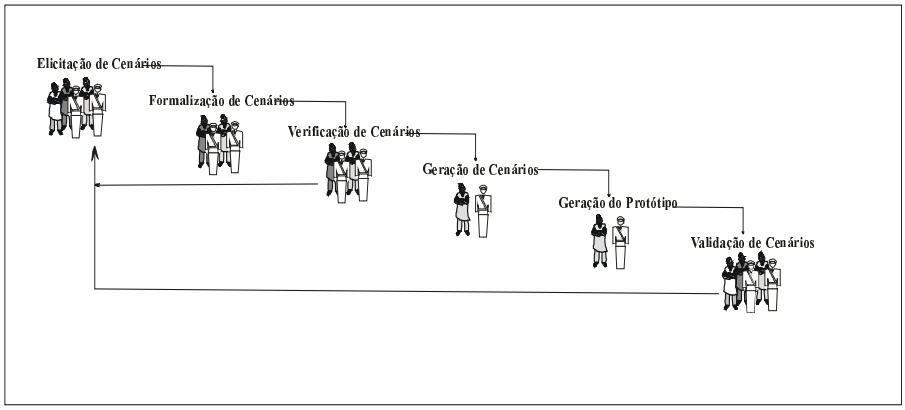

Figura. 2.8: Modelo de processo de análise de cenários.

1 - Elicitação de cenários (analista de requisitos com usuários)

Nesta fase é construída a árvore de cenários – que é um mecanismo que descreve e representa todos os cenários na visão de um usuário particular. Para construir uma árvore são identificadas e classificadas as visões de usuários. Uma visão de usuário é um conjunto de cenários específicos como vistos por um determinado grupo de usuário. Uma instância de visão de usuário é um conjunto de cenários vistos por um usuário particular. Cada visão de

usuário consiste de cenários, que são iniciados por agentes, que são usuários externos, estímulos externos ou componentes de software, como módulos ou objetos.

2- Formalização de cenários

Nesta fase é convertida cada árvore de cenário em uma gramática regular equivalente. Esta gramática é utilizada para construir uma máquina de estados conceitual que modela o comportamento do sistema de uma visão do usuário particular.

Os nós do modelo são estados do sistema e as setas são os eventos. A gramática e a correspondente máquina de estados conceitual constituem o modelo formal abstrato.

2 - Verificação de cenário

É verificado então o modelo formal abstrato automaticamente para descobrir inconsistências, redundâncias e incompletezas internas dos cenários.

3 - Geração de cenários

É utilizado este processo automático para gerar os cenários no modelo formal verificado usando a máquina de estados conceitual.

4 - Geração de protótipo

É utilizada prototipagem rápida para construir o sistema esperado na base do cenário gerado.

6- Validação do cenário

Finalmente, o analista utiliza o protótipo do sistema para validar os cenários e demonstrar sua validade para o usuário.

Esse processo é adaptável apenas para sistemas que tem uma resposta única para um estímulo, sendo ambos eventos semelhantes a interface do usuário, não sendo adaptável para sistemas complexos como um controlador para múltiplos elevadores, uma vez que não permite manipular respostas e estímulos concorrentes.

2.2.2 Análise de requisitos baseado em um modelo cíclico de averiguação

Algumas abordagens enfatizam detalhes lingüísticos, transformações e outros formalismos como chave para obter requisitos claros e válidos. Outras preferem visualizar o processo de análise como essencialmente baseado em averiguação - uma série de questões e respostas projetadas para apontar a origem da informação e quando esta deve ser utilizada.

Esta averiguação enfatiza o que é apropriado para os projetos contratuais e dirigidos ao mercado. Em projetos contratuais, o cliente usualmente escreve um documento de requisitos, mas ele tem muitas ambigüidades, incertezas e lacunas que os desenvolvedores devem avaliar cuidadosamente, incrementalmente, refinando e formalizando a informação até produzir uma especificação funcional.

Em projetos dirigidos ao mercado, o cliente não é facilmente identificável e não há documento de requisitos sancionado pelo cliente. Neste caso, os desenvolvedores devem produzir uma especificação a partir de uma declaração vaga de oportunidades e objetivos.

Com o objetivo de identificar requisitos, em [39] foi desenvolvido um modelo cíclico de pesquisa que se constitui em uma estrutura formal e dinâmica para descrever discussões sobre requisitos. O modelo cíclico descrito refere-se à necessidade de suportar comunicação durante o processo de requisitos. Este modelo consiste de uma série de questões e respostas projetadas para capturar o processo de elaboração de requisitos e documentação através de refinamento. Os principais problemas encontrados referem-se à comunicação, acordo sobre requisitos e gerenciamento de mudanças.

Como mostrado na figura 2.9, os requisitos são incrementalmente elaborados por meio de discussões e confirmação contribuindo assim para o refinamento dos requisitos.

O modelo de averiguação/Indagação é suportado de modo que os participantes (i.e os *stakeholders* e os analistas) saibam qual informação está ausente e qual está pendente. Uma vez que este modelo é baseado em artefato, ele habilita os *stakeholders* a participarem do processo de análise para discutirem artefatos que são visíveis e explícitos.

O Modelo cíclico de averiguação/indagação possui três fases: Documentação, Discussão e Evolução como mostrado na figura 2.9 e descritas posteriormente.

A fase de documentação de requisitos refere-se ao processo dos *stakeholders* (usuários finais, usuários indiretos, desenvolvedores e clientes) escreverem os requisitos propostos, sendo que cada requisito é um nó hipertexto separado.

No entanto, observa-se que, em alguns projetos, a análise de requisitos começa com alguma forma de documentação de requisitos. Em outros, pode haver uma declaração de uma ou duas páginas dos objetivos.

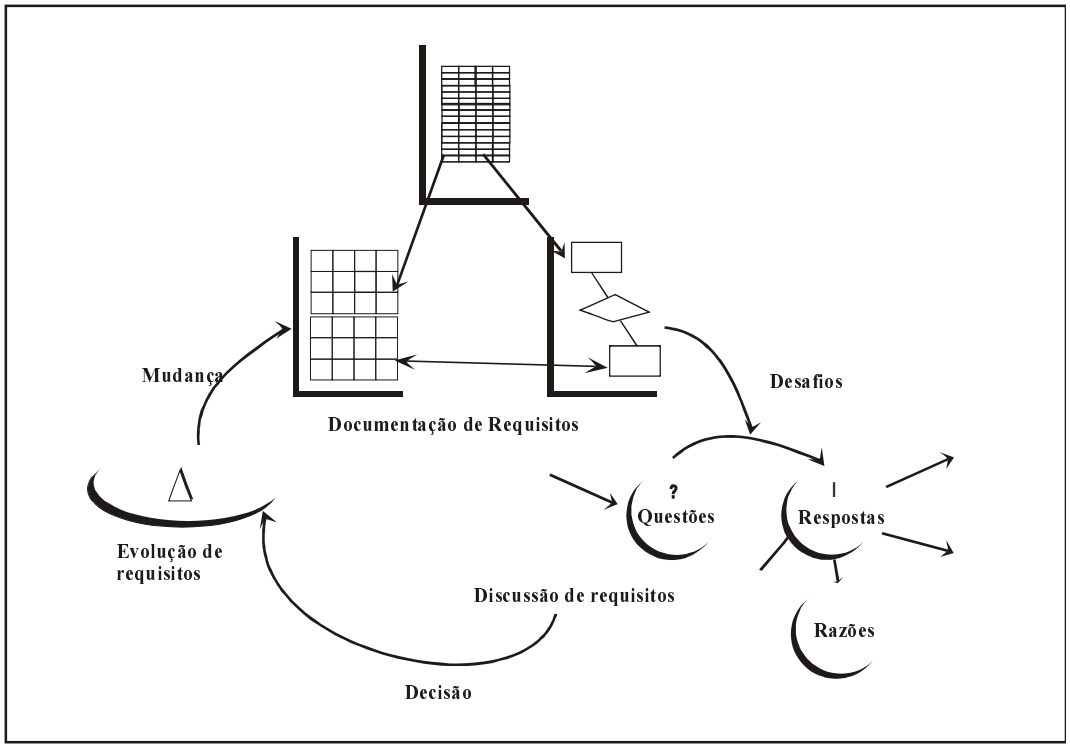

Figura 2.9: Modelo cíclico de averiguação.

Considerando que o ponto de início seja vago, assume-se que o resultado de utilizar o modelo cíclico de averiguação será uma especificação refinada consistindo essencialmente de um conjunto de requisitos que descrevem o sistema desejado. Caso não exista documento de requisitos, o modelo provê um processo sistemático e incremental para escrevê-lo.

No estágio de documentação existem algumas maneiras de analisar requisitos. No caso de haver um documento de requisitos existente, este pode ser revisado. Caso não

haja, deve-se começar a partir do nada para escrever requisitos baseados em entrevistas, documentação técnica para sistemas similares e outros.

Uma valorosa técnica nesta fase é a análise de cenários (um cenário é simplesmente o uso específico proposto do sistema). Mais especificamente, um cenário é uma descrição de uma ou mais transações envolvendo o sistema requerido e o ambiente. Cenários podem ser documentados de diferentes maneiras, dependendo do nível de detalhe necessário. A forma mais simples é o caso de uso, que consiste meramente de uma descrição com um número relacionado. Formas mais detalhadas são chamadas *scripts*. Eles são usualmente representados como tabelas ou diagramas e envolvem a identificação de uma ação e um agente da ação. Por esta razão, um *script* pode ser chamado tabela de ação.

Cenários são úteis na aquisição e validação de requisitos, mas eles não constituem requisitos porque eles descrevem o comportamento do sistema apenas em situações específicas; uma especificação, por outro lado descreve o que o sistema deve fazer em geral.

A segunda fase desse modelo consiste na discussão de requisitos, onde os *stakeholders* discutem os requisitos propostos com anotações. Existem três elementos nesta fase: questões, onde muitas discussões começam porque um *stakeholder* tem uma questão sobre um requisito; respostas onde são descritas soluções para problemas na forma de refinamentos candidatos ou revisões que respondem às questões. Uma questão pode gerar muitas respostas. Respostas esclarecem os *stakeholders* e auxiliam a noticiar ambigüidades, requisitos perdidos e inconsistências. O último elemento corresponde às razões, que consistem em respostas que requerem justificativas se os requisitos não são imediatamente óbvios.

Na terceira fase, denominada de evolução de requisitos, os *stakeholders* relacionam as requisições de mudanças aos requisitos na base de discussão, refinando-os quando os requisitos de mudanças são aprovados.

O objetivo da fase de discussão de requisitos é a confirmação de permanecer ou modificar o requisito. Uma requisição de mudança pode ser trilhada de volta para uma discussão, que constitui sua razão e avançar para os requisitos modificados, uma vez que eles tenham sido executados.

Assim como na fase de discussão, a fase de evolução pode ocorrer gradualmente e informalmente, ou em discretas mudanças seguindo uma revisão formal e procedimentos aprovados.

Esse modelo, por sua informalidade, apresenta-se sujeito a ambigüidades e incertabilidades, dificuldades referentes à comunicação entre os envolvidos no processo, acordo sobre requisitos e o gerenciamento de mudanças nos mesmos.

2.2.3 Abordagem CREWS-L`ECRITOIRE

A abordagem CREWS-L'Ecritoire descrita em [1,2,3,45,50] se propõe a acoplar objetivos e cenários para auxiliar diretamente na atividade de engenharia de requisitos, descobrindo requisitos por meio de um acoplamento objetivo-cenário.

O processo de elicitação de requisitos pode ser organizado em duas fases: Obtenção de cenário e descoberta de objetivos, executados iterativamente. Dessa forma, uma das características principais dessa abordagem consiste na utilização de um acoplamento bi-direcional cenário-objetivo permitindo mover-se a partir de cenários para objetivos e vice-versa. Considerando a direção objetivo-cenário, à medida em que é descoberto um objetivo, este é operacionalizado por um cenário. Explorando o relacionamento objetivo-cenário na direção reversa, i. é, de cenários para objetivos, a abordagem guia o processo de elicitação de requisitos descobrindo novos objetivos através da análise de cenários textuais.

Outra característica importante dessa abordagem concerne à utilização dos relacionamentos de refinamento e AND/OR entre objetivos. Isto conduz à organização de uma coleção de requisitos como uma hierarquia de RCs (Pedaços de requisitos) relacionados através dos relacionamentos AND, OR e de refinamento.

Além disso, essa abordagem possui um suporte metodológico provido na forma de guias de regras executáveis personificadas em um ambiente de software denominado CREWS- L'Ecritoire . Como resultado, é possível guiar o processo de elicitação de requisitos através da modelagem de objetivo intercalada com a obtenção de cenário.

A abordagem tem como núcleo o RC, definido como o par <objetivo, cenário>, organizados em três níveis de abstração: contextual, interação do sistema e interno do sistema, onde cada nível corresponde a um tipo de RC.

No nível contextual, são identificados os serviços que o sistema deve prover para uma organização e suas razões. O RC contextual acopla um objetivo de projeto a um cenário de serviço. Um cenário de serviço descreve o fluxo de serviços entre agentes que são necessários para executar o objetivo de projeto.

O nível de interação de sistema focaliza-se nas interações entre o sistema e seus usuários necessárias para executar os serviços associados ao sistema no nível contextual. Um RC de interação do sistema combina um objetivo de serviço a um cenário de interação do sistema, onde são descritos uns fluxos de interações entre o sistema e seus usuários para executar o objetivo de serviço.

O nível interno do sistema focaliza-se no que o sistema pode internamente executar. Um RC interno do sistema combina um objetivo do sistema a um cenário interno do sistema. Um objetivo do sistema expressa uma maneira possível de executar uma ação identificada em um cenário de interação do sistema.

Os RCs podem ser interligados através de três tipos de relacionamentos, a saber: AND (composição), OR (alternativa) e de refinamento, sendo organizados como uma coleção de requisitos em uma hierarquia de RCs.

Os relacionamentos de composição (AND) interligam aqueles RCs que requerem um ao outro para definir um sistema funcionando completamente. Os relacionamentos alternativos (OR) representam maneiras alternativas de executar o mesmo objetivo. Os relacionamentos de refinamento são utilizados para descrever RCs em diferentes níveis de abstração, sendo direcionado pela parte do cenário que considera toda interação do cenário no nível i como um objetivo a ser executado no nível i + 1.

O modelo de processo da abordagem CREWS- L'Ecritoire é representado como um mapa , i.e, um grafo direcionado com intenções como nós e estratégias como setas, mostrado na figura 2.10.

Um aspecto importante do modelo de processo CREWS- L'Ecritoire consiste na provisão de diferentes estratégias para suportar a elicitação de objetivos: estratégia alternativa, de composição e refinamento. A essas estratégias são associadas regras para descoberta de novos objetivos, implementadas em um protótipo de software L' Ecritoire.

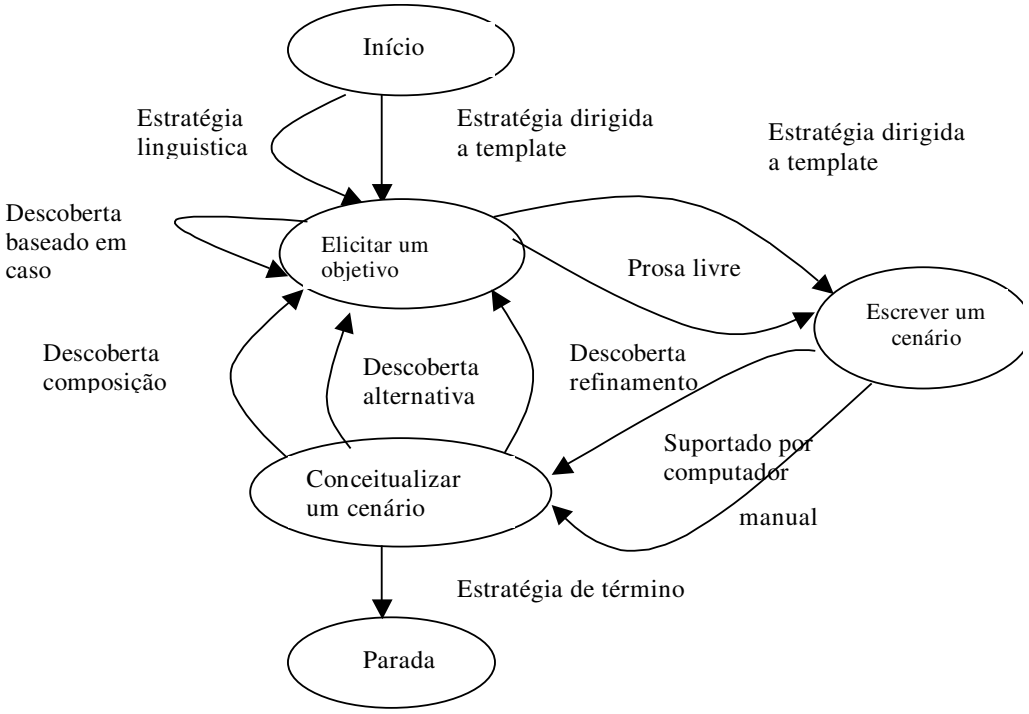

Figura 2.10.: Modelo de Processo da abordagem CREWS-L' Ecritoire.

O modelo de produto mostrado na figura 2.11 representa os principais conceitos utilizados por esta abordagem, relacionados por *links* de composição, associação ou *is-a.*

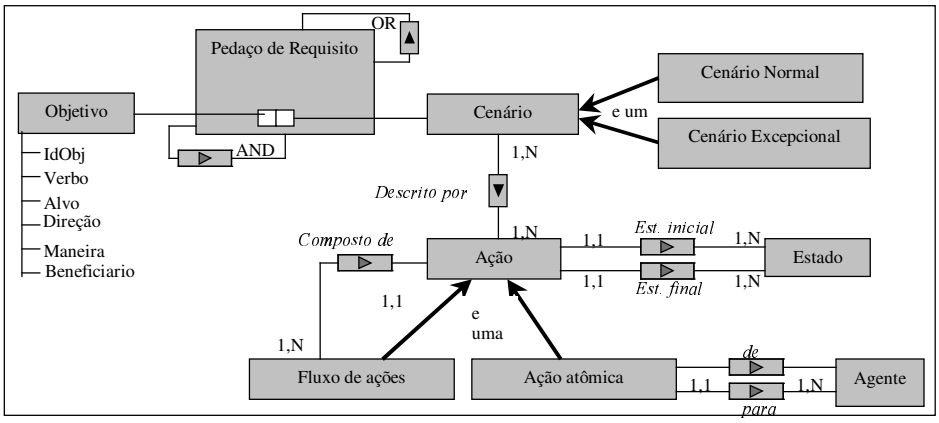

Figura. 2.11. Modelo de Produto da abordagem CREWS- L'Ecritoire.

Em suma, a abordagem CREWS- L`Ecritoire auxilia no endereçamento dos seguintes aspectos:

- Obtenção de cenários [2], provendo guias de estilo e conteúdo informais para a escrita de cenários, auxiliando na limitação do tamanho dos mesmos, com sugestões de templates para a escrita de sentenças.
- Organização de cenários, através da integração de cenários em coleções de cenários [3].
- Transição de cenários informais para formais.
- Suporte metodológico, com a definição de regras para a descoberta de objetivos e verificação de cenários, implementadas em um ambiente de software L'Ecritoire.
- Análise de cenários, através da descoberta de requisitos/objetivos analisando os cenários de uma coleção objetivo-cenário.
- Fragmentação de cenário, através da organização de coleções objetivos-cenários de uma maneira hierárquica.

A CREWS- L'Ecritoire é uma abordagem que apresenta os aspectos positivos mencionados acima. Um ponto não tratado com muitos detalhes por essa abordagem refere-se a outros tipos de relacionamentos entre objetivos, assim como uma estratégia eficaz na elicitação de objetivos iniciais.. Ela também não focaliza o conceito de "casos de uso", nem seu relacionamento com "objetivos".

Em [41] é ilustrado um método integrado a partir da abordagem CREWS-L´Ecritoire com o método OOSE [27]. A integração dos dois métodos consiste na integração dos seus modelos de processos e modelos de produtos utilizando operadores específicos. Essa integração visa adicionar à construção do modelo de caso de uso do método OOSE os aspectos de descoberta do objetivo e obtenção de cenário da abordagem CREWS- L´Ecritoire. Em contrapartida, importar as estratégias de abstração e extensão do método OOSE não existentes na abordagem CREWS- L´Ecritoire. A figura 2.12 mostra o modelo de processo integrado.

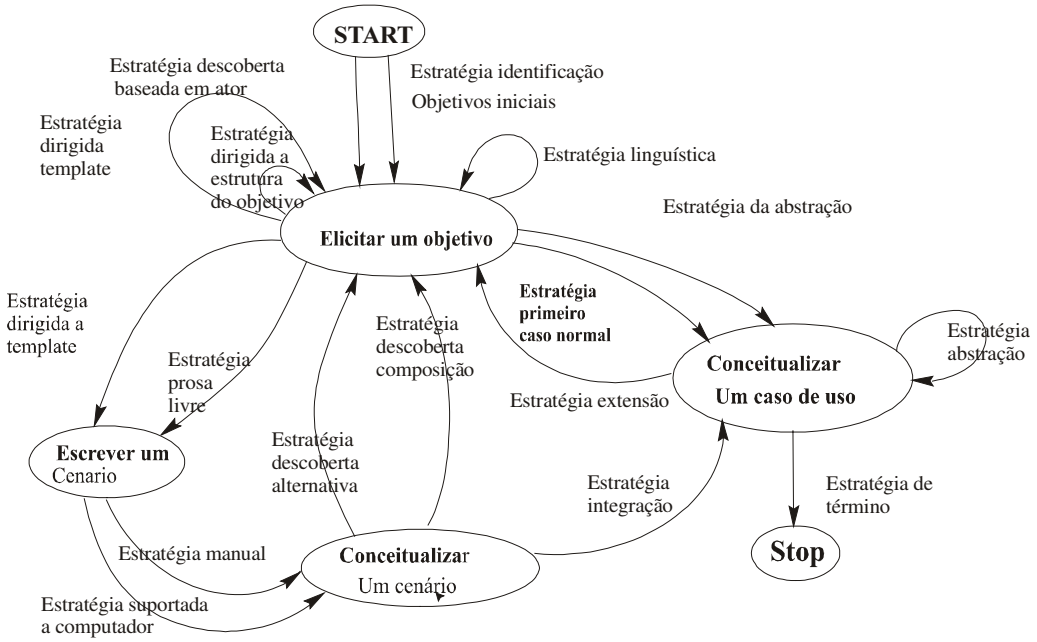

Figura. 2.12: Modelo de processo integrado.

2.2.4 Modelo de cenários de Julio César Leite et al

Em [34,35] é apresentado uma proposta para adicionar uma visão do cenário à estrutura de requisitos, composta das seguintes visões: visão de modelo básico, visão do modelo léxico, visão do modelo de cenários, visão de hipertexto e visão de configuração.

Antes de falarmos sobre cada uma das visões, dar-se-á um enfoque maior no modelo de cenários, onde foram encontradas as seguintes definições, a seguir descritas:

- Um cenário começa descrevendo situações no macrosistema, e sua relação com o sistema mais externo. Isto é, primeiro devem ser consideradas as interfaces do macrosistema, posteriormente devem ser descritas as interfaces do software com seu macrosistema.

- Um cenário evolui assim como se processa a construção de software;

- Cenários são naturalmente relacionados ao LEL (Léxico Extendido da Linguagem) e à visão do modelo básico, da estrutura de requisitos.

- Um cenário descreve situações, com uma ênfase na descrição do comportamento. Cenários, similarmente à visão do Modelo Básico, usa descrições de Linguagem Natural como sua representação básica.

A visão do modelo básico (BMV) é uma estrutura que incorpora sentenças sobre o sistema desejado. Estas sentenças são escritas em linguagem natural seguindo padrões definidos. No que concerne à visão do modelo de cenário, esta é uma estrutura composta de objetivo, contexto, recursos, atores e episódios.

Na fig. 2.13 é utilizada uma estrutura entidade-relacionamento que descreve o modelo de cenário. Observe que um episódio pode ser explicado como um próprio cenário em si, habilitando dessa forma a possibilidade de decomposição de um cenário em um sub-cenário.

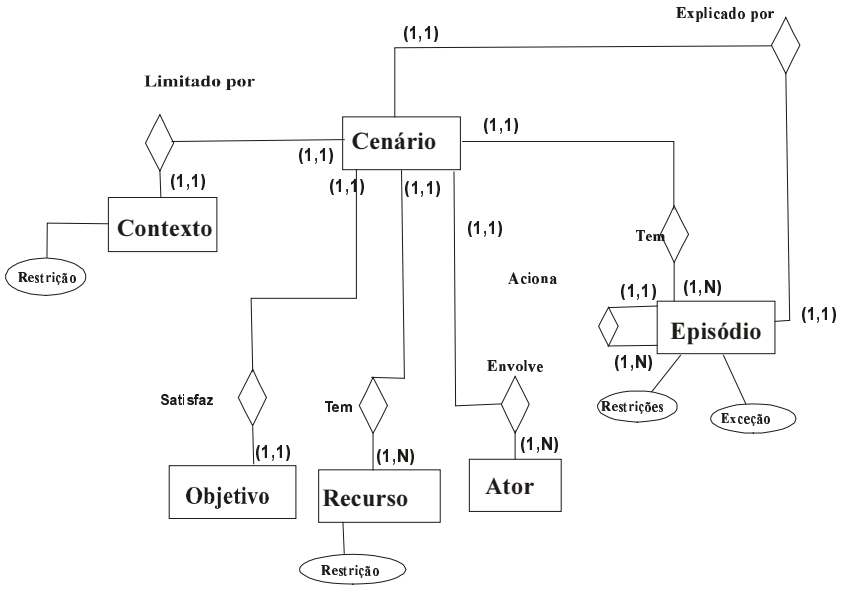

Figura. 2.13. Diagrama ER do Modelo de Cenários de Julio C. Leite et al.

O modelo léxico corresponde à representação dos símbolos na linguagem de domínio do problema, denominada LEL (Léxico Extendido da Linguagem), onde sua obtenção consiste em entender a linguagem do problema, sem preocupar-se com o entendimento do problema.

O LEL é uma representação da linguagem natural que auxilia a capturar o vocabulário da aplicação, possuindo como principal objetivo registrar os sinais (palavras ou frases), peculiares ao domínio. Seu objetivo consiste em descobrir a semântica dos símbolos de uma aplicação no contexto em que ela se aplica, descrevendo precisamente a semântica dos símbolos próprios da linguagem de aplicação. A descrição semântica dos símbolos segue um sistema de representação no qual cada símbolo deve ser descrito através de noções e impactos.

A elicitação da linguagem de aplicação possui como objetivo a identificação de termos que sejam próprios da linguagem em questão, ou seja, a identificação de palavras ou frases peculiares ao meio social da aplicação sob estudo. Após essa etapa, procura-se associar cada palavra ou frase a um significado.

São consideradas três fases para a coleta de fatos: identificação de símbolos da linguagem, identificação da semântica de cada símbolo e identificação das regras de produção de uma gramática que gera a linguagem de aplicação.

A validação interliga as três fases anteriores por meio de um ciclo de validação apoiado em três tipos de retroalimentação, a saber: necessidade de identificar novos símbolos, correções de noções e impactos, correção ou inclusão de regras.

A visão hipertexto é ortogonal à visão do modelo básico, LEL e à visão do modelo de cenário. No entanto, será focalizada somente a visão hipertexto a partir de um ponto de vista do cenário, uma vez que será o modelo mais enfatizado nesta seção.

Existem muitos relacionamentos a serem explorados entre cenários e outros componentes da estrutura de requisitos. A partir de cada cenário é derivado um ou mais nós hipertextos que são relacionados por relacionamentos hipertexto. Pode-se ter um nó classe para cada tipo de entidade definida no modelo de cenários.

A justificativa para que nós hipertextos sejam derivados a partir de cenários é que podem ser construídas diferentes visões do mesmo cenário, de acordo com o perfil ou tarefa do usuário. Essa abordagem é usual em abordagens de projeto hipermídia moderna e permite-nos, por exemplo, concentrar toda a informação desejada a partir de um cenário em um nó composto. O nó contém Título, Objetivo, Contexto, Recursos como atributos e o conjunto de episódios como partes do nó, alcançáveis por relacionamentos estruturais.

A visão de configuração é essencial para manter a rastreabilidade dos produtos e suas revisões. O LEL (Léxico Extendido da Linguagem), o BMV (Visão do Modelo Básico) são todos sujeitos a um controle de versão e de configuração. Em um dado momento, uma visão da estrutura pode ser requisitada baseada na configuração atual ou em configurações passadas.

Observa-se que a versão de cada modelo mantém as seguintes informações: data, hora, usuário (pessoa que faz a mudança), razões da mudança, acionamento da mudança, data do acionamento, autorização e tipo de mudança/entrada, modificação (entrada, modificação, exclusão).

A consistência da configuração é garantida pelas restrições de consistência determinadas por cada modelo. Assim, gerações de mudanças acionam um processo de checagem de consistência responsável pela consistência de uma dada configuração.

A abordagem citada nesta seção apresenta-se como um modelo fechado, sendo fundamental o conhecimento do LEL e todo o seu processo de elaboração, para que possa ser realizada alguma extensão ao modelo com alguma particularidade.

2.2.5 Estudo comparativo entre as abordagens de cenários selecionadas.

Um estudo comparativo entre algumas abordagens que utilizam cenários, explicadas nas seções anteriores é realizado nesta seção e sumarizado na tabela 2.2. Foram descritos a fase de Engenharia de requisitos da abordagem formal de Hsia et al, o modelo cíclico de Potts, a abordagem CREWS-L'Ecritoire e a abordagem de Julio César et al.

Como critérios de comparação foram escolhidos os produtos obtidos como resultado do processo do método, as fases desse processo, os principais conceitos em que se baseiam os métodos escolhidos, e as particularidades e características de cada método que utiliza cenários apresentam.

2.3 Abordagens que utilizam objetivos para elicitação e modelagem de requisitos

2.3 .1 Método GBRAM

Em [4,5,6] é apresentado o Método GBRAM (Método de Análise de Requisitos Baseados em Objetivos) que pressupõe que os objetivos não tenham sido previamente documentados ou explicitamente elicitados a partir dos *stakeholders*. Assim, o analista deve trabalhar a partir de todas as fontes de informação disponíveis como fluxos de processos ou de informação, declarações textuais de necessidades, e/ou fontes adicionais de informação como *transcripts* de entrevistas com os *stakeholders*, para determinar os objetivos do sistema desejado.

O Método GBRAM descrito na figura 2.14 envolve duas fases: a fase de análise de objetivos e a fase de refinamento de objetivos, produzindo como saída o DRS (Documento de Requisitos do Software), que provê uma comunicação não ambígua entre *stakeholders*, usuários, analistas e desenvolvedores, além de suportar evolução e validação de requisitos.

As atividades de análise de objetivo podem ser sumarizadas como segue:

- Exploração da documentação existente para a identificação inicial de objetivos;
- Identificação de objetivos, *stakeholders* e seus agentes responsáveis;
- Organização de objetivos de acordo com as relações de dependência e classificação dos objetivos conforme as condições alvo.

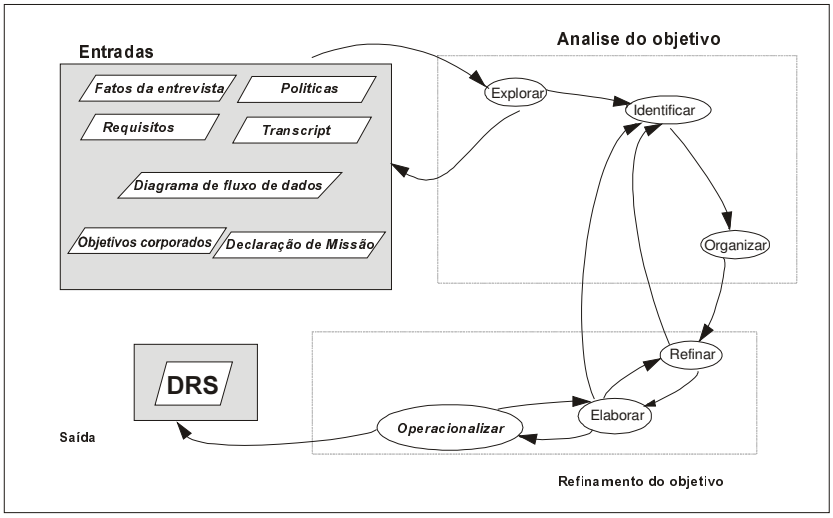

Figura 2.14: Atividades do Método GBRAM.

O Método GBRAM provê técnicas para suportar este processo de identificação do objetivo, que pode ser aplicada a vários tipos de descrições informais do sistema desejado. Para identificação dos objetivos são utilizadas técnicas dirigidas a perguntas e a da localização de palavras de ação.

Para identificar objetivos, pode ser utilizada uma técnica dirigida a questão onde cada declaração (ou pedaço de informação) é analisada e questionada através de perguntas tais como "Quais objetivos este fragmento da declaração exemplifica?" "Quais objetivos esta declaração bloqueia ou obstrui?".

Outro indicativo para identificar objetivos seria considerar todas as palavras de ação ou determinados tipos de verbo como, por exemplo, alocar, realizar, executar, satisfazer, arranjar, melhorar, garantir, trilhar, entre outros.

Os *stakeholders* são definidos como “alguém que reinvidica um interesse na empresa ou sistema". Observa-se que um *stakeholder* não é simplesmente um usuário do sistema, mas preferivelmente qualquer representação afetada pela execução ou prevenção de um objetivo particular.

Um *stakeholder* pode ser um cliente, um ator, um dono ou representante da organização. Um cliente é um beneficiário do sistema. Um ator é alguém que atualmente executa funções no sistema e um dono é um cliente no sentido contratual.

As perguntas utilizadas para identificação dos *stakeholders* consistem em "Quem ou o que é reinvidicado neste objetivo?" "Quem ou o que se ganha ou perde com a realização ou prevenção deste objetivo?".

Uma vez especificados os objetivos e os *stakeholders*, o próximo passo consiste em associar os objetivos a seus agentes responsáveis. Os agentes são responsáveis pela realização e/ou satisfação dos objetivos dentro de uma organização ou sistema. É importante observar que apenas um agente é responsável por garantir a realização de um objetivo em um dado tempo; entretanto, diferentes agentes podem ser responsáveis pela realização do mesmo objetivo em diferentes tempos.

Geralmente a pergunta utilizada para identificação dos agentes consiste em "Quem ou quais agentes [é/deveria ser/poderia ser] responsável por este objetivo?”. Uma vez identificados os objetivos, *stakeholders* e os agentes, o método descrito sugere a classificação

dos objetivos de acordo com as condições alvo. No GBRAM, os objetivos são classificados em objetivos de manutenção e objetivos de realização, sendo que os objetivos de manutenção podem definir o escopo dos objetivos de realização.

Os objetivos de manutenção sugerem um estado contínuo dentro do sistema; estes são identificados considerando cada objetivo e perguntando "Este objetivo garante que alguma condição seja mantida verdadeira por todas as outras operacionalizações de objetivos?" "Este objetivo afeta decisões em vários níveis dentro da organização?" e "Este objetivo requer um estado contínuo dentro do sistema?". Além dessas perguntas, outro auxílio à descoberta de objetivos de manutenção consiste em utilizar palavras-chaves como "prover" e "fornecer", entre outros.

Os objetivos de realização mapeiam para ações que ocorrem no sistema e auxiliam na identificação dos requisitos funcionais necessários à satisfação dos *stakeholders* e clientes. Esses objetivos são descobertos através de questionamentos tais como "A realização deste objetivo depende da realização de outro objetivo?" "A realização de outro objetivo depende da conclusão deste objetivo?".

Uma vez descobertos os objetivos, são estabelecidas relações de dependência entre estes objetivos. O método GBRAM provê a dependência de precedência que se subdivide em dependência de contrato e dependência de agente, a seguir explicadas.

Relações de dependência existem entre pares de objetivos e significa que um dado objetivo é dependente de outro objetivo para sua realização. A relação de dependência realizada no Método GBRAM para ordenar os objetivos é a dependência de precedência, que define que um objetivo G1 deve ser realizado antes do objetivo G2, sendo expresso por G1 < G2. A pergunta a ser realizada para cada objetivo nesta etapa é "Quais objetivos devem seguir este objetivo?" "Quais objetivos dependem da disponibilidade desta informação para sua realização?".

Uma dependência de contrato entre objetivos G1 e G2, onde o objetivo G2 deve ser realizado se o objetivo G1 ocorre, é expresso por G1 → G2. A dependência entre agentes consiste em, para que um agente realize um dado objetivo, outro agente pode ter que primeiro realizar outro objetivo.

Posteriormente à identificação dos relacionamentos entre objetivos e especificadas as dependências, o próximo passo do Método GBRAM consiste em construir uma topografia de objetivo que consiste na representação dos aspectos do DRS (Documento de requisitos de software) expressos na forma de hierarquia que pode ser mapeada para o DRS. Essa topografia de objetivos possui dois estilos: manual e hierárquico, sendo este último o mais indicado, pois auxilia o analista a descobrir requisitos e relacionamentos escondidos.

No que concerne às atividades de refinamento do objetivo, estas podem ser sumarizadas como segue:

- Refinamento de um conjunto objetivo com a diminuição do tamanho do conjunto objetivo;
- Elaboração de cenários para descobrir objetivos e requisitos escondidos;
- Operacionalização dos objetivos em requisitos operacionais.

Na fase de elaboração dos objetivos são identificados os obstáculos dos objetivos, considerando as possíveis maneiras de falhas dos objetivos, e como estes podem ser bloqueados facilitando a antecipação de casos excepcionais. Os obstáculos são identificados por meio de perguntas como

- "De quais outros objetivos ou condições este objetivo depende?".
- "Pode o agente responsável pela falha do objetivo realizar o objetivo?".
- "Pode a falha de outro objetivo completamente causar o bloqueio deste objetivo?".
- "Se este objetivo é bloqueado, quais são as conseqüências?".
- "Quais outros objetivos ou condições este objetivo depende?" Ou através das palavras-chaves “não”, “entretanto” e “quando”.

Uma vez especificados os obstáculos de objetivos, os analistas devem considerar os cenários possíveis que são prováveis para cada obstáculo. Cenários são descrições comportamentais de um sistema e seu ambiente que surgem a partir de situações restritas. Eles oferecem uma maneira natural para descrever circunstâncias excepcionais, especiais, auxiliando os analistas a descobrirem objetivos escondidos ou aspectos necessários a uma resolução adicional que poderia de outra forma ser negligenciada.

Para identificar cenários, são sugeridas por este método as seguintes perguntas "O que acontece se este objetivo não é executado?" "Por que este objetivo não foi executado?" "Quais as circunstâncias sob a qual este obstáculo ocorre?" "Por que este obstáculo ocorre?".

Realizada a especificação de objetivos, esta informação deve ser operacionalizada e traduzida em expressões de linguagem natural de requisitos no DRS. Durante a operacionalização, as ações descritas pelos *stakeholders* e extraídas a partir da documentação disponível são relacionadas de volta aos objetivos.

Assim, os objetivos operacionalizados, agentes responsáveis, *stakeholders*, restrições, obstáculos e cenários são mapeados em ações firmadas em um conjunto de esquemas objetivo (modelos que especificam os relacionamentos entre objetivos e agentes em termos de eventos que causam uma mudança de estado).

O conjunto de esquemas objetivos é mapeado para um documento de requisitos de software, incorporando toda a informação adquirida durante a análise e elaboração dos objetivos.

2.3.2 Modelo de objetivos (A G. Sutcliffe & N.A. M. Maiden)

Em [49] é proposto um modelo que suporta três atividades no processo de Engenharia de requisitos: investigação de problema/política; refinamento política/objetivo e análise do objetivo. Objetivos são decompostos em níveis de detalhe sucessivos para formar uma árvore de profundidade arbitrária. Os nós pais terão relacionamentos AND com os nós filhos, assim como dois ou mais subobjetivos suportarão a realização de um objetivo de nível mais baixo**.** Entretanto, pode haver ocasiões em que relações OR são utilizadas para representar alternativas.

Com o propósito de estruturar o processo de análise de requisitos, são identificados três níveis de objetivo ou declarações de requisitos de política.

1 - Nível de política: este nível descreve o que deve ser realizado, por exemplo, melhorar a satisfação do cliente. Observa-se que nenhuma restrição particular está localizada na expressão de objetivos neste nível.

2 - Nível de objetivo funcional: tem expressões lingüísticas que contém alguma informação sobre o que pode ser feito para realizar objetivos no nível político.

3 - Nível de objetivo do domínio: neste nível descrições de objetivos são refinadas por três tratamentos: a) análise dos engenheiros de requisitos em conjunto com os usuários sobre a necessidade de prover aos objetivos um processo automatizado ou se deve ser realizada uma decisão gerencial. b) *templates* são utilizadas para refinar expressões de objetivos, onde são relacionadas a visão funcional dos objetivos a um modelo de domínio em termos de agentes, objetos, processos de informação; c) objetivos, combinados com requisitos não funcionais expressos como um estado que o sistema tem que realizar.

Durante a análise do objetivo, estes podem ser classificados em seis classes conforme é descrito a seguir:

1 - Objetivos de estado positivo - estes objetivos indicam estados que devem ser realizados, sugerindo especificações de funcionalidade adicionais para garantir que o estado correto seja realizado e, adicionalmente, que o estado incorreto seja evitado.

2 - Objetivos de estado negativo - neste caso, o estado expresso deve ser evitado, e objetivos são expressos como invariantes ou restrições no comportamento do sistema. Estados negativos, assim como estados positivos indicam refinamento de especificação, entretanto a ênfase está no controle dos procedimentos para garantir um estado que não aconteça. Um estado positivo pode ser executado pela prevenção da ocorrência de estados negativos.

3 - Objetivos de estado alternativo (OR) - dois ou mais estados são possíveis, mas a escolha de qual estado a ser aplicado depende da entrada do sistema. Neste caso, a escolha do estado é postergada até a execução quando estiver disponível a informação adicional.

4 - Objetivos de reparo-exceção: neste caso nada pode ser feito sobre o estado que um objeto realiza, embora ele seja insatisfatório e, portanto deva de alguma maneira, ser corrigido. Este objetivo é um subtipo do estado de objetivo positivo em que o estado do objeto é insatisfatório e alguma ação corretiva deva ser tomada.

5 - Objetivos de feedback- estes são associados com um estado desejado e uma faixa de exceções que podem ser toleradas. O sistema tem que monitorar a miscelânea de um ou mais, usualmente uma população de objetos e tomar ações corretivas por meio de alguma norma prédefinida.

Este tipo é uma variante dos estados de objetivo positivo; entretanto, o objetivo é declarado como um alvo com uma faixa de tolerância, um requisito não funcional, e uma ação é necessária para retornar o sistema para o conjunto alvo.

6- Objetivos de estado misturado: estes objetivos requerem que o sistema mantenha algum equilíbrio entre dois estados. Esses objetivos estão entre subtipos de estados de objetivos selecionados exceto que ele é uma população de objetos que podem ser controlados.

Dois subestados podem existir, e a distribuição de instâncias entre os dois subestados tem que ser realizada. Este objetivo implica que o critério deva ser declarado para determinar proporcionalidade entre os dois subestados. O estado a ser executado depende da freqüência do evento.

A estas seis classes de objetivos que foram descritas podem ser adicionadas outras classes. O propósito desta classificação é sugerir análise de requisitos em determinadas direções, de acordo com a classe de objetivos.

O propósito final desta classificação é relacionar a análise do objetivo descrita acima a modelos genéricos que são incorporados na teoria do domínio. A teoria propõe classes de modelos de domínio que tem uma base na semântica objeto-estado. Se as classes de objetivos podem ser comparadas ao modelo de domínio apropriado e o modelo descreve um estado adaptável para realizar o objetivo, então o conhecimento de domínio pode ser reusado para elaborar adicionalmente a especificação.

2.3.3 Aquisição de requisitos direcionada a objetivos (Meta-modelo KAOS)

Em [18] é proposto um metamodelo para capturar requisitos iniciais e uma estratégia para conduzir o processo de aquisição de requisitos. Os requisitos considerados referem-se ao sistema composto inteiro - i.é, a parte a ser automatizada, seu ambiente físico e a maneira que ambas as partes tem que cooperar. Essa abordagem para aquisição de requisitos foi desenvolvida no projeto KAOS e consiste em três componentes:

1 - O modelo conceitual para adquirir e estruturar modelos de requisitos com uma linguagem de aquisição associada.

2 - Um conjunto de estratégias para elaborar modelos de requisitos nesta estrutura.

3 - Assistente automatizado para prover guias no processo de aquisição de acordo com as estratégias.

O modelo conceitual provê um número de abstrações em termos dos quais modelos de requisitos são adquiridos. O metamodelo para aquisição de requisitos pode ser representado como um grafo conceitual onde nós representam abstrações e setas representam *links* de estruturação. A estratégia de aquisição define uma série de passos para aquisição de componentes do modelo de requisitos como instâncias dos componentes do metamodelo. O assistente de aquisição é auxiliado a prover suporte automatizado na adoção de uma estratégia de aquisição ou outra.

Essa abordagem para aquisição de requisitos envolve três níveis de modelagem: Nível meta, onde são elaboradas as abstrações independentes do domínio. Nível domínio, onde são definidos conceitos específicos do domínio da aplicação e nível instância onde são definidas instâncias específicas dos conceitos do nível do domínio.

Na figura. 2.15 é apresentada uma parte significativa do metamodelo KAOS, onde são descritas as várias abstrações envolvidas na parte do metamodelo relevante para a estratégia de aquisição direcionada a objetivo.

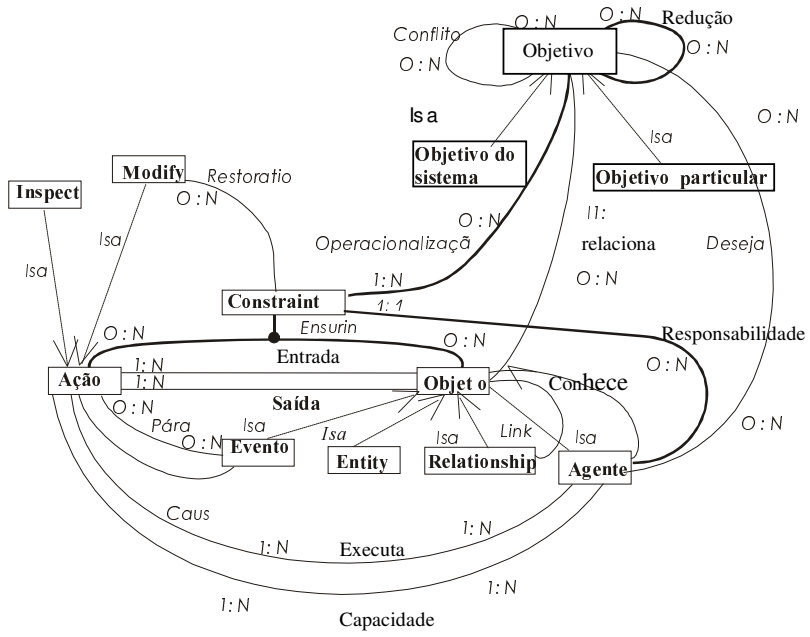

Figura. 2.15: Uma parte do Metamodelo conceitual da abordagem KAOS.

O metamodelo KAOS é um modelo conceitual para o nível meta, assim consistindo de conceitos de nível meta, relacionamentos, atributos e restrições. A linguagem de aquisição utilizada é uma estrutura de dois níveis. Um nível mais externo para declarar conceitos de nível de domínio em termos dos componentes do metamodelo, e um nível mais

interno para expressar valores para alguns meta-atributos. O nível de declaração mais externa possui uma estrutura entidade-relacionamento produzindo a estrutura de banco de dados de requisitos. O nível interno corresponde a uma lógica de 1ª ordem temporal. A seguir são descritas as várias abstrações envolvidas na parte do metamodelo (vide figura. 2.15) relevante para a estratégia de aquisição direcionada a objetivos.

- Um objeto é algo de interesse que pode ser referenciado nos requisitos. Possui meta-atributos, como por exemplo, nome e descrição informal. Os meta-conceitos "entidade", "relacionamento", "evento" e "agente" herdam todos os aspectos do meta-conceito "objeto".
- Uma "entidade" é um objeto autônomo. Suas instâncias podem existir independentemente de outras instâncias "objeto".
- Um "relacionamento" é um objeto subordinado, i.e, sua existência depende da existência das instâncias de objetos correspondentes ligados pelo relacionamento.
- Um "evento" é um objeto instantâneo, i.e, suas instâncias existem apenas em um estado.
- Ação: é uma relação matemática sobre objetos. Aplicações de ações definem transições de estados. Entre os meta-atributos do meta-conceito "ação" estão incluídas as pré-condições, condições de acionamento e pós-condições, assim como as condições de parada e duração.
- O meta-conceito ação é ligado ao objeto através dos meta-relacionamentos "Entrada/Saída" e ao meta-relacionamento "Evento" através do meta-relacionamento "causa/pára".
- Duas especializações de ações são distinguidas no meta-modelo: ações de inspeção e de modificação. Para uma ação de modificação de saída significa que instâncias dos objetos são criados, deletados e atualizados. Os meta-relacionamentos de entrada e saída possuem como meta-atributos argumento e resultado.
- Agente: é um objeto processado por algumas ações. Agentes podem ser humanos, dispositivos físicos ou programas que existem ou são desenvolvidos na parte automatizada do sistema composto. O meta-conceito "Agente" possui dois meta-relacionamentos com "ação": "capacidade" e "executa". Dessa forma, um agente é definido por dois conjuntos de ações: o conjunto de ações que ele pode executar e o conjunto de ações que ele deve executar.

- Um outro meta-relacionamento existente é denominado "conhece" posicionado entre os meta-conceitos “Agente” e “Objeto”.
- Objetivos: refere-se a um objetivo não operacional a ser executado pelo sistema composto. O meta-relacionamento “relacionamento” liga os meta-conceitos “Objetivo” e “Objeto”. Objetivos podem ser distinguidos entre objetivos do sistema e objetivos particulares. Para refinamento dos objetivos, é utilizado o meta-relacionamento “Redução” onde combinações alternativas de subobjetivos são construídas.
- Outro meta-relacionamento utilizado é o de “conflito”, onde objetivos não podem ser executados juntos. Nesse caso, um meta-atributo “Prioridade” pode ser ligado ao meta-conceito “objetivo”.
- O meta-relacionamento “Deseja” é utilizado entre o meta-conceito “Agente” e o meta-conceito “Objetivo”.
- Restrições: é um objetivo não operacional a ser executado pelo sistema composto. Uma restrição é formulada em termos dos objetos e ações disponíveis para algum agente no sistema.
- “Objetivos” são operacionalizados através de “restrições”. Um *link* entre objetivos e restrições é capturado no meta-relacionamento “operacionalização”. Um objetivo pode ser operacionalizado através de algumas combinações alternativas de restrições.
- Restrições podem ser reduzidas da mesma maneira como objetivos. Elas podem ser conflitantes e ter prioridades associadas.
- O meta-conceito “restrição” possui duas especializações: “HardConstraint” e “SoftConstraint” onde as primeiras nunca podem ser violadas e as segundas podem ser temporariamente violadas. Nesse caso são necessárias ações específicas para restaurá-las.

Os processo de aquisição são guiados por estratégias e modelos de domínio. Estratégias definem maneiras específicas de adquirir instâncias dos vários nós e *links* pertencentes ao grafo do metamodelo. A estratégia descrita é direcionada para objetivo consistindo nos seguintes passos:

1 - Adquirir estrutura do objetivo e identificar objetos relacionados: são adquiridas instâncias do meta-conceito “objetivo” e do meta-relacionamento “redução”. A elaboração da estrutura do objetivo consiste em identificar os objetivos do sistema, sua categoria e padrões, associando-os com os objetivos fontes que eles reduzem, identificar os objetos referenciados

pelos objetivos e elaborar uma definição preliminar de seus aspectos e identificar possíveis conflitos entre objetivos do sistema, i. e, definir instâncias do meta-relacionamento “conflito”.
2 - Identificar agentes potenciais e suas capacidades: consiste em identificar as instâncias direcionadas a objetivo do meta-conceito “agente”, meta-relacionamento “capacidade” e meta-conceito “ação”.
3 - Operacionalizar objetivos em restrições: os objetivos folha na estrutura objetivo, elaborados no passo um são transformados em objetivos do sistema formulados em termos de objetos e ações.
4 - Refinar objetos e ações: neste passo, o analista define os objetos e ações e completa a descrição dos objetos e ações já identificados. Também são definidos os relacionamentos de responsabilidade.
5 - Derivar ações e objetos para garantir restrições: neste passo, ações e objetos são derivados para garantir que todas as restrições requeridas sejam satisfeitas.
6 - Identificar responsabilidades alternativas: para cada restrição obtida no passo três, são identificados os vários *links* de responsabilidade. A identificação é baseada nas capacidades dos agentes determinados no passo dois.
7 - Associar ações a agentes responsáveis: *links* de execução são efetivamente associados a agentes para as várias ações elaboradas nos passos dois e quatro na base do *link* alternativo de responsabilidade estabelecido no passo seis. Uma ação é associada a um agente apenas se o agente tiver sido determinado entre os candidatos alternativos que tomam responsabilidade sobre as restrições que as ações garantem.

O metamodelo KAOS suporta a noção de decomposição de objetivos, inclusive decomposição alternativa para comportamentos que podem satisfazer os mesmos objetivos básicos. Ela inclui os relacionamentos de conflitos entre objetivos, assim como os de redução and e or.

O principal problema percebido dentro dessa abordagem refere-se ao número de passos durante o processo, difíceis de executar. Na aplicação do método, foi encontrada dificuldade para gerar a estrutura objetivo (passo um), originário da identificação de subobjetivos de um objetivo, e saber quando todos os subobjetivos requeridos foram identificados.

A aquisição de ações e objetos para um dado sistema é completa no passo quatro. Entretanto, não existe garantia de que estes irão corresponder às restrições. Dessa

forma, no passo cinco cada restrição deve ser verificada com as ações, para identificar se as transições de estados definidas pelas ações correspondem às restrições, o que pode revelar que uma precondição, póscondição ou invariante associada com uma ação deva ser reforçada para garantir que a restrição seja correspondente.

Para essa atividade são oferecidas ao analista algumas regras de inferência para derivar asseguradas precondições, pós-condições e invariantes. No entanto, é difícil para o analista selecionar a regra manualmente e aplicá-la de modo correto.

Para dar suporte à Metodologia KAOS, em [19] pode ser encontrada uma ferramenta de aquisição de requisitos desenvolvida no ambiente GRAIL/KAOS. O Kernel GRAIL combina uma visão gráfica, uma visão textual, uma visão abstrata da sintaxe e uma visão da base de objetos das especificações.

Em [31] é ilustrado a integração de obstáculos na metodologia KAOS. Um obstáculo define um conjunto de comportamentos não desejados. A análise de obstáculos precisa ser realizada o mais cedo possível dentro do processo de engenharia de requisitos, ou seja, no nível de objetivos, de modo a se obter especificações de requisitos mais completas e realistas.

Uma vez identificados os obstáculos, estes precisam ser resolvidos, resultando em uma estrutura de objetivo modificada. Entre as estratégias de resolução de obstáculos, encontra-se a transformação de objetivos, a introdução de novos objetivos para que sejam evitados os obstáculos e a investigação de projetos alternativos com diferentes associações de agentes de modo que o obstáculo não mais exista.

Em [33] essas idéias são expandidas e são ilustradas várias técnicas formais e heurísticas para geração e refinamento de obstáculos a partir de especificações de objetivos e propriedades de domínio. A geração de resolução de obstáculos é realizada através de várias estratégias para eliminar, reduzir ou tolerar os obstáculos.

O gerenciamento de conflitos também foi explorado dentro da metodologia KAOS. Em [32] são propostas técnicas formais e heurísticas para identificação de conflitos e divergências a partir de especificações de objetivos/requisitos e a partir de propriedades conhecidas sobre o domínio. São discutidas também várias técnicas para resolução de

conflitos e divergências, que resultam em uma modificação na estrutura do objetivos ou de objetos.

2.3.4 Comparação entre as abordagens selecionadas que utilizam objetivos como técnica de elicitação e modelagem de requisitos.

Um estudo comparativo entre algumas abordagens que utilizam objetivos, explicadas nas seções anteriores é realizado nesta seção e sumarizado na tabela 2.3. Foram descritos o Método GBRAM, o modelo de objetivos de Sutcliffe et al e o metamodelo KAOS.

Como critérios de comparação foram escolhidos os produtos obtidos como resultado do processo do método, as fases desse processo, os principais conceitos em que se baseiam os métodos escolhidos e as particularidades e características de cada uma das abordagens que utilizam objetivos apresentam.

3. MÉTODO PROPOSTO PARA ELICITAÇÃO E MODELAGEM DE REQUISITOS

O método proposto para elicitação e modelagem de requisitos se destina a acoplar objetivos e casos de uso numa decomposição top-down em diferentes níveis de abstração [12]. Ele combina o uso de dois conceitos: casos de uso e objetivo em um pedaço de requisitos (RC). Um RC é um par <objetivo, caso de uso>, relacionados através de relacionamentos AND, OR e de refinamento. Esses três tipos de relacionamentos entre RCs conduzem a uma organização hierárquica de RCs em três níveis de abstração: o funcional, o comportamental e o físico.

O nível funcional focaliza-se nos serviços e casos de uso associados, onde um objetivo de serviço é acoplado ao conjunto de casos de uso necessários para executar um serviço, formando o RC funcional. Cada um dos casos de uso é adicionalmente detalhado pelos RCs comportamentais no nível comportamental.

O nível comportamental se destina a identificar os objetivos dos casos de uso descobertos no nível anterior com a operacionalização dos casos de uso através de cenários estruturais. Dessa forma o RC comportamental acopla um objetivo funcional a um cenário estrutural.

O objetivo funcional expressa uma maneira de prover uma funcionalidade identificada no nível anterior e, portanto estabelece um *link* hierárquico com o pedaço funcional. O cenário estrutural descreve o fluxo de ações do usuário e do sistema necessários para executar o objetivo funcional.

O nível físico detalha uma maneira possível em que o sistema pode internamente executar uma interação identificada no cenário estrutural. O RC físico associa um objetivo do sistema a um cenário interno do sistema.

O objetivo do sistema expressa uma maneira de executar uma ação identificada no cenário estrutural e o cenário interno do sistema descreve o fluxo de interações entre os objetos do sistema para executar o objetivo do sistema

3.1 Relacionamentos entre RCs

- **Relacionamentos AND**

Estes relacionamentos são utilizados entre modelos de casos de uso formando um pacote de serviços no nível funcional e, entre os cenários estruturais complementares no nível comportamental descrevendo um serviço e, por ultimo entre os cenários internos que compõem um cenário estrutural.

- **Relacionamentos OR**

Estes relacionamentos representam maneiras alternativas de executar o mesmo objetivo, sendo expressos entre RCs dentro do mesmo nível de abstração representando variações dos cenários normais ou cenários excepcionais no nível funcional, comportamental ou no nível físico.

- **Refinamento**

Relacionamentos de refinamento relacionam RCs em diferentes níveis de detalhe. Ele é utilizado para refinar um caso de uso do nível i em cenários estruturais no nível i + 1. Além disso, ele pode ser também empregado para especificar (refinar) os cenários estruturais em cenários internos do sistema.

Os relacionamentos de refinamento são definidos a partir de um RC funcional para um RC comportamental e, de um RC comportamental para um RC físico, como ilustra a figura 3.1.

A numeração utilizada pelos RCs refere-se à colocação de índices ou um subitem, de acordo com o tipo de relacionamento. Para os relacionamentos alternativos, é acrescentada à notação de RCs um expoente que inicia em 1 e vai sendo incrementado à medida que novos RCs alternativos vão sendo descobertos. Assim, no nível comportamental para o RC1.1 temos um RC cujo cenário é normal e ligados a ele tem-se vários RCs cuja numeração será $RC1.1^1$, $RC1.1^2$, e assim por diante.

Para os relacionamentos do tipo “AND”(composição), o último algarismo do RC é incrementado. Assim para o RC 1.1, teremos a ele ligados pelo relacionamento AND, RC1.2, RC1.3, e assim por diante.

No caso dos relacionamentos de refinamento é acrescentado um novo nível à numeração e, portanto uma nova numeração. Dessa forma para o 1º nível funcional temos RC1, onde por meio da estratégia de refinamento obtém-se o RC1.1 no nível comportamental e o RC1.1.1 no nível interno. Para o RC2, teremos o RC2.1 no nível comportamental e o RC2.1.1 no nível interno. Os relacionamentos AND e OR ocorrem dentro do mesmo nível, enquanto o relacionamento de refinamento ocorre de um nível para o outro.

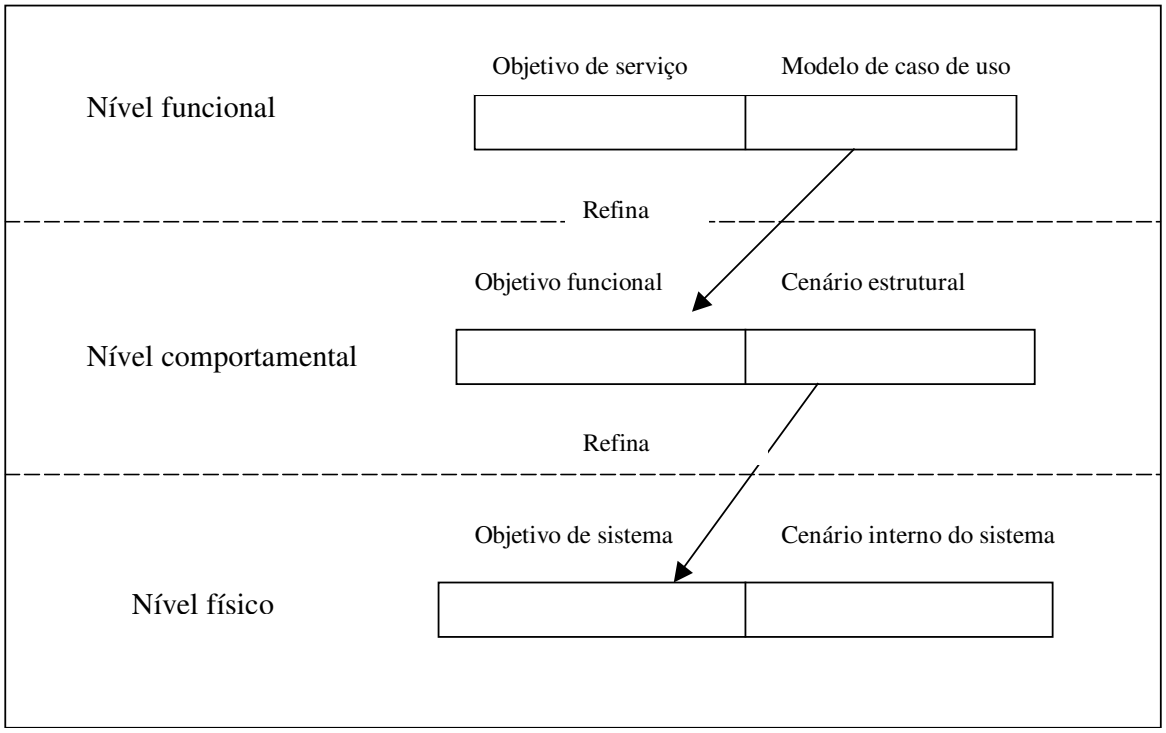

Figura. 3.1. Estrutura de refinamento entre os três níveis de abstração.

3.2 Fases do Método para elicitação e modelagem de requisitos

O método proposto consiste de um modelo de processo e um modelo de produto representando o processo de desenvolvimento do produto. O mapa do processo (vide figura 3.2) mostra várias fases: Elicitar um objetivo, especificar caso de usos, elaborar cenários, reorganizar objetivos e cenários, refinar objetivos e cenários e operacionalizar objetivos e cenários.

3.2.1 Modelo de processo

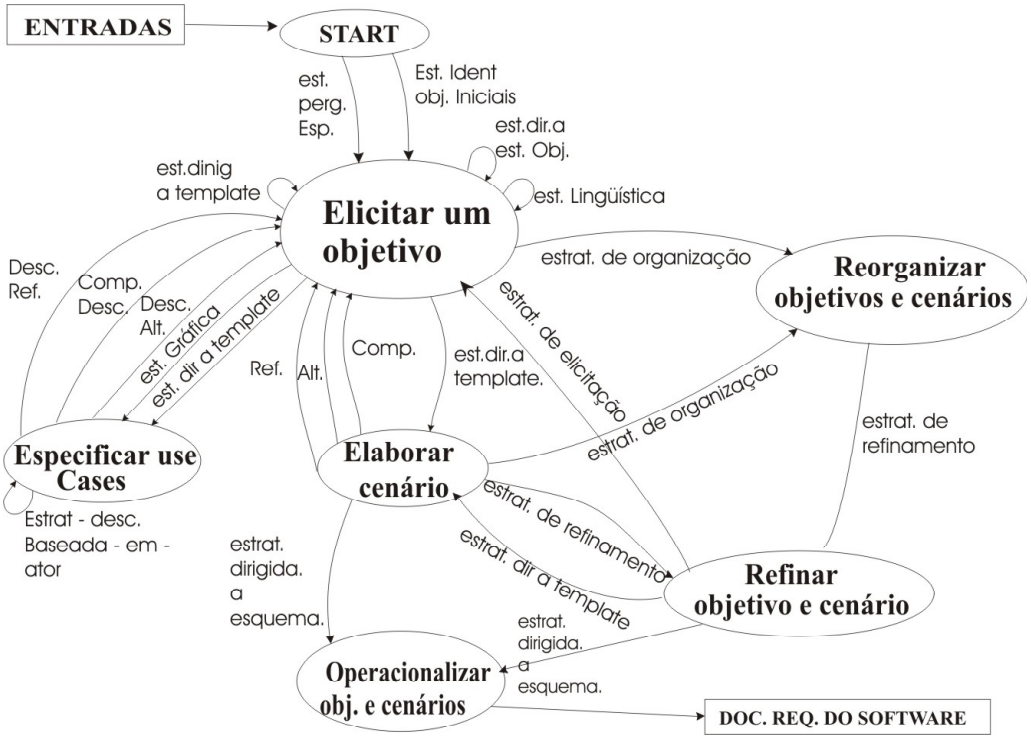

Figura 3.2: Modelo de processo do Método Proposto.

A seguir são explicadas cada uma das fases, numeradas de acordo com a seqüência que devem ser executadas, assim como as estratégias relacionadas.

1 - Elicitação de objetivos de serviço

Nessa fase são identificados, inicialmente os objetivos de serviço por meio da estratégia de identificação dos objetivos iniciais e/ou estratégia de perguntas específicas, definidas a seguir:

- ✓ Estratégia de identificação dos objetivos iniciais: consiste em extrair os objetivos, a partir da análise da documentação existente ou das entrevistas realizadas entre clientes, usuários e engenheiro de requisitos. Essa análise da documentação existente é realizada examinando-se descrições de processos, fluxogramas, ou quaisquer documentos informais que existam acerca do domínio da aplicação onde o sistema será desenvolvido, à procura por palavras

(verbos) de ação, uma vez que os usuários tendem a expressar seus requisitos através de operações.

- ✓ Estratégia de perguntas específicas: concerne em estabelecer perguntas específicas de maneira a identificar os objetivos para o primeiro nível de nossa abordagem.

A principal diferença entre a estratégia de identificação dos objetivos iniciais e identificação de perguntas específicas é que a primeira supõe a existência de alguma documentação, enquanto para a segunda não é necessariamente imprescindível a existência de documentos. Além dessas duas estratégias definidas acima, outras estratégias que são utilizadas para elicitação de objetivos são:

- ✓ Estratégia dirigida a template: propõe uma template para elicitação dos objetivos, onde essa template consiste no nome do objetivo, identificação do objetivo e número do RC à qual ele pertence.
- ✓ Estratégia dirigida a estrutura do objetivo: a aplicação dessa estratégia consiste em combinar valores alternativos para os diversos parâmetros, resultando em objetivos possíveis a serem escolhidos pelo usuário para elicitação. A estrutura do objetivo adotada consiste em um verbo + parâmetros, onde parâmetros:: objeto + maneira +[de]/ [para] [fonte] [destino], sendo fonte e destino parâmetros opcionais.
- ✓ Estratégia lingüística: visa à fase de formalização posteriormente definida dentro da abordagem.

Vamos considerar, com o intuito de facilitar o entendimento da proposta, um exemplo de um sistema de bibliotecas. Com a aplicação das estratégias para identificação dos objetivos iniciais e da estratégia de perguntas específicas, foram encontrados os seguintes objetivos de serviço: G1 – Proceder Circulação, G2 – Manter Cadastros e G3 – Efetuar Consultas.

2 – Especificar casos de uso

Essa etapa é realizada para cada um dos objetivos de serviço identificados na fase anterior. Para especificar um caso de uso são propostas três estratégias:

- ✓ Estratégia dirigida a template: propõe especificar casos de uso por meio de uma template, composta pelo nome do ator[1], código do caso de uso, nome do caso de uso, [identificação do ator] e código do objetivo.

- ✓ Estratégia gráfica: os casos de uso são construídos conforme a notação de Jacobson [27], utilizando o conceito de caso de uso definido em Regnell et al [42], onde a descrição de casos de uso não permite múltiplos atores, existindo uma visão única para cada ator envolvido no caso de uso.

- ✓ Estratégia baseada em ator: propõe identificar os atores do sistema como um meio de identificar os casos de uso

Dando continuidade ao nosso exemplo para ilustrar as fases do método, será aplicada a estratégia dirigida a template para cada um dos objetivos de serviço identificados na etapa anterior.

Para o objetivo G1: Proceder Circulação, os casos de uso pertencentes a este objetivo foram especificados utilizando as seguintes templates:

Nome do ator: Bibiliotecário
Código do caso de uso: UC1
Nome do caso de uso: Realizar empréstimo
[Código do ator]: A1
Código do objetivo: G1

Nome do ator: Bibiliotecário
Código do caso de uso: UC2
Nome do caso de uso: Realizar devolução
[Código do ator]: A1
Código do objetivo: G2

Nome do ator: Bibiliotecário
Código do caso de uso: UC3
Nome do caso de uso: Efetuar renovação
[Código do ator]: A1

[1] Ator, dentro do método proposto, significa alguém que executa interações com o caso de uso.

Código do objetivo: G3

Após especificados os casos de uso, devem ser identificados os objetivos funcionais, correspondentes ao segundo nível de abstração de nossa abordagem.

3 – Elicitação de objetivos funcionais

Nesse caso, podem ser consideradas três estratégias para obtenção dos objetivos funcionais, a seguir descritas.

- ✓ Estratégia de descoberta do refinamento: essa estratégia é aplicada sobre os casos de uso originando os objetivos funcionais, que são operacionalizados pelos cenários estruturais.
- ✓ Estratégia de descoberta alternativa: essa estratégia permite obter variações para os objetivos funcionais no nível comportamental.
- ✓ Estratégia de composição: permite identificar os objetivos complementares para um dado objetivo e auxilia a identificar um pacote de serviços para um dado sistema.

De acordo com a estratégia de refinamento, cada caso de uso origina um objetivo funcional. Dessa forma, o caso de uso "Realizar empréstimo" origina no nível comportamental o G1.1 : Realizar empréstimo de uma maneira norma, que é operacionalizado pelo cenário C1.1.

4 – Especificação de cenários estruturais

Durante essa fase, os cenários são escritos segundo algumas regras de estilo e conteúdo definidas em [2] seguindo a estratégia dirigida a template, composta pelo código do cenário, agente, pré e pós-condições, ações e código do objetivo.

Observa-se que existe um elemento da *template* Ator/Agente onde se fazem necessárias algumas explicações. Para o nível funcional, utiliza-se o "Ator" definido como "Alguém que executa interações com o caso de uso". Para o nível comportamental e físico, utiliza-se o termo agente, definido como "aquele que é responsável pela execução do cenário". No caso do nível interno do sistema, o agente será por *default* o sistema, uma vez

que este nível descreve as interações entre os objetos do sistema para executar um objetivo do sistema.

As pré-condições tem um formato que consiste em uma estrutura normalmente condicional ou iterativa. A condicional é realizada com *if*, como por exemplo, *if* código de acesso do usuário = válido; *if* senha do usuário = válida; *if* código do cardápio= válido. No caso da iterativa, é utilizado o enquanto*,* como, por exemplo, enquanto código do cardápio for válido.

As pós-condições consistem em um estado obtido após a realização do cenário. Possuem um formato de um *DO ação* como, por exemplo*. DO* status do cardápio = "recebido".

Dando prosseguimento ao nosso exemplo, os cenários estruturais serão elicitados por uma template, ilustrada a seguir.

Código do cenário: C1.1

Agente: Bibiliotecário

Pré-condições: if código de acesso = válido

If Situação do usuário = OK

If status do livro = disponível

Ações

1. O usuário entra com código de acesso
2. O sistema valida o código de acesso
3. O usuário entra com código do usuário
4. O sistema verifica a situação do usuário
5. O usuário entra com código do livro
6. O sistema verifica se o livro pode ser emprestado
7. O sistema exibe mensagem “Empréstimo efetuado com sucesso” para o usuário.

Pós-condições: DO Status do livro = emprestado

5 – Elicitação de objetivos do sistema

No nível comportamental, a elicitação de novos objetivos é realizada utilizando três diferentes estratégias:

- ✓ Estratégia-descoberta-alternativa: permite identificar todos os objetivos alternativos para um dado objetivo.
- ✓ Estratégia da descoberta da composição: essa estratégia permite identificar os objetivos complementares para um dado objetivo.
- ✓ Estratégia de refinamento: no nível comportamental, durante a realização dessa estratégia são identificados os objetivos do sistema, pertencentes ao nível físico, oriundos das interações ocorridas no cenário estrutural. Assim, observa-se que ela ocorre a partir de um RC comportamental para um RC físico.

Para cada ação do sistema pertencente a um cenário estrutural, origina-se um objetivo do sistema. Assim, para a ação do sistema "O sistema verifica situação do usuário" origina-se o objetivo G1.1.2 : Verificar situação do usuário para empréstimo, pertencente ao nível físico

6– Especificação de cenários físicos

Durante essa fase, os cenários são escritos seguindo a estratégia dirigida a template, composta pelo código do cenário, agente, pré e pós-condições, ações e código do objetivo, a seguir exemplificada.

Código do cenário: C1.1.2
Agente: Bibiliotecário
Pré-condições: if código de usuário digitado
Ações

1. Abrir tabela de empréstimo no BD.
2. Pesquisar situação por código do usuário na tabela
3. Se situação = OK , então mostrar mensagem " Usuário apto para empréstimo"
4. Senão exibir mensagem "Usuário não habilitado"

5. Fechar tabela

Pós-condições: DO situação do usuário para empréstimo verificada

7 – Reorganização de objetivos e cenários

A fase de reorganização de objetivos e cenários caracteriza-se por serem estabelecidas relações de dependência entre objetivos e cenários, que são reorganizados de modo a satisfazer os critérios definidos.

Uma das relações de dependência utilizada entre objetivos foi a relação de dependência por precedência. Uma dependência de precedência entre objetivos G1 e G2, onde objetivo G1 deve ser executado antes do objetivo G2, é expressa por G1 < G2. As relações de precedência são identificadas para cada objetivo perguntando-se "Quais objetivos são pré-requisitos para este objetivo?" As respostas para estas questões facilitam a organização de objetivos com os objetivos pré-requisitos listados a priori para um dado objetivo.

Outra dependência de precedência definida entre objetivos é a dependência de contrato entre objetivos G1 e G2, onde objetivo G2 deve ser executado obrigatoriamente se objetivo G1 ocorre, sendo expresso por G1 $\rightarrow$ G2; assim uma dependência de contrato difere de uma dependência de precedência por um acionamento.

Neste trabalho, extendemos a dependência de contrato entre objetivos para dependência de contrato alternativa, dependência de contrato composicional, dependência condicional exclusiva e dependência condicional inclusiva entre objetivos G1, G2, G3, G4...Gn. Vejamos cada uma dessas dependências a seguir.

Na dependência de contrato alternativa, se G1 for realizado, então G2 v G3 v G4 v ... v Gn são também automaticamente realizados, sendo expresso por G1 $\rightarrow$ G2 v G3 v G4 v ... Gn. É importante observar que vários agentes podem executar um mesmo cenário em diferentes tempos, mas somente um agente pode executar um cenário em um tempo t, não existindo portanto a execução paralela de objetivos. Dessa forma, os objetivos G1, G2, G3,..., Gn podem ser acionados automaticamente, mas somente um será executado em um tempo t.

Na dependência de contrato composicional, se G1 for realizado, então G2 ^ G3^ G4 ^ ... ^ Gn são também automaticamente realizados, sendo expresso por G1 $\rightarrow$ G2^ G3^ G4^ ... ^ Gn.

Outra extensão proposta à dependência de contrato é dependência condicional exclusiva, onde objetivo G1 ocorre se pelo menos um dos objetivos G2 vG3 vG4 ocorrem, sendo representado por G2 v G3 v G4 ... Gn $\rightarrow$ G1. Nesse caso, quando algum dos objetivos condicionais ocorrem, os demais são bloqueados e G1 é realizado.

No caso da dependência de contrato condicional inclusiva, o objetivo G1 ocorre se todos os objetivos antecedentes ocorrem, sendo expresso por G2 $\wedge$ G3 $\wedge$ G4 ... $\wedge$ Gn $\rightarrow$ G1. Assim, G1 será realizado somente quando todos os objetivos condicionais ocorrem.

Com relação aos agentes, o método proposto permite a execução paralela de agentes, desde que sejam cenários diferentes. Outro aspecto que deve ser ressaltado é que um agente pode executar um mesmo cenário em diferentes tempos.

Objetivos podem também compartilhar dependências entre agentes. Assim, para um agente completar um objetivo, outro agente pode ter que primeiro completar outro objetivo (indicando uma pré-condição). Essa dependência se destina a facilitar a produção de um modelo integrado a ser construído durante a fase de integração da abordagem proposta.

No estudo de caso de bibliotecas, ao ser aplicada a estratégia de reorganização de objetivos e cenários, surgiram as seguintes relações de dependência por precedência , para o nível comportamental

Considerando G1.1 – Realizar empréstimo de uma maneira normal,
G1.2 – Realizar devolução de uma maneira normal e
G1.3 – Efetuar renovação de uma maneira normal

temos,

G1.1 < G1.2 (Ou seja, a devolução de um livro deve ser precedida de um empréstimo)
G1.1 < G1.3 (Para efetuar a renovação de um item da biblioteca, este deve ter sido emprestado primeiramente).

8 – Refinamento de objetivos e cenários

Após estabelecidas relações de dependências entre objetivos e, conseqüentemente entre cenários, inicia-se a fase de refinamento que utiliza uma estratégia de refinamento que consiste em um processo manual realizado pelo engenheiro de requisitos com o propósito de eliminar objetivos duplicados, unificar sinônimos, descobrir inconsistências, redundâncias e incompletezas internas nos cenários e objetivos. No caso

desse processo ser semi-automatizado, faz-se necessário primeiramente formalizar os objetivos e cenários para que estes possam então serem verificados**,** por meio de uma estratégia de verificação, utilizada na fase de verificação da abordagem proposta.

São eliminadas redundâncias e reconciliados objetivos sinônimos eliminando o objetivo cujo conteúdo é descrito por outro. É preferível que sejam identificados redundâncias e objetivos sinônimos, após os objetivos terem sido ordenados de acordo com as relações de precedência, uma vez que estes objetivos se encontram adjacentes um com o outro, em um conjunto ordenado refletindo suas relações de precedência comum compartilhadas.

Incentiva-se também a participação dos usuários e clientes neste processo de refinamento, de modo que algumas discrepâncias sejam trazidas para sua atenção permitindo-lhes indicar qual nome do objetivo é mais apropriado.

Antes de adentrar na fase de operacionalização de objetivos e cenários, faz-se necessário distinguir entre template e esquema, sendo o esquema uma forma de template extendida, onde novos campos são adicionados e os objetivos, caso de usos, cenários e ações são descritos com mais detalhes visando facilitar uma tradução dos esquemas no documento de requisitos de software.

9 – Operacionalização de objetivos e cenários

Na fase de operacionalização de objetivos e cenários, estes são descritos em mais detalhes e mapeados para seus respectivos esquemas. Dessa forma, possibilita a tradução da informação dos objetivos, casos de uso e cenários em um documento de requisitos de software por meio da tradução dos resultados do esforço das fases anteriores em um conjunto de esquemas.

Embora existam uma grande variedade de representações disponíveis, o método proposto emprega um estilo informal similar ao Método GBRAM. Durante essa fase, a estratégia utilizada consiste na estratégia dirigida a esquema, que se baseia na definição de esquemas-objetivos, esquemas-casos-de-uso e esquemas-cenários a seguir descritos.

O esquema-objetivo especifica os relacionamentos entre objetivos e cenários. Será definida a sintaxe para especificar esquemas de objetivos e seus atributos associados, bem como uma explicação de como objetivos podem ser expressos em uma série de esquemas

de objetivos que podem ser mapeados para um documento de requisitos de software e organizados de acordo com a topografia objetivo.

O modelo de objetivo e cenário incorpora toda a informação adquirida durante a análise de objetivo e elaboração de cenários. Ela é expressa em uma série de esquemas. Deve haver um esquema para cada objetivo e para cada cenário. A sintaxe do esquema para modelos objetivo utilizados para especificar objetivos consiste de um Número de RC, Nome do objetivo, Identificação do objetivo, Nível, Tipo, descrição, Agente/Ator, Código do cenário/Código do caso de uso, [pré-condições], [pós-condições], código do obstáculo e [subobjetivos]. Alguns desses elementos serão detalhados a seguir.

È importante notar que o nível pode ser funcional, comportamental ou físico (interno) e que o tipo de objetivo pode ser definido como serviço, funcional e interno. Obstáculos referem-se a condições ou objetivos que oferecem impedimentos à realização do objetivo, sendo mecanismos que auxiliam na descoberta de cenários excepcionais.

Cada caso de uso é expresso na forma de um esquema de caso de uso. Existe pelo menos um esquema caso de uso para cada objetivo; na maioria dos casos, vários esquemas casos de uso devem ser providos para cada objetivo. A semântica de cada cláusula em um esquema caso de uso é composta pelo número do RC, nível, tipo, nome do ator, código do caso de uso, descrição, nome do caso de uso e código do objetivo.

No caso do esquema para modelos de cenários e ações utilizados para especificar cenários e ações, a sintaxe do esquema consiste em: código do cenário, nível, tipo, descrição, ator/agente, [pré-condição], [pós-condição], ação, código do objetivo e [subobjetivos/episódios].

Percebe-se então que cada cenário é expresso na forma de um esquema de cenário. O nível do cenário pode ser funcional, comportamental ou interno e o tipo pode ser normal, variacional ou excepcional. Uma ação envolve tanto ações do usuário como ações do sistema. As cláusulas opcionais estão colocadas entre colchetes.

Cada ação de um cenário também é expressa na forma de um esquema ação, salientando-se que existe pelo menos um esquema ação para cada cenário. Na maioria dos casos, vários esquemas ações devem ser providos para cada cenário. A seguir descreve-se a

semântica de cada claúsula em um esquema ação: ação, tipo, entradas, modificações, número da seqüência, código do cenário.

Ressalta-se que o agente e as pré e pós-condições não foram colocados no esquema ação por serem os mesmos do esquema cenário. A notação utilizada pelos esquemas propostos não é uma notação formal, sendo muito similar a uma modelagem de dados. Assim objetivos, cenários, casos de uso e ações são especificados utilizando os modelos esquemas apresentados.

A seguir forma descritos alguns esquemas-objetivos, esquemas-casos de uso, esquemas-cenários e esquemas-ações para o exemplo de bibiliotecas.

- **Esquemas-objetivos**

  Nº do RC: RC1
  Código do objetivo: G1
  Nome do objetivo: Proceder circulação
  Nível: funcional
  Tipo: serviço
  Descrição: Consiste em gerenciar todos as operações de empréstimos, renovação e devolução de objetos solicitadas pelo usuário.
  Agente/Ator: bibiliotecário
  Cod-cenário/Cod-caso de uso: MUC1.1

  Nível funcional

  Nº do RC: RC1.1
  Cod-obj: G1.1
  Objetivo: Realizar empréstimo de uma maneira normal
  Nível: comportamental
  Tipo: funcional
  Descrição: Consiste no gerenciamento de empréstimos de livros solicitados pelos usuários
  Agente/Ator: bibliotecário
  Cod-cenário/Cod-caso de uso: C1.1

  Nível comportamental

  Nº do RC: RC1.1.1
  Cod-obj: G1.1.1
  Objetivo: Verificar validade do código de acesso
  Nível: interno
  Tipo: sistema
  Descrição: Consiste em verificar se o código de acesso é válido
  Agente/Ator: sistema
  Cod-cenário/Cod-caso de uso: C1.1.1

  Nível físico

- **Esquema-cenário**

  Cod-cenário: C1.1
  Nível: comportamental
  Tipo: normal

Descrição: Consiste em gerenciar os empréstimos realizados pelo usuário.
Código do cenário: C1.1

Agente: Bibiliotecário

Pré-condições: if código de acesso = válido

If Situação do usuário = OK

If status do livro = disponível

Ações

1. O usuário entra com código de acesso
2. O sistema valida o código de acesso
3. O usuário entra com código do usuário
4. O sistema verifica a situação do usuário
5. O usuário entra com código do livro
6. O sistema verifica se o livro pode ser emprestado
7. O sistema exibe mensagem "Empréstimo efetuado com sucesso" para o usuário.

Pós-condições: DO Status do livro = emprestado

- **Esquema-ação**

  Ação: O sistema valida código de acesso
  Tipo: sistema
  Reads: código
  Changes:
  Nº da sequência: 1
  Cod-cenário: C1.1

3.2.2 Modelo do produto

O modelo de produto combina os conceitos de serviços, pacotes de serviços, atores, casos de uso, objetivos e cenários em uma única estrutura, conforme ilustra a figura 3.3.

O modelo de produto do método proposto é centrado no conceito de um pedaço de requisito, i.e, um acoplamento de um objetivo a ser executado e um modelo de casos de uso explicando como o sistema irá interagir com os atores para executar o objetivo.

A definição de um cenário no modelo de produto proposto é baseada na noção de agente e ação, onde agente é “aquele que executa o cenário”. Um cenário tem como escopo suas pré e pós-condições, sendo classificado em normal, excepcional e variacional.

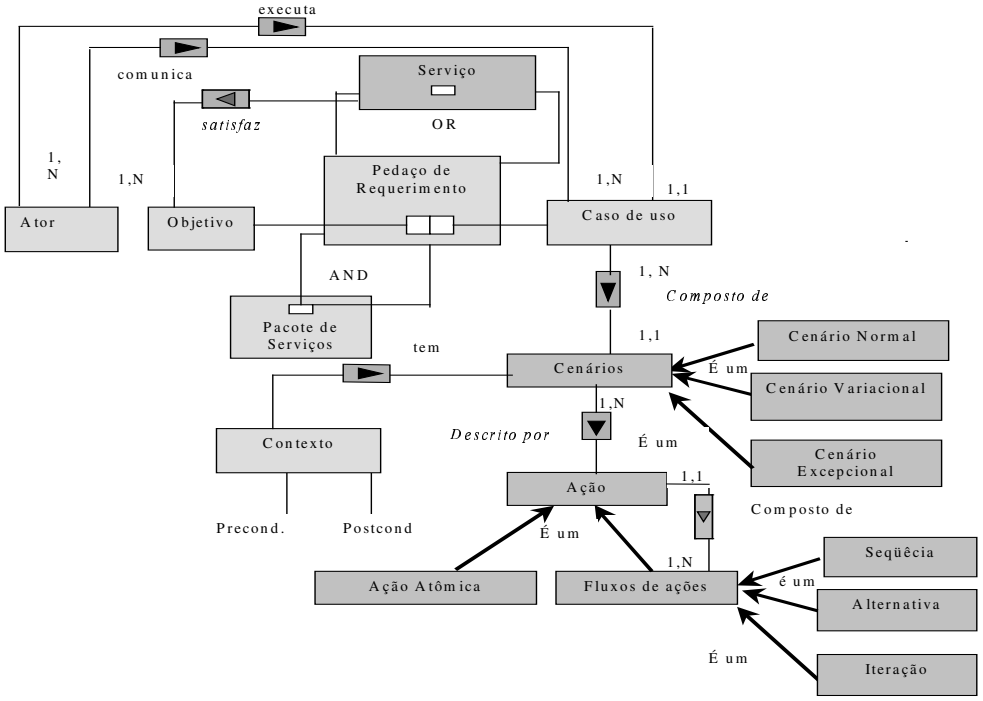

Figura. 3.3: Modelo do produto do Método Proposto.

O cenário normal consiste em um conjunto de ações executadas de maneira que o objetivo seja executado. Caso existam situações não normais, onde mesmo assim o objetivo é realizado estaremos diante de cenários variacionais. Os cenários excepcionais referem-se a um conjunto de ações onde o objetivo não é realizado.

Como visto na figura 3.3, uma ação pode ser uma ação atômica ou um fluxo de ações, sendo que estas constituem um fluxo alternativo, seqüencial ou iterativo. A outra característica do modelo de produto proposto que merece ser considerada referem-se aos novos conceitos de “serviço” e “pacote de serviços”. Estes conceitos são criados a partir da junção de um ou mais casos de uso originando dessa forma o conceito de serviço, que satisfaz somente um objetivo. Já o conceito de pacote de serviços origina-se da junção de vários modelos de casos de uso, por meio de relacionamentos AND.

Observa-se também na figura 3.3 que um caso de uso pode ser executado por um e somente um ator, conforme é descrito em [42], tornando assim o conceito de caso de uso

mais simples. Por outro lado, um ator executa um ou vários casos de uso e comunica-se com vários casos de uso. Da mesma forma, um caso de uso pode se comunicar com vários atores, mas pode ser executado por apenas um.

3.3 Sumário

A abordagem apresentada neste capítulo é proveniente da integração da abordagem CREWS-L´Ecritoire [45] com o trabalho de Regnell et al [42,43] e o Método GBRAM [4,5]. Na Tabela 3.1 podem ser vistos os elementos de outros métodos utilizados na abordagem proposta.

Abordagem CREWS – L"Ecritoire	**Método GBRAM**	**Método UORE**	**Método Objectory**	**Modelo de Caso de uso hierárquico**
- Noção de RC como núcleo da abordagem e resultado de um acoplamento bidirecional de dois conceitos. - Relacionamento AND, OR e de refinamento entre RCs	- Relações de dependência entre objetivos	- Descrição de casos de uso, onde não são permitidos múltiplos atores, existindo uma visão única para cada ator envolvido no caso de uso	- Notação gráfica	-Utilização de alguns conceitos do modelo conceitual. - Idéia de organização do modelo em níveis de hierarquia, que se diferenciam pelo detalhamento cada vez maior dos conceitos

Tabela 3.1 – Elementos de outros métodos utilizados na abordagem proposta.

O método criado para elicitação e modelagem de requisitos acopla objetivos e casos de uso em um pedaço de requisito (RC), relacionados através de relacionamentos AND, OR e de refinamento, organizados hierarquicamente em três níveis de abstração: o funcional, o comportamental e o físico.

O método proposto consiste em um modelo de processo e um modelo de produto. O modelo de processo engloba as fases: Elicitar objetivos de serviço, especificar

casos de uso, elicitar objetivos funcionais, especificar cenários estruturais, elicitar objetivos do sistema, especificar cenários físicos, reorganizar objetivos e cenários, refinar objetivos e cenários, operacionalizar objetivos e cenários e, gerar o documento de requisitos, produto principal do método proposto.

O modelo de produto gerado (vide figura 3.3) acopla os vários conceitos obtidos com a aplicação do modelo de processo. Assim um objetivo é acoplado a um modelo de caso de uso, que relacionados originam os conceitos de serviços e pacote de serviços. Um caso de uso é descrito por vários cenários, compostos por ações. Os cenários podem ser normais, variacionais ou excepcionais e ações consistem em uma ação atômica ou um fluxo de ações seqüenciais, alternativas ou iterativas. Os cenários normais e variacionais conduzem à realização do objetivo com sucesso, enquanto no cenário excepcional, o objetivo falha.

Na próxima seção será realizado um estudo comparativo do método proposto com algumas abordagens explicadas no Cap II.

3.4 Estudo comparativo com outros métodos existentes

A Abordagem CREWS- L`Ecritoire tem como um dos seus principais aspectos a utilização de um acoplamento bidirecional cenário-objetivo, onde à medida em que é descoberto um objetivo, este é operacionalizado por um cenário. Na direção reversa, i.e., de cenários para objetivos, a abordagem guia o processo de elicitação de requisitos descobrindo novos objetivos através da análise de cenários textuais.

Na abordagem proposta observa-se um relacionamento bidirecional entre objetivo e caso de uso, onde ocorre a descoberta de casos de uso a partir de objetivos de serviço, assim como a geração de novos objetivos a partir dos casos de uso, sendo operacionalizados através de cenários estruturais cujas ações do sistema geram novos objetivos concretizados pelos cenários internos do sistema.

Ambas as abordagens adotam o RC como construtor básico para organização de cenários, sendo que na abordagem CREWS-L`Ecritoire são definidos os níveis contextual, interação do sistema e interno do sistema. Na abordagem proposta é adotado o conceito de caso de uso de acordo com [42] que o define como uma maneira de utilizar o sistema descrito a partir da perspectiva de um único ator organizados nos níveis funcional, comportamental e físico.

Outra diferença que pode ser encontrada concerne na geração de novos objetivos do nível físico, onde eles se originam das ações do sistema pertencentes às descrições dos cenários estruturais no nível comportamental. Na abordagem CREWS-L`Ecritoire, a cada interação ocorrida no nível de interação do sistema origina-se um objetivo do sistema.

O Método GBRAM (Método de Análise de Requisitos baseados em objetivos) envolve duas fases até a produção do documento de software (DRS): a fase de análise dos objetivos e a fase de refinamento dos objetivos. Na fase de análise dos objetivos, são identificados os objetivos, os *stakeholders* e os agentes responsáveis pela execução do objetivo. Objetivos são classificados em objetivos de manutenção e objetivos de realização. São estabelecidas relações de dependência entre os objetivos e construída a topografia de objetivos.

A abordagem proposta engloba as fases de elicitação de objetivo, especificação de casos de uso, elaboração de cenários, reorganização de objetivos e cenários, refinamento de objetivos e cenários e operacionalização de objetivos e cenários. Durante a aplicação do método proposto são identificados os objetivos, os casos de uso, os cenários dentro dos três níveis estabelecidos: funcional, comportamental e físico. Dessa forma, os objetivos são classificados em objetivos de serviço, objetivos funcionais e objetivos do sistema. São mantidas as relações de dependência do Método GBRAM e são criadas novas relações de dependência entre os objetivos, através da extensão da dependência de contrato para as variantes dependência de contrato alternativa, composicional, dependência condicional exclusiva e inclusiva.

Ambas as abordagens adotam as dependências de precedência e de contrato, sendo diferenciadas por um acionamento. As extensões criadas para as dependências de contrato são uma das partes diferenciais em nossa abordagem. Na dependência de contrato alternativa, a realização do objetivo antecedente depende da realização de apenas um ou mais objetivos consequentes.

Na dependência de contrato composicional, a realização do objetivo antecedente depende da realização de todos os objetivos conseqüentes. No caso da dependência de contrato condicional exclusiva, o objetivo consequente depende da realização

de um ou mais objetivos antecedentes. Entretanto, na dependência de contrato condicional inclusiva, o objetivo consequente depende da realização de todos os objetivos antecedentes.

Ambas as abordagens (GBRAM e a proposta) possuem uma fase de refinamento de objetivos onde são eliminados os objetivos redundantes e unificados os objetivos sinônimos.

Durante a segunda fase do Método GBRAM que corresponde à fase de refinamento do objetivo, são identificados os obstáculos como meios possíveis de falhas dos objetivos, enquanto os cenários oferecem uma maneira natural para descrever circunstâncias excepcionais, permitindo a consideração de alternativas possíveis de operacionalizações de objetivos.

No método proposto, adotou-se o conceito de obstáculos da mesma forma como foi adotado pelo Método GBRAM, ou seja, como um mecanismo de auxílio na descoberta de cenários excepcionais. No entanto, dentro da nossa abordagem, os cenários descrevem tanto situações excepcionais, como normais ou variacionais. No Método GBRAM, são considerados apenas os cenários excepcionais; outra diferença que pode ser visualizada refere-se à atividade de operacionalização onde objetivos podem ser expressos em uma série de esquemas objetivos mapeados para o documento de requisitos do software e organizados de acordo com a topografia de objetivos. No caso específico da nossa abordagem, existem esquemas-objetivo, esquemas-cenários, esquemas-casos de uso e esquemas-ações, dependendo do nível de detalhamento.

4. ESTUDO DE CASO

No estudo de caso considerado, tem-se uma rede de restaurantes interligados, que funcionam similarmente, onde o nome, os funcionários e as características dos restaurantes, como tipo de comida, ambiente, foram mantidas, sendo efetuadas alterações em pontos considerados estratégicos.

Dentro desta política o restaurante continuou a crescer, até que num dado momento seus dirigentes perceberam que os seus lucros poderiam ser maiores se o departamento de compras fosse centralizado. Os restaurantes conseguiriam as matérias primas a um custo menor e de melhor qualidade, conseqüentemente diminuindo os preços do produto final e aumentando a satisfação de seus clientes.

Para tanto foi escolhido um restaurante central (matriz) cuja finalidade é receber o cardápio semanal de cada restaurante, verificar se há uma duplicidade de pratos no mesmo dia para restaurantes próximos e sugerir a mudança para os mesmos. Cadastrar os fornecedores, cotar os produtos, solicitar itens alternativos quando da falta dos mesmos no mercado, receber as mercadorias e manter um centro de controle de qualidade, são algumas das funções deste restaurante.

O ciclo de atuação do restaurante matriz é semanal (de segunda à sexta-feira). Ele recebe de todos os restaurantes o cardápio da semana seguinte, quando é gerada uma lista única concatenada da matéria-prima necessária para toda a rede. Uma lista separada por tipo de fornecedor é entregue aos compradores na segunda-feira pela manhã. Os compradores efetuam a cotação, cadastram os fornecedores e marcam os itens não disponíveis no mercado.

Na terça-feira, uma lista contendo os produtos hortifrutigranjeiros da época com seus respectivos custos são distribuídos aos restaurantes da rede. Quanto aos produtos não disponíveis, os restaurantes retornam com as opções alternativas.

Na quarta-feira, os pedidos de compra são enviados aos fornecedores com a devida programação de entrega. Os fornecedores de horti-fruti devem efetuar a entrega diariamente até às 08:00 e a sua retirada deve ser feita das 8:30 às 12:30 hs. Os produtos não perecíveis devem ser entregues nos períodos da tarde às segundas, quartas e sextas-feiras e a sua distribuição às terças e quintas-feiras. No ato do recebimento a matriz se encarrega de checar a qualidade de todos os produtos, assim como a pontualidade dos fornecedores.

O preço é calculado sobre o próprio custo, + 10% de taxa de administração + frete. Os pagamentos são efetuados por meio de duplicatas automaticamente pagas em banco, de tal forma que dispense uma vasta equipe de contas a pagar e receber.

De acordo com a descrição acima, foi escolhido um domínio de restaurantes para aplicação do método proposto. Inicialmente, foram aplicadas as estratégias "estratégia-perguntas específicas" e "estratégia-identificação-objetivos-iniciais" para obtenção dos objetivos de serviço.

Por meio da aplicação dessas estratégias, foi realizada uma pequena descrição do funcionamento dos procedimentos executados sobre o domínio de restaurantes, sintetizando assim toda a informação adquirida pelo engenheiro de requisitos.

Dessa forma, foram encontrados para o exemplo de restaurantes os seguintes objetivos de serviço: G1: Controlar Cardápio; G2: Efetuar cotação; G3: Comprar produtos, G4: Distribuir alimentos e G5: Controlar qualidade, que acoplados aos seus respectivos modelos de casos de uso formam os RCs funcionais RC1, RC2, RC3, RC4 e RC5 pertencentes ao nível funcional, conforme mostra a figura. 4.1:

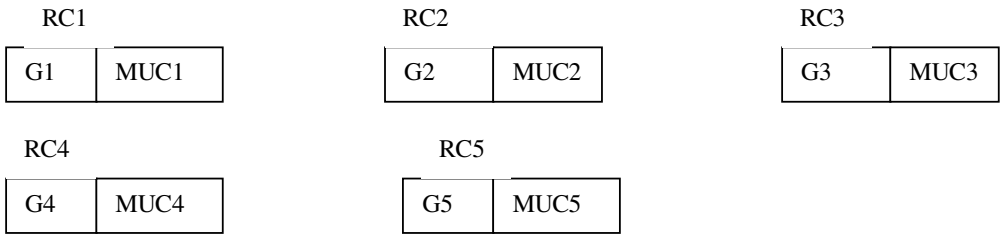

Figura. 4.1. Nível funcional para o Sistema de Restaurantes**.**

Para o RC funcional RC1, foi elaborado o modelo de caso de uso correspondente utilizando-se a estratégia gráfica e a estratégia de descoberta baseada em ator. Cada caso de uso do nível funcional dá origem a um objetivo funcional no nível comportamental. Dessa forma, o caso de uso "Receber cardápio" origina no nível comportamental o G1.1 por meio da estratégia de refinamento "Receber cardápio de uma maneira normal", que é operacionalizado pelo cenário C1.1.

O caso de uso "Verificar duplicidade" origina no nível comportamental o G1.2 ("Verificar duplicidade de uma maneira normal"), através da aplicação da estratégia de refinamento, sendo operacionalizado pelo cenário C1.2.

Ainda dentro do RC1, o caso de uso "Sugerir alternativa" origina no nível comportamental o G1.3: ("sugerir alternativa de uma maneira normal"), através da aplicação da estratégia de refinamento, sendo que este objetivo G1.3 é operacionalizado pelo cenário C1.3.

Para o objetivo de serviço G2: Efetuar cotação, obtido através da aplicação das estratégias de perguntas específicas e identificação objetivos iniciais, existem três casos de uso acoplados: Cadastrar fornecedores, cadastrar produtos e Cotar produtos.

Assim, o caso de uso “Cadastrar fornecedores” origina o objetivo funcional G2.1 "Cadastrar fornecedores de uma maneira normal" pela estratégia de refinamento; o caso de uso "Cadastrar produtos" dá origem ao objetivo funcional G2.2 "Cadastrar produtos de uma maneira normal" e o caso de uso “Cotar produtos” dá origem ao objetivo funcional G2.3 "Cotar produtos de uma maneira normal" operacionalizado pelo cenário C2.3.

Por conseguinte o RC3 formado pelo objetivo de serviço "Comprar produtos" e modelo de caso de uso correspondente, obtido a partir da estratégia gráfica/template e também por meio da estratégia-descoberta-baseada-em-ator é melhor especificado no nível comportamental pelo RC3.1 e RC3.2 compostos pelo objetivo funcional G3.1 “Verificar cotação de uma maneira normal” e pelo objetivo funcional G3.2 "Fazer pedido de compra de uma maneira normal", operacionalizados respectivamente pelos cenários C3.1 e C3.2.

Ao longo do processo foi aplicada a estratégia alternativa para gerar objetivos alternativos. Assim, no nível funcional para o RC1.1 foi gerado o RC1.1 ^{1} , para o RC1.2 o RC 1.2^{1} e para o RC 1.3 foi gerado o RC1.3^{1} .

Ainda no nível comportamental, a aplicação da estratégia AND sobre o G1.1 permite identificar os objetivos complementares G1.2 e G1.3. No nível físico, tem-se o RC1.1.1, RC1.2.1 e RC 1.3.1 obtidos a partir da aplicação da estratégia de refinamento sobre os RC1.1, RC1.2 e RC1.3 respectivamente dentro do nível comportamental.

Para o RC1.1.1, são obtidos os RC1.1.2, RC 1.1.3 a partir da aplicação da estratégia AND sobre o mesmo, gerando objetivos complementares para o primeiro.

Para o RC 1.2.1, por meio da estratégia AND são obtidos os RC 1.2.2, RC 1.2.3 e RC 1.2.4; e para o RC 1.3.1, através da aplicação da mesma estratégia origina os RC 1.3.2, RC 1.3.3 e RC 1.3.4.

Para a elaboração do modelo de caso de uso, foi utilizada a notação de Jacobson onde são representados os atores externos (que interagem com o sistema) e os casos de uso, compostos de um conjunto de interações entre os atores e o sistema. Também poderia ter sido utilizada uma template composta pelo código do caso de uso, nome do caso de uso, [identificação do ator] e código do objetivo.

Para a especificação dos cenários foi utilizada uma template com os seguintes elementos: código do cenário, agente, [pré-condições], [pós-condições], ações e código do objetivo.

RC1:
G1: Controlar Cardápio
MUC1:

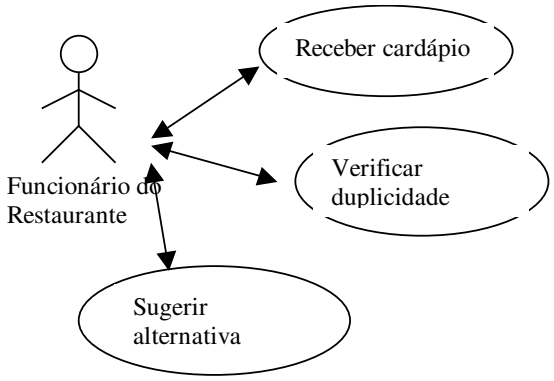

Figura. 4.2.a: Nível funcional RC1

RC2
G2: Efetuar cotação
MUC2:

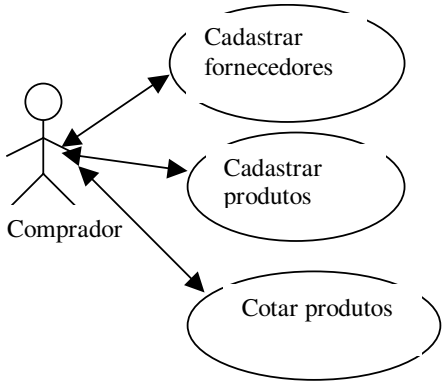

Figura. 4.2.b: Nível funcional RC2

RC3
G3: Comprar produtos
MUC3:

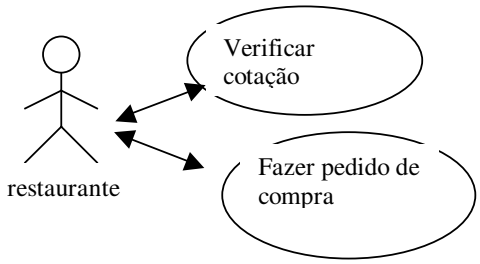

Figura. 4.2.c: Nível funcional RC3

RC4
G4: Distribuir alimentos
MUC4:

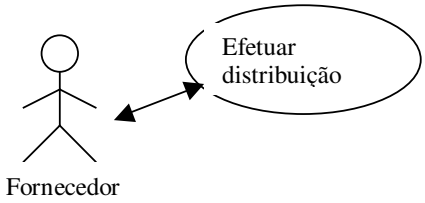

Figura 4.2.d: Nível funcional RC4

RC5
G5: Controlar qualidade
MUC5:

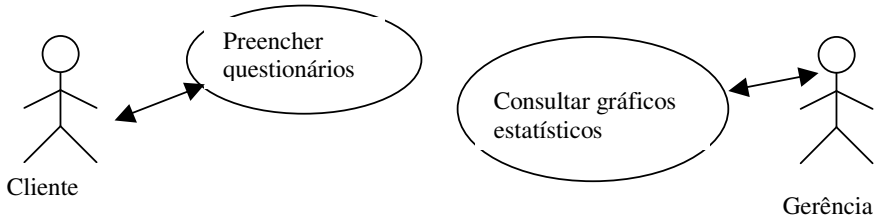

Figura. 4.2.e: Nível funcional RC5.

Na fig. 4.3 pode ser vislumbrada a hierarquia de RCs para o primeiro RC do exemplo dos restaurantes.

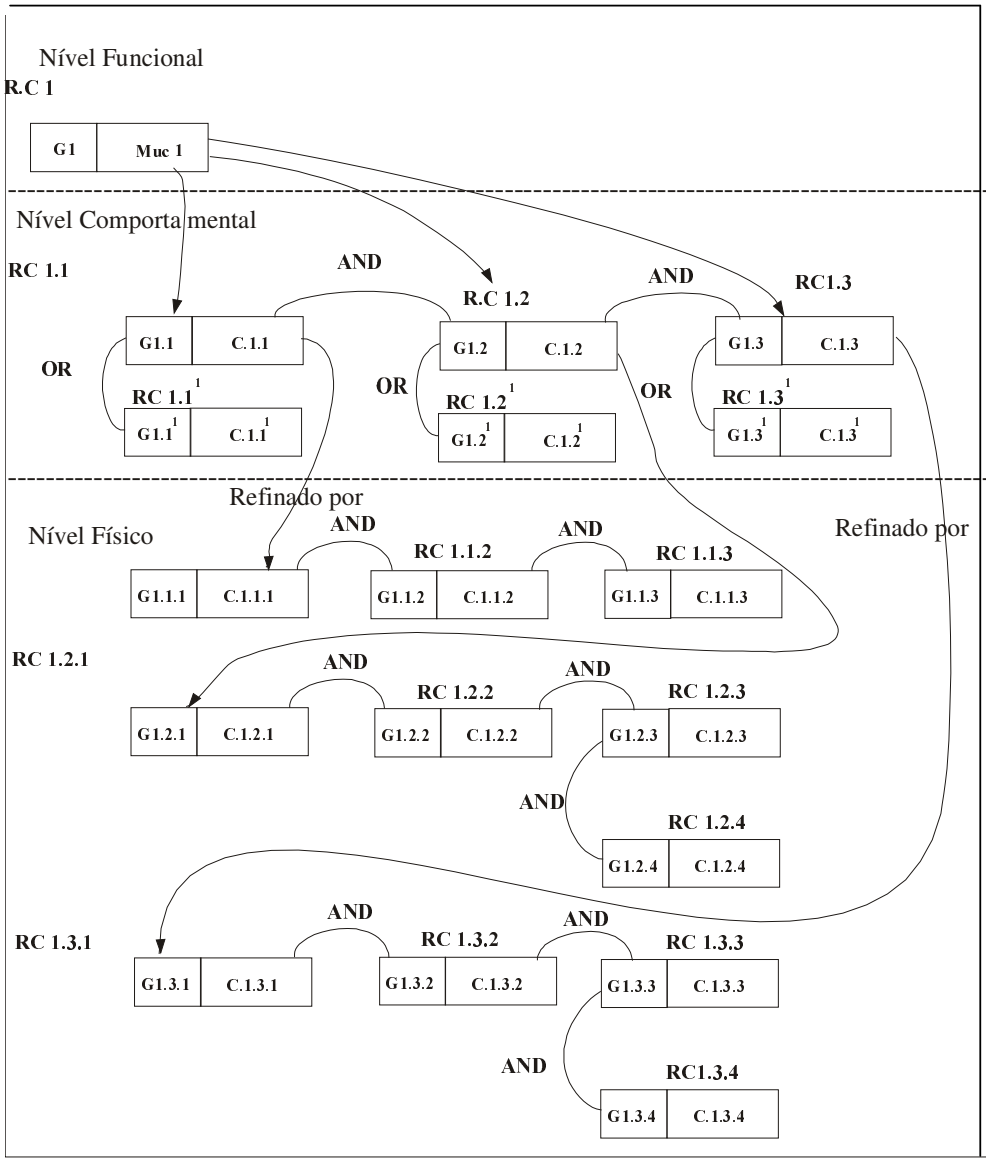

Figura 4.3: Hierarquia de RCs para o RC1.

RC1.1 - Nível comportamental
G1.1: Receber cardápio de uma maneira normal
Agente/Ator: restaurante
Pré-condições: if código de acesso do usuário = válido
If Senha do usuário = válida
If Código do cardápio = válido
C1.1:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.

3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio
6. O sistema valida código do cardápio.
7. O usuário confirma recebimento
8. O sistema exibe mensagem “Cardápio recebido com sucesso" para o usuário.
Pós-condições: DO Status do cardápio = "recebido"

RC1.1¹
G1.1¹: Receber cardápio com código do cardápio inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário válido
If Código do cardápio inválido
C1.1¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio
6. O sistema verifica código do cardápio.
7. Enquanto código do cardápio inválido
8. O usuário entra com novo código do cardápio.
9. O sistema verifica código do cardápio
10. Fim-enquanto.
11. O sistema valida código do cardápio.
12. O usuário confirma recebimento
13. O sistema exibe mensagem “Cardápio recebido com sucesso" para o usuário.
Pós-condições: DO Status do cardápio = "recebido"

RC1.1^2
G1.1^2: Receber cardápio com código de acesso inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário inválido
C1.1^2:
1. O usuário entra com código de acesso
2. O sistema verifica código de acesso
3. O sistema exibe mensagem “Código de acesso inválido” para o usuário
4. if nº entradas de códigos de acesso = 3
5. O sistema trava entrada do usuário na rede
6. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO Status do cardápio = " "

RC1.1^3
G1.1^3: Receber cardápio com senha inválida.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida

C1.1[3]:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO Status do cardápio = " "

RC1.2 - Nível comportamental
G1.2: Verificar duplicidade de uma maneira normal
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário válido
If Código do cardápio válido
C1.2:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio.
6. O sistema valida código do cardápio.
7. O sistema verifica duplicidade de cardápio.
8. O sistema exibe mensagem “Cardápio duplicado" para o usuário.
Pós-condição: DO duplicidade do cardápio verificada

RC1.2[1] - Nível comportamental
G1.2[1]: Verificar duplicidade com código do cardápio inválido.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário = válido
If Senha do usuário válido
If Código do cardápio inválido
C1.2:[1]
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio inválido.
6. O sistema verifica código do cardápio.
7. Enquanto código do cardápio inválido
8. O usuário entra com novo código do cardápio.
9. O sistema verifica código do cardápio.
10. O sistema verifica duplicidade do cardápio
11.Fim-enquanto.
12. O sistema exibe mensagem “cardápio duplicado" para o usuário.
Pós-condição: DO código do cardápio verificado.

RC1.2^2
G1.2^2: Verificar duplicidade com código de acesso inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário inválido
C1.2^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO código do cardápio não verificado

RC1.2^3
G1.2^3: Verificar duplicidade com senha inválida.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C1.2^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO código do cardápio não verificado

RC1.3 - Nível comportamental
G1.3: Sugerir alternativa de uma maneira normal
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Código do cardápio duplicado
C1.3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio.
7. O sistema verifica duplicidade de cardápio.
8. O sistema mostra lista de cardápios alternativos.
Pós-condições: DO opções de cardápios visualizadas

RC1.3¹ - Nível comportamental
G1.3¹: Sugerir alternativa com código do cardápio inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Código do cardápio duplicado

C1.3¹:
1. O usuário entra com código de acesso.
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha.
5. O usuário entra com código do cardápio.
6. O sistema verifica código do cardápio.
7. O sistema exibe mensagem “código do cardápio inválido" para o usuário.
8. O usuário entra com novo código do cardápio.
9. O sistema verifica duplicidade do cardápio.
10. O sistema mostra lista de cardápios alternativos.
Pós-condição: DO opções de cardápios visualizadas

RC1.3^2
G1.3^2: Sugerir alternativa com código de acesso inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário inválido
C1.3^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO opções de cardápios não visualizadas

RC1.3^3
G1.3^3: Sugerir alternativa com senha inválida.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C1.3^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO Status do cardápio = " "

RC2.1
G2.1: Cadastrar fornecedores de uma maneira normal
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Dados do fornecedor válidos
C2.1:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.

3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados do fornecedor
6. O sistema verifica validade das informações
7. O sistema exibe mensagem “Inclusão realizada com sucesso" para o usuário
Pós-condição: DO fornecedor cadastrado

RC2.1¹
G2.1¹: Cadastrar fornecedores com dados incompletos
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Dados do fornecedor incompletos
C2.1¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados do fornecedor
6. O sistema verifica validade das informações
7. O sistema exibe mensagem “Dados incompletos. Preencha os campos em branco” para o usuário.
8. O usuário completa os dados do fornecedor
9. O sistema exibe mensagem “Inclusão realizada com sucesso" para o usuário
Pós-condição: DO Fornecedor cadastrado

RC2.1^2
G2.1^2: Cadastrar fornecedores com código de acesso inválido
Agente/Ator: comprador
Pré-condições: If código de acesso do usuário inválido
C2.1^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO fornecedor não cadastrado.

RC2.1^3
G2.1^3: Cadastrar fornecedores com senha inválida.
Agente/Ator: Comprador
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C2.1^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede

7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO fornecedor não cadastrado.

RC2.2
G2.2: Cadastrar produtos de uma maneira normal
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Dados do produto válidos

C2.2:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados do produto
6. O sistema verifica validade das informações
7. O sistema exibe mensagem “Inclusão realizada com sucesso" para o usuário
Pós-condição: DO Produto cadastrado

RC2.2¹
G2.2¹: Cadastrar produtos com dados incompletos
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Dados do produto incompletos

C2.2¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados do produto
6. O sistema verifica validade das informações
7. O sistema exibe mensagem “Dados incompletos. Preencha os campos em branco" para o usuário.
8. O usuário completa os dados do fornecedor
9. O sistema exibe mensagem "Inclusão realizada com sucesso" para o usuário.
Pós-condição: DO dados do produto cadastrado

RC2.2^2
G2.2^2: Cadastrar produtos com código de acesso inválido
Agente/Ator: comprador
Pré-condições: If código de acesso do usuário inválido
C2.2^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.

Pós-condições: DO dados do produto não cadastrado.

RC2.2[3]
G2.2[3]: Cadastrar produtos com senha inválida.
Agente/Ator: comprador
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C2.2[3]:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO dados do produto não cadastrado.

RC2.3
G2.3 Cotar produtos de uma maneira normal
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Código do produto válido
C2.3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do produto
6. O sistema valida código do produto
7. Enquanto nome do fornecedor <> Ø
8. O usuário entra com nome do fornecedor
9. O usuário entra com respectivo preço.
10. Fim-enquanto
11. O sistema exibe mensagem "Produto cotado com sucesso" para o usuário.
Pós-condição: produto cotado

RC2.3¹
G2.3¹ Cotar produtos com código do produto inválido
Agente/Ator: comprador
Pré-condições: If código de acesso válido
If Senha válida
If Código do produto inválido
C2.3¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do produto

6. O sistema verifica código do produto
7. Enquanto código do produto inválido
    8. O usuário entra com novo código do produto.
    9. O sistema verifica código do produto
10. Fim-enquanto.
11. O sistema valida código do produto
12. Enquanto nome do fornecedor <> b
    13. O usuário entra com nome do fornecedor
    14.O usuário entra com respectivo preço
15. Fim-enquanto
16. O sistema exibe mensagem "Produto cotado com sucesso" para o usuário.
Pós-condição: DO produto cotado

RC2.3^2
G2.3^2: Cotar produtos com código de acesso inválido
Agente/Ator: comprador
Pré-condições: If código de acesso do usuário inválido
C2.3^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO produto não cotado.

RC2.3^3
G2.3^3: Cotar produtos com senha invalida.
Agente/Ator: Comprador
Pré-condições: If código de acesso do usuário válido
    If Senha do usuário inválida
C2.3^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO produto não cotado.

RC3.1
G3.1: Verificar cotação de uma maneira normal
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
    If Senha válida
    If Código do produto válido
C3.1:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.

3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do produto
6. O sistema verifica validade do código do produto
7. O sistema mostra fornecedores e respectivos preços
Pós-condição: DO cotação verificada.

RC3.1¹
G3.1¹: Verificar cotação com o código do produto inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Código do produto inválido
C3.1¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do produto
6. O sistema verifica validade do código do produto
7. Enquanto código do produto inválido
8. O usuário entra com novo código do produto
9. O sistema verifica validade do código do produto
10.Fim-enquanto
11. O sistema mostra fornecedores e respectivos preços para o usuário.
Pós-condição: DO cotação verificada

RC3.1^2
G3.1^2: Verificar cotação com código de acesso inválido
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário inválido
C3.1^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO cotação não verificada.

RC3.1^3
G3.1^3: Verificar cotação com senha inválida.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C3.1^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário

5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO cotação não verificada.

RC3.2
G3.2: Fazer pedido de compra de uma maneira normal
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Dados da compra válidos
C3.2:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados da compra
6. O sistema exibe mensagem "Inclusão realizada com sucesso" para o usuário
7. O usuário envia pedido de compra para fornecedor escolhido
8. O sistema exibe mensagem “Pedido de compra efetuado com sucesso" para o usuário.
Pós-condição: DO status do pedido de compra = "encaminhado"

RC3.2¹
G3.2¹: Fazer pedido de compra com fornecedor inexistente
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Dados da compra válidos
If Código do fornecedor inexistente
C3.2¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados da compra
6. O sistema exibe mensagem "Inclusão realizada com sucesso" para o usuário
7. O usuário envia pedido de compra para fornecedor escolhido.
8. Enquanto fornecedor inexistente
9. O usuário seleciona outro fornecedor.
10. Fim-enquanto.
11. O usuário envia pedido de compra para um novo fornecedor.
12. O sistema exibe mensagem "Pedido de compra efetuado com sucesso" para o usuário.
Pós-condição: DO status do pedido de compra = "encaminhado"

RC3.2²
G3.2²: Fazer pedido de compra com dados da compra incompleto.
Agente/Ator: restaurante
Pré-condições: If código de acesso válido
If Senha válida
If Dados da compra incompletos

C3.2²:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados da compra
6. O sistema exibe mensagem "Dados da compra incompletos" para o usuário
7. O usuário completa dados da compra em aberto.
8. O sistema exibe mensagem "Inclusão realizada com sucesso" para o usuário.
9. O usuário envia pedido de compra para fornecedor escolhido.
10. O sistema exibe mensagem “Pedido de compra efetuado com sucesso" para o usuário.
Pós-condição: DO status do pedido de compra = "encaminhado"

RC3.2^3
G3.2^3: Fazer pedido de compra com código de acesso inválido
Agente/Ator: comprador
Pré-condições: If código de acesso do usuário inválido
C3.2^3:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário.
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO status do pedido de compra = “ “.

RC3.2^4
G3.2^4: Fazer pedido de compra com senha inválida.
Agente/Ator: restaurante
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C3.2^4:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO status do pedido de compra = “ “.

RC4.1
G4.1: Efetuar distribuição de uma maneira normal
Agente/Ator: Fornecedor
Pré-condições: If código de acesso válido
If Senha válida
If Nº da nota fiscal válida
C4.1:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.

3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com nº da nota fiscal
6. O sistema verifica se nº da nota fiscal é válida
7. O sistema encaminha nº da nota fiscal para restaurante destino
Pós-condição: DO Status da nota fiscal = "encaminhado"

RC4.1¹:
G4.1¹: Efetuar distribuição com nº de nota fiscal inválido
Agente/Ator: Fornecedor
C4.1¹:
Pré-condições: If código de acesso válido
If Senha válida
If Nº da nota fiscal inválida
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com nº da nota fiscal
6. O sistema verifica se nº da nota fiscal é valido
7. Enquanto nº da nota fiscal for inválido
8. O usuário entra com novo Nº de nota fiscal.
9. O sistema verifica se Nº da nota fiscal é válido.
10. Fim-enquanto.
11. O sistema valida nº da nota fiscal
12. O sistema encaminha Nº de nota fiscal para restaurante destino.
Pós-condição: DO status da nota fiscal = "encaminhado"

RC4.1^2
G4.1^2: Efetuar distribuição com código de acesso inválido
Agente/Ator: Fornecedor
Pré-condições: If código de acesso do usuário inválido
C4.1^2:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO status da nota fiscal = “ “.

RC4.1^3
G4.1^3:Efetuar distribuição com senha inválida.
Agente/Ator: Fornecedor
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C4.1^3:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário

5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO status da nota fiscal = “ “.

RC5.1 - Nível comportamental
G5.1: Preencher questionários de uma maneira normal
Agente/Ator: Cliente
Pré-condições: If código de acesso válido
If Senha válida
If Dados pessoais válidos
If Dados do questionário OK
C5.1:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados pessoais
6. O sistema verifica dados pessoais
7. O usuário preenche dados solicitados.
8. O sistema verifica dados preenchidos corretamente
9. O sistema exibe mensagem “Preenchimento OK" para o usuário.
Pós-condição: DO clientes_avaliados = clientes_avaliados + 1

RC5.1¹ - Nível comportamental
G5.1¹: Preencher questionários com dados pessoais incompletos
Agente/Ator: Cliente
Pré-condições: If código de acesso válido
If Senha válida
If Dados pessoais incompletos
C5.1¹:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados pessoais
6. O sistema verifica dados pessoais
7. O sistema exibe mensagem "Falta preencher campo obrigatório"
8. O usuário preenche campos em branco.
9. O sistema verifica se todos os dados pessoais estão preenchidos.
10.O usuário preenche dados solicitados do questionário.
11. O sistema verifica se dados estão preenchidos corretamente
12. O sistema exibe mensagem “Preenchimento OK" para o usuário.
Pós-condição: DO clientes_avaliados= clientes_avaliados +1

RC5.1² - Nível comportamental
G5.1²: Preencher questionários com dados em branco
Agente/Ator: Cliente
Pré-condições: If código de acesso válido
If Senha válida

If Dados do questionário incompletos

C5.1²:

1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com dados pessoais
6. O sistema verifica dados
7. O usuário preenche dados solicitados do questionário.
8. O sistema verifica se dados estão preenchidos corretamente
9. O sistema exibe mensagem "Respostas incompletas. Preencha os campos vazios".
10. O usuário entra com dados para completar o questionário de avaliação do atendimento.
11. O sistema verifica se dados estão preenchidos corretamente
12. O sistema exibe mensagem “Preenchimento OK" para o usuário.

Pós-condição: DO Questionário OK

DO Clientes_avaliados = clientes_avaliados + 1

RC5.1[3]

G5.1[3]: Preencher questionários com código de acesso inválido

Agente/Ator: Cliente

Pré-condições: If código de acesso do usuário inválido

C5.1[3]:

1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem “Procure administrador da rede" para o usuário.

Pós-condições: DO questionário não OK

RC5.1[4]

G5.1[4]: Preencher questionários com senha invalida.

Agente/Ator: Cliente

Pré-condições: If código de acesso do usuário válido

If Senha do usuário inválida

C5.1[4]:

1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.

Pós-condições: DO questionário não OK.

Nível comportamental - RC 5.2

G5.2: Consultar gráficos estatísticos de uma maneira normal

Agente/Ator: Gerente

Pré-condições: If código de acesso válido

If Senha válida

If Parâmetros de pesquisa válidos

C5.2:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com parâmetros de pesquisa.
6. O sistema verifica se parâmetros são válidos.
7. O sistema mostra gráficos estatísticos para o usuário.
Pós-condição: DO Gráficos estatísticos visualizados

RC 5.2 ¹
G5.2¹: Consultar gráficos estatísticos com correção dos parâmetros de pesquisa.
Agente/Ator: Gerente
Pré-condições: If código de acesso válido
If Senha válida
If Parâmetros de pesquisa inválidos
C5.2¹
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com parâmetros de pesquisa.
6. O sistema verifica se parâmetros são válidos.
7. Enquanto parâmetros inválidos para pesquisa
8. O usuário entra com novos parâmetros de pesquisa.
9. O sistema verifica se parâmetros são válidos.
10. Fim-enquanto.
11. O sistema mostra dados solicitados.
Pós-condição: DO dados solicitados visualizados

RC 5.2 ²
G5.2²: Consultar gráficos estatísticos com parâmetros de pesquisa insuficientes
Agente/Ator: Gerente
Pré-condições: If código de acesso válido
If Senha válida
If Parâmetros de pesquisa insuficientes
C5.2²:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com parâmetros de pesquisa.
6. O sistema verifica se parâmetros são válidos.
7. O sistema exibe mensagem "Parâmetros insuficientes para pesquisa"
8. O usuário completa os parâmetros necessários para a pesquisa.
9. O sistema verifica se parâmetros são válidos.
10. O sistema mostra dados consultados.
Pós-condição: DO dados solicitados visualizados

RC5.2[3]
G5.2[3]: Consultar gráficos estatísticos com código de acesso inválido
Agente/Ator: Gerente
Pré-condições: If código de acesso do usuário inválido
C5.2[3]:
1. O usuário entra com código de acesso
2. O sistema exibe mensagem “Código de acesso inválido” para o usuário
3. if nº entradas de códigos de acesso = 3
4. O sistema trava entrada do usuário na rede
5. O sistema exibe mensagem "Procure administrador da rede" para o usuário.
Pós-condições: DO gráficos estatísticos não mostrados.

RC5.2[4]
G5.2[4]: Consultar gráficos estatísticos com senha inválida.
Agente/Ator: Gerente
Pré-condições: If código de acesso do usuário válido
If Senha do usuário inválida
C5.2[4]:
1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha inválida
4. O sistema exibe mensagem “Senha inválida” para o usuário
5. if nº entradas de senhas inválidas = 3
6. O sistema trava entrada do usuário na rede
7. O sistema exibe mensagem “Procure administrador da rede" para o usuário.
Pós-condições: DO gráficos estatísticos não mostrados.

Nível físico
RC1.1.1
G1.1.1: Verificar validade do código de acesso.
Pré-condições: If Código de acesso digitado
Agente/Ator: sistema
C1.1.1:
1. Abrir tabela de usuário no BD.
2. Pesquisar código de acesso na tabela.
3. Se código de acesso encontrado, então mostrar mensagem "código de acesso válido" para o usuário.
4. Senão exibir mensagem "código de acesso inválido" para o usuário.
5. Fechar tabela.
Pós-condição: DO Código de acesso validado.

RC1.1.2
G1.1.2: Verificar validade da senha.
Pré-condições: If senha digitada
Agente/Ator: sistema
C1.1.2:
1. Abrir tabela de usuário no BD.
2. Pesquisar senha por código de acesso na tabela.
3. Se senha encontrada, então mostrar mensagem “Faltam x dias para expirar sua senha”.
4. Senão exibir mensagem "Senha não confere" para o usuário.

5. Fechar tabela.
Pós-condição: DO senha verificada

RC1.1.3
G1.1.3: Verificar validade do código de cardápio.
Pré-condições: If Código do cardápio digitado
Agente/Ator: sistema
C1.1.3:
1. Abrir tabela de cardápio no BD.
2. Pesquisar código do cardápio na tabela.
3. Se código do cardápio encontrado, então mostrar mensagem “código do cardápio válido" para o usuário.
4. Senão mostrar mensagem "código do cardápio inválido" para o usuário.
5. Fechar tabela.
Pós-condição: DO código do cardápio verificado.

RC1.2.1
G1.2.1: Verificar validade do código de acesso
Pré-condições: If código de acesso digitado
Agente/Ator: sistema
C1.2.1:
1. Abrir tabela de usuário no BD.
2. Pesquisar código de acesso na tabela.
3. Se código de acesso encontrado, então mostrar mensagem "código de acesso válido" para o usuário.
4. Senão exibir mensagem "código de acesso inválido" para o usuário.
5. Fechar tabela.
Pós-condição: DO código de acesso verificado

RC1.2.2
G1.2.2: Verificar validade da senha
Pré-condições: If Senha digitada
Agente/Ator: sistema
C1.2.2:
1. Abrir tabela de usuário no BD.
2. Pesquisar senha por código de acesso na tabela.
3. Se senha encontrada, então mostrar mensagem “Faltam x dias para expirar sua senha”.
4. Senão exibir mensagem "Senha não confere" para o usuário.
5. Fechar tabela.
Pós-condição: DO senha verificada

RC1.2.3
G1.2.3: Verificar validade do código de cardápio
Pré-condições: If código do cardápio digitado
C1.2.3:
1. Abrir tabela de cardápio no BD.
2. Pesquisar código do cardápio na tabela.
3. Se código do cardápio encontrado, então mostrar mensagem “código do cardápio válido" para o usuário.
4. Senão mostrar mensagem "código do cardápio inválido" para o usuário.

5. Fechar tabela.
Pós-condição: DO código do cardápio verificado.

RC1.2.4
G1.2.4: Verificar duplicidade do cardápio
Pré-condições: If código do cardápio digitado
Agente/Ator: sistema
C1.2.4:
1. Abrir tabela de cardápio no BD.
2. Pesquisar código do cardápio na tabela.
3. Se código do cardápio encontrado, então mostrar mensagem “código do cardápio duplicado" para o usuário.
4. Senão mostrar mensagem "código do cardápio inexistente" para o usuário.
5. Fechar tabela.
Pós-condição: DO código do cardápio verificado.

Assim continua a descrição dos cenários do nível físico para todos os RCs comportamentais.

Após elicitados os objetivos e elaborados os cenários, pode-se seguir várias possibilidades. Uma delas é refinar objetivos e cenários (estratégia de refinamento), onde é verificada a existência de inconsistências, incompletezas, redundâncias internas nos cenários e objetivos assim como a eliminação de objetivos duplicados e a unificação de sinônimos. A outra possibilidade é operacionalizar objetivos e cenários por meio de esquemas, aplicada em nosso exemplo posteriormente.

A possibilidade indicativa do fluxo natural do modelo de processo é a fase de reorganização de objetivos e cenários. Nesta fase, como explicado no capítulo anterior, consiste em estabelecer relações de dependência entre objetivos e cenários, sendo utilizada a relação de dependência por precedência, a relação de dependência por contrato ou a dependência entre agentes.

Os objetivos normais do nível comportamental são listados, juntamente com seu código identificador.

G1.1 - Receber cardápio de uma maneira normal
G1.2 - Verificar duplicidade de uma maneira normal
G1.3 - Sugerir alternativa de uma maneira normal
G2.1 - Cadastrar fornecedores de uma maneira normal
G2.2:- Cadastrar produtos de uma maneira normal
G2.3 - Cotar produtos de uma maneira normal
G3.1 - Verificar cotação de uma maneira normal
G3.2 - Fazer pedido de compra de uma maneira normal

G4.1 - Efetuar distribuição de uma maneira normal
G5.1 - Preencher questionários de uma maneira normal
G5.2 - Consultar gráficos estatísticos de uma maneira normal

Aplicando a estratégia de reorganização no estudo de caso de restaurantes, verifica-se que existem as seguintes relações de dependência por precedência, para o nível comportamental.

G1.1 < G1.2 < G1.3

G2.1 ^ G2.2 < G2.3

G2.3 < G3.1

[G3.1] < G3.2 (opcional, pode ser que seja necessário verificar cotação ou não, antes de fazer o pedido de compra)

G5.1 ^ G2.1 ^ G2.2 ^G1.1 ^ G2.3 ^ G4.1 < G5.2

Um exemplo de dependência entre agentes no exemplo ilustrado ocorre no nível comportamental onde o ator cliente necessita preencher questionários, cadastrar produtos, receber cardápios e efetuar distribuição primeiramente, antes que o ator gerente realize o objetivo funcional “Consultar gráficos estatísticos de uma maneira normal".

Dependência entre agentes também pode ser verificada entre “Verificar Cotação” e “Fazer Pedido de Compra”, onde o agente comprador deve primeiramente verificar cotação para só então fazer pedido de compra. Da mesma forma, o ator comprador deve realizar o caso de uso “Cotar Produtos” para que posteriormente o restaurante possa verificar cotação.

E o outro exemplo ocorre para o ator restaurante, no nível funcional, que deve realizar o caso de uso “Receber Cardápio” antes do caso de uso “Verificar duplicidade”, que precede a realização do caso de uso “Sugerir alternativa”. Ou seja, primeiramente o restaurante deve receber os cardápios de todos os restaurantes filiais. Caso haja alguma duplicidade, são enviadas alternativas de substituição para os mesmos.

No caso de dependência por contrato, podemos exemplificar que o objetivo funcional G1.2 aciona automaticamente o objetivo funcional G1.3 “Sugerir alternativa”, uma vez que quando cardápios duplicados são encontrados, o sistema mostra os cardápios alternativos existentes para o agente.

A estratégia de refinamento foi aplicada sobre o exemplo proposto, de maneira que foi eliminada uma redundância dentro do RC4 para o nível funcional. Anteriormente, o objetivo G4: Distribuir alimentos era composto pelos casos de uso “Efetua Distribuição" e “Efetuar Recebimento". Com a aplicação da estratégia foi constatado que os casos de uso eram sinônimos, não havendo necessidade de manter os dois. Assim, foi eliminado o caso de uso “efetuar recebimento", ficando assim um modelo mais representantivo e consistente com os requisitos reais dos clientes.

Na fase de operacionalização de objetivos e cenários, estes foram descritos em mais detalhes e mapeados para seus respectivos esquemas. A seguir foram descritos alguns esquemas-objetivos, esquemas-casos-de-uso, esquemas-cenários e esquemas-ações do exemplo proposto.

- **Esquemas-objetivos**

Nº do RC: RC1
Código do objetivo: G1
Nome do objetivo: Controlar cardápio
Nível: funcional
Tipo: serviço
Descrição: consiste em gerenciar todos as solicitações de cardápios enviadas pelos restaurantes filiais
Agente/Ator: restaurante
Cod-cenário/Cod-caso de uso: MUC1.1

Nível funcional

Nº do RC: RC1.1
Cod-obj: G1.1
Objetivo: Receber cardápio de uma maneira normal
Nível: comportamental
Tipo: funcional
Descrição: Consiste no recebimento dos cardápios enviados pelos restaurantes filiais
Agente/Ator: restaurante
Cod-cenário/Cod-caso de uso: C1.1

Nível comportamental

Nº do RC: RC1.1.1
Cod-obj: G1.1.1
Objetivo: Verificar validade do código de acesso
Nível: interno
Tipo: sistema
Descrição: Consiste em verificar se o código de acesso é válido
Agente/Ator: sistema
Cod-cenário/Cod-caso de uso: C1.1.1

Nível físico

- **Esquema-cenário**

Cod-cenário: C1.1

Nível: comportamental
Tipo: normal
Descrição:
Pré-condições: If código de acesso do usuário válido
If Senha do usuário válido
If Cod- cardápio válido
Pós-condições: Do status do cardápio = “recebido"
Agente: restaurante
Ação:

1. O usuário entra com código de acesso
2. O sistema valida código de acesso.
3. O usuário entra com senha
4. O sistema valida senha
5. O usuário entra com código do cardápio
6. O sistema valida código do cardápio.
7. O usuário confirma recebimento
8. O sistema exibe mensagem “Cardápio recebido com sucesso" para o usuário.

- **Esquema-ação**

Ação: O sistema valida código de acesso
Tipo: sistema
Reads: código
Changes:
Nº da sequência: 1
Cod-cenário: C1.1

5. IMPLEMENTAÇÃO DA FERRAMENTA

Este capítulo apresenta a implementação da ferramenta de suporte ao método de elicitação e modelagem de requisitos baseado em objetivos, apresentado no capítulo quatro [13]. A ferramenta é semi-automatizada, implementando algumas fases do método, de acordo com algumas estratégias ilustradas. A ferramenta foi desenvolvida utilizando o Power Designer Data Architect, Delphi 5.0 [11,21] e Access 2000.

Inicialmente será mostrado o modelo de processo do método proposto, ilustrando quais fases e estratégias foram implementadas pela ferramenta. Posteriormente, será visualizado o modelo entidade-relacionamento do protótipo da ferramenta, ao que se seguirá um manual de operação da mesma. Alguns exemplos de ferramentas podem ser encontradas em [6,7,9,50].

5.1 Fases e estratégias implementadas

Conforme indica a figura. 5.1, o modelo de processo é composto das seguintes fases: Elicitar um objetivo, especificar casos de uso, elaborar cenários, reorganizar objetivos e cenários, refinar objetivos e cenários e operacionalizar objetivos e cenários até o produto final que é o documento de requisitos de software.

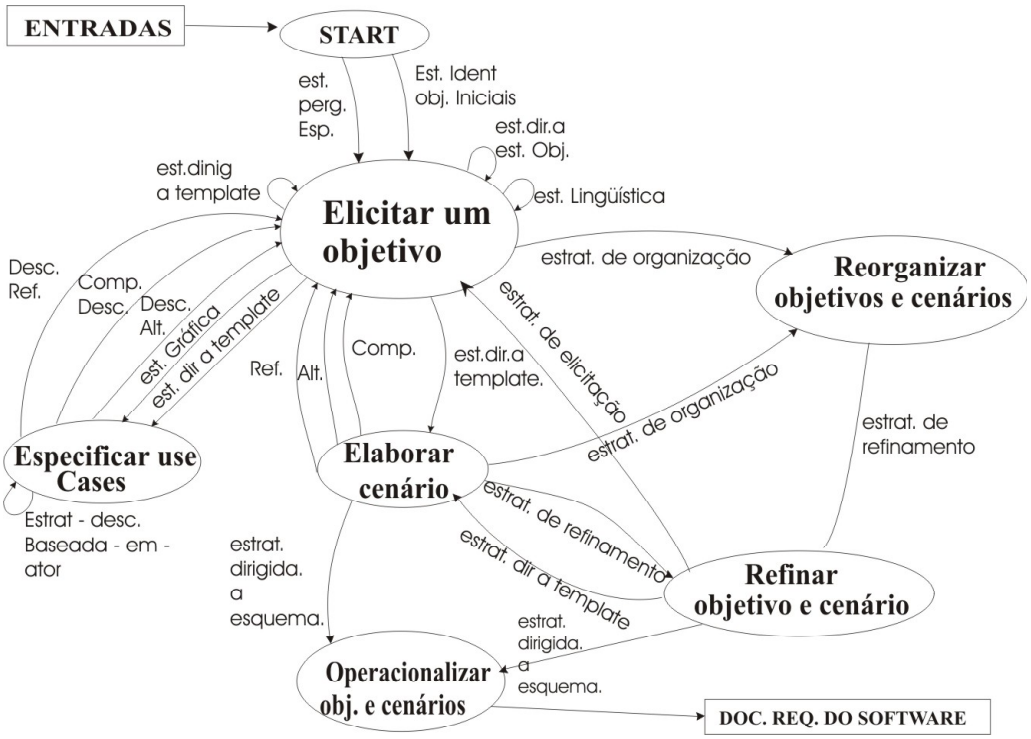

Figura. 5.1 – Modelo de processo do Método Proposto com o fluxo do processo em negrito implementado pela ferramenta.

Na figura 3.2 do modelo de processo do método criado no capítulo três existem as fases de especificação de cenários e operacionalização de objetivos e cenários. Na implementação da ferramenta essas fases foram unificadas e realizadas de uma única vez.

A ferramenta implementada pressupõe que tenham sido aplicadas as estratégias de perguntas específicas e de identificação dos objetivos iniciais conforme descritas no Capítulo quatro.

O protótipo da ferramenta tem seu início a partir da aplicação da estratégia dirigida a template/estratégia dirigida a estrutura do objetivo, onde são elicitados os objetivos de serviço. Posteriormente, a ferramenta utiliza a estratégia dirigida a template para especificação dos casos de uso relacionados.

De acordo com o fluxo do processo da ferramenta implementada, esta segue elicitando os objetivos funcionais, elabora os cenários estruturais, elicita os objetivos internos do sistema e elabora os cenários físicos. Todos esses elementos são operacionalizados produzindo o documento de requisitos de software.

5.2 Arquitetura do protótipo da ferramenta

A ferramenta de suporte ao Método de Elicitação e Modelagem de requisitos baseado em objetivos foi dividida em várias partes e aplicada sobre o estudo de caso de restaurantes, conforme ilustra a figura 5.2.

A primeira fase corresponde ao núcleo do método (fases de elicitação dos objetivos, cenários e casos de uso nos níveis funcional, comportamental e físico) cujo modelo do processo é descrito na figura 3.2. A segunda fase destina-se à manutenção das tabelas do sistema. A terceira fase corresponde a reorganização de objetivos e cenários (vide figura 3.2), onde são implementadas as relações de dependência e contrato entre objetivos.

Posteriormente são gerados relatórios, inclusive o documento de requisitos de software - produto principal do processo.Todas essas fases serão melhor detalhadas na seção 5.3.

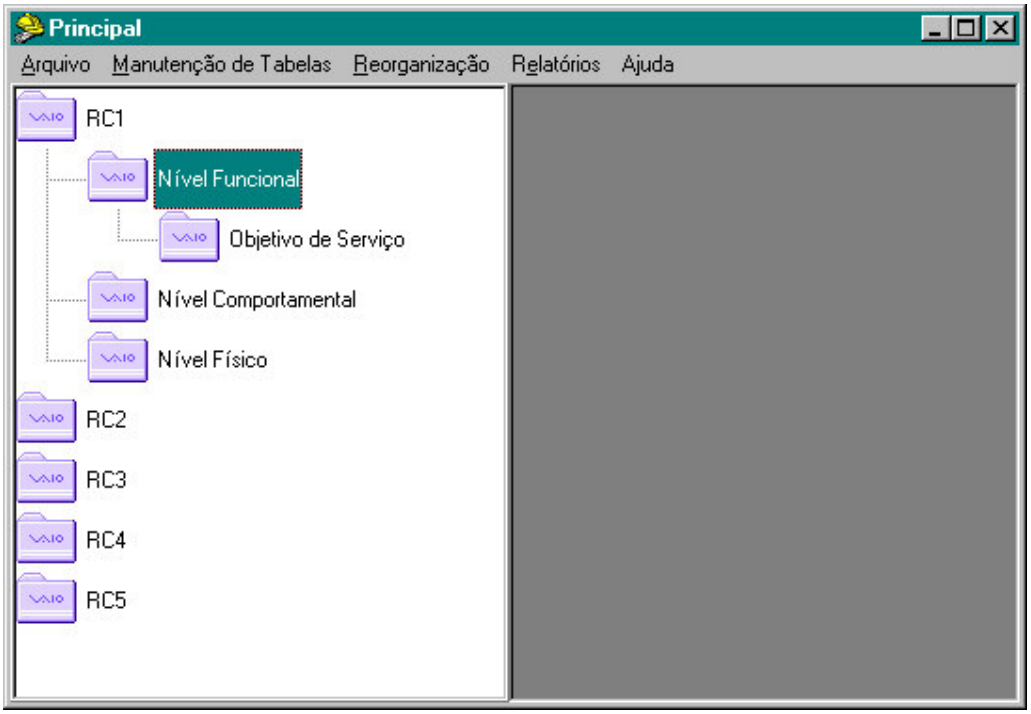

Figura 5.2 – Interface principal do protótipo da ferramenta.

A estrutura da base de dados pode ser vista na figura. 5.3 e mostra as entidades do modelo conceitual e seus relacionamentos, de acordo com a notação encontrada em [36].

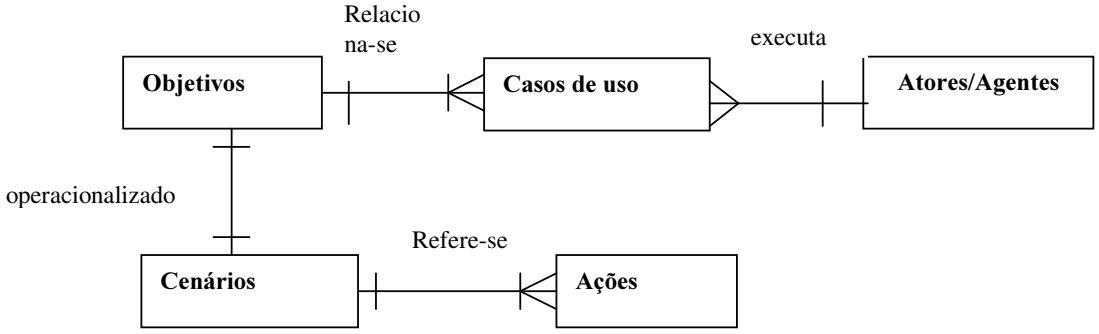

Figura. 5.3 – Modelo Entidade Relacionamento Geral da Ferramenta.

De acordo com a figura 5.3, cada objetivo está relacionado a um ou mais casos de uso e operacionalizado por apenas um cenário. Um cenário está relacionado a apenas um objetivo e é descrito por uma ou várias ações, onde uma ação refere-se a somente um cenário. Cada caso de uso refere-se a somente um objetivo e é executado por apenas um ator. Por outro lado, um ator pode executar vários casos de uso.

Ao implementarmos a ferramenta, foi detectado que a estrutura da base de dados, ilustrada na figura 5.4 seria mais adequada, por oferecer uma maior facilidade na

implementação. Assim, a entidade objetivo foi dividida em objetivos de serviço, objetivos funcionais e objetivos do sistema. Por sua vez, a entidade cenário foi dividida em cenário estrutural e cenário físico, como pode ser visto na figura 5.4.

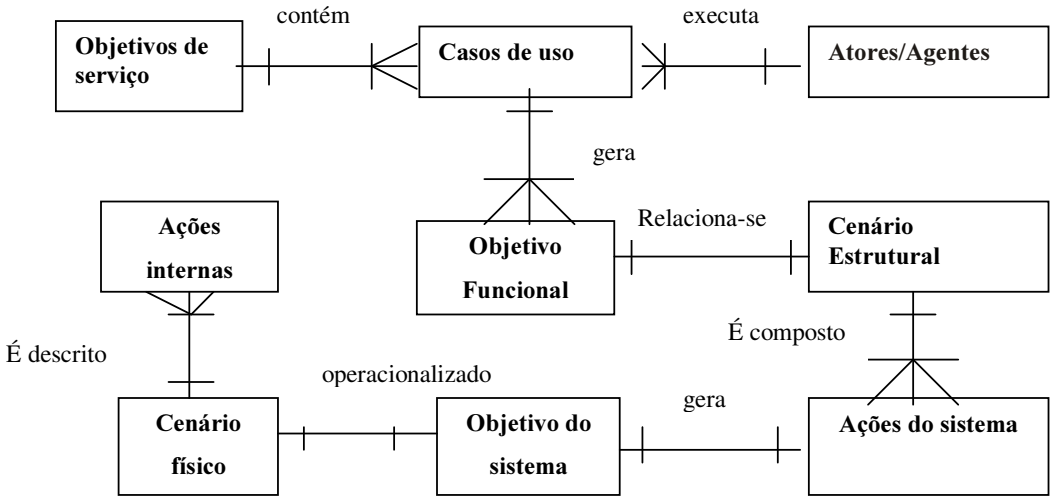

Figura. 5.4 – Modelo Entidade Relacionamento Detalhado da Ferramenta.

De acordo com a figura 5.4, um objetivo de serviço está relacionado a um ou mais casos de uso, onde um caso de uso refere-se a somente um objetivo de serviço. Um caso de uso gera um ou mais objetivos funcionais, que está relacionado a somente um cenário estrutural. Um cenário estrutural é descrito por várias ações. Cada ação do sistema dará origem a um objetivo do sistema, operacionalizado por um cenário físico. Um cenário físico é descrito por várias ações internas.

5.3 Manual de operação

A ferramenta possui o intuito de agilizar o método proposto, semi-automatizando algumas fases. Inicialmente são elicitados os objetivos de serviço (vide figura. 5.5), onde são inseridos o código do RC, o código do objetivo, o nome do objetivo e sua descrição. O caminho consiste em Arquivos\Abrir projeto\RC1\Nível funcional\Objetivo de serviço e corresponde a elicitar um objetivo, utilizando a estratégia dirigida a template, de acordo com a figura 3.2.

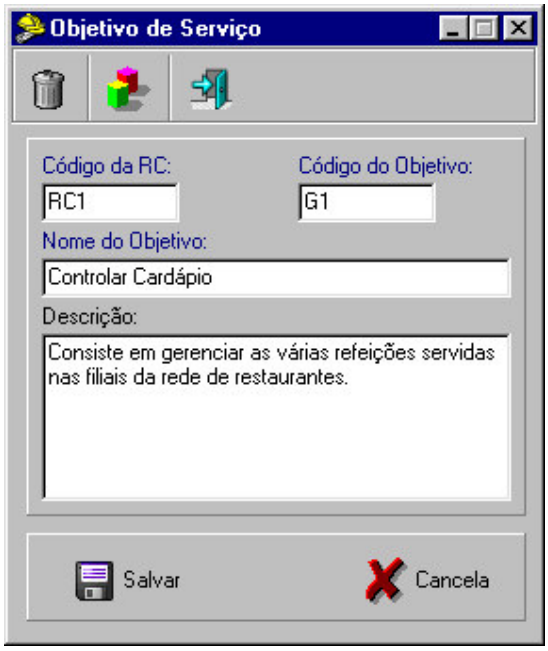

Figura. 5.5 – Tela para elicitação dos objetivos de serviço pela estratégia dirigida a template.

Após elicitados os objetivos de serviço, o próximo passo consiste em especificar os casos de uso correspondentes, utilizando a estratégia dirigida a template, como pode ser visto na figura. 5.6, onde são inseridos os nomes dos casos de uso pertencentes àquele RC, assim como os atores que os executam e uma pequena descrição do que faz cada caso de uso.

Na descrição do processo (vide figura 3.2), essa parte corresponde a Especificar casos de uso – estratégia dirigida a template.

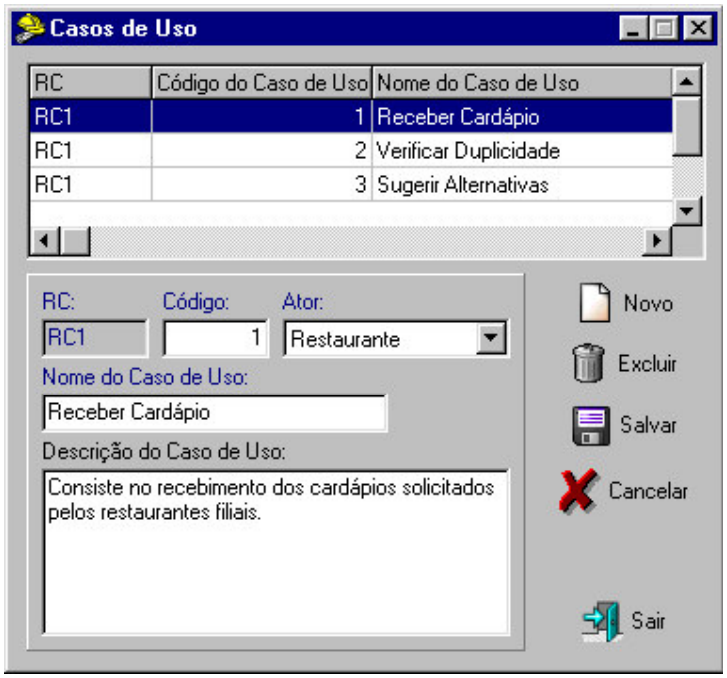

Figura. 5.6 - Tela para especificação dos casos de uso pela estratégia dirigida a template.

Na figura 5.7 são elicitados os objetivos funcionais, utilizando a estratégia de refinamento e especificados os cenários estruturais, pela estratégia dirigida a template. A estratégia alternativa que permite a geração dos objetivos e cenários variacionais e excepcionais é aplicada através da mudança de índice de 0 para 1, 2,..., já que 0 indica normal e as numerações sucessivas referem-se aos RCs alternativos.

A estratégia de composição é aplicada quando elicita-se um novo RC funcional, composto pelo objetivo de serviço e casos de uso associados.

As ações do cenário estrutural são cadastradas na fig. 5.8. As pré-condições e pós-condições são selecionadas respectivamente, de acordo com as telas das figuras 5.9 e 5.10. O exemplo refere-se as pré e pós condições do cenário estrutural “Receber cardápio de uma maneira normal”

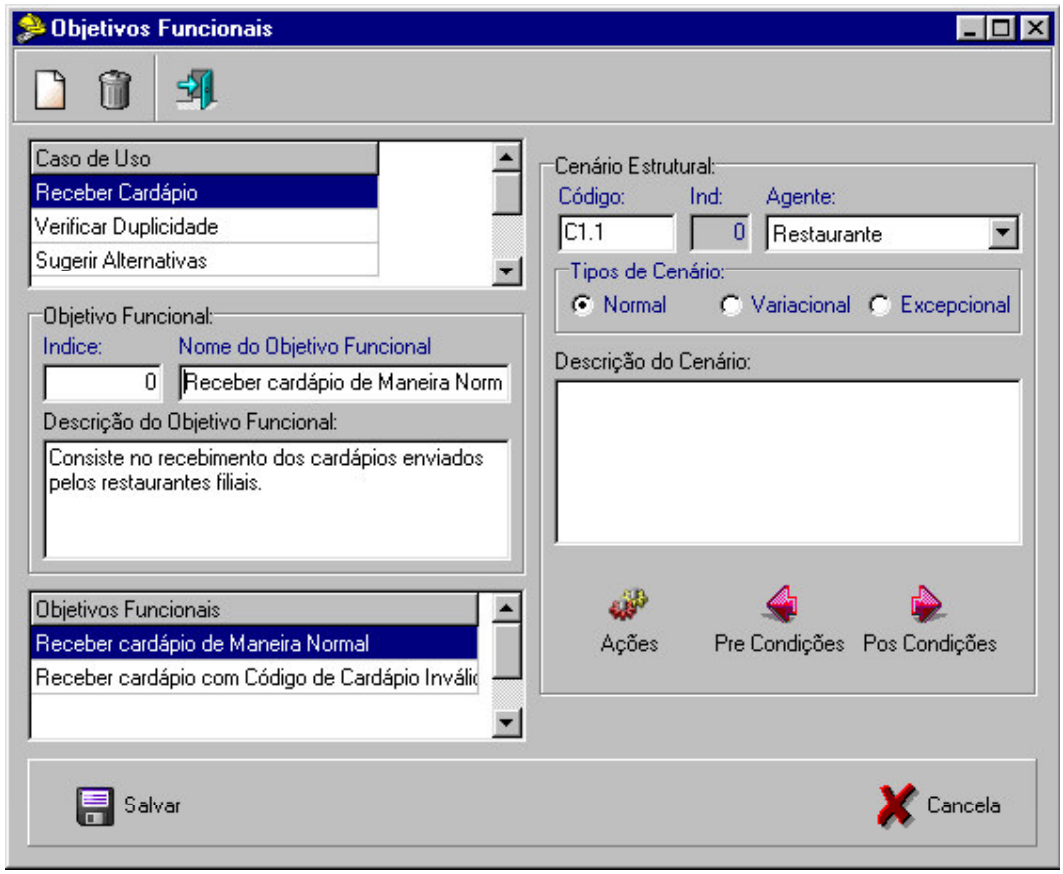

Figura. 5.7 – Tela para elicitação dos objetivos funcionais e especificação dos cenários estruturais.

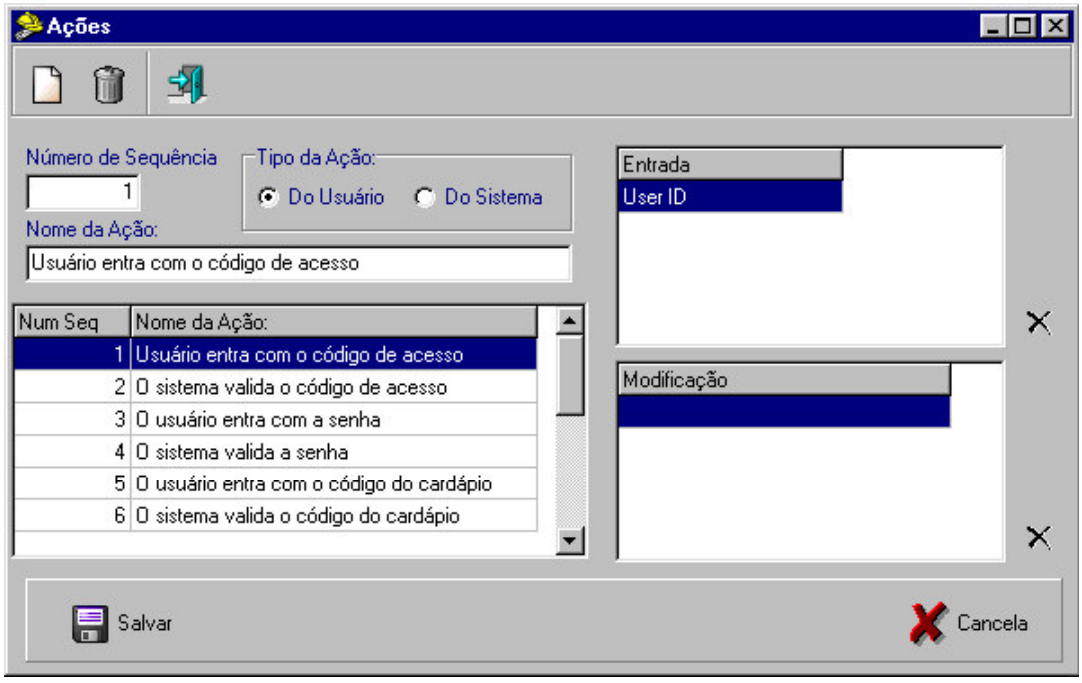

Figura. 5.8 – Tela para cadastramento das ações dos cenários estruturais.

Dentro do cenário estrutural devem ser cadastradas as ações com seu número de seqüência dentro deste cenário, o tipo de ação (usuário/sistema), o nome da ação, com suas entradas e modificações, como ilustra a figura. 5.8.

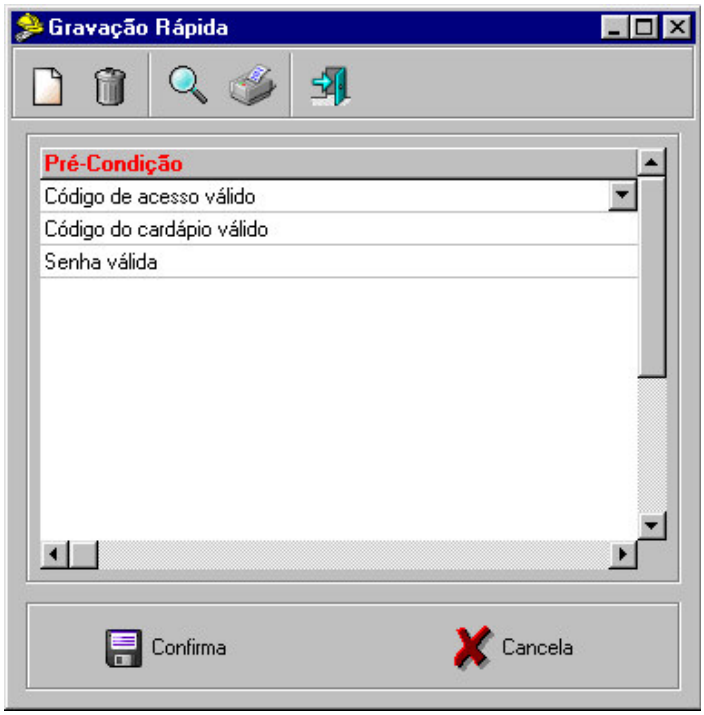

Figura. 5.9 – Tela para seleção das pré-condições.

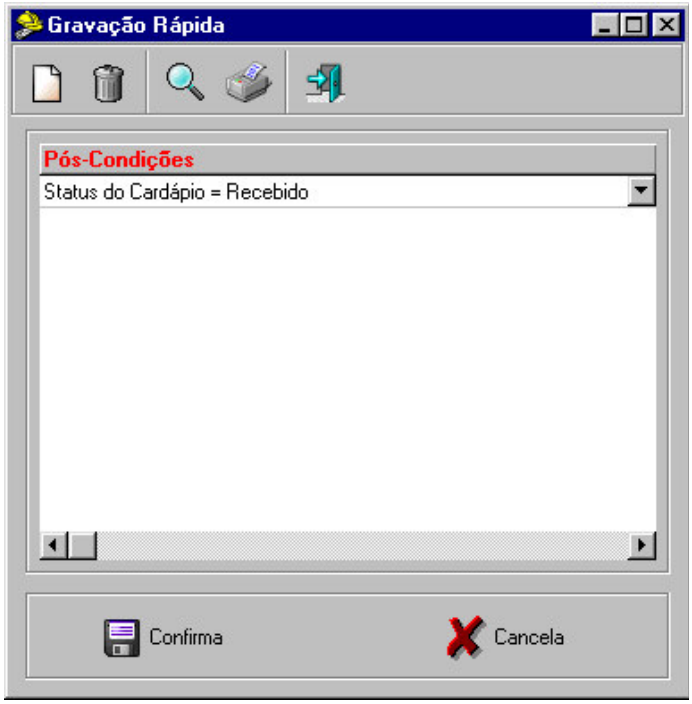

Figura. 5.10 – Tela para seleção das pós-condições.

Para elicitação dos objetivos do sistema, deve ser escolhida, dentro das ações do sistema, a ação que irá originar o nome do objetivo do sistema. A seguir pode ser vislumbrado o preenchimento dessa tela (vide figura. 5.11) com código do objetivo do sistema, a ação que dará origem ao objetivo do sistema e o nome do objetivo do sistema.

Ainda na figura. 5.11 são especificados os cenários físicos, e acionamos as telas para cadastramento das ações internas (vide figura 5.12), assim como as seleções das pré e pós condições dos cenários físicos.

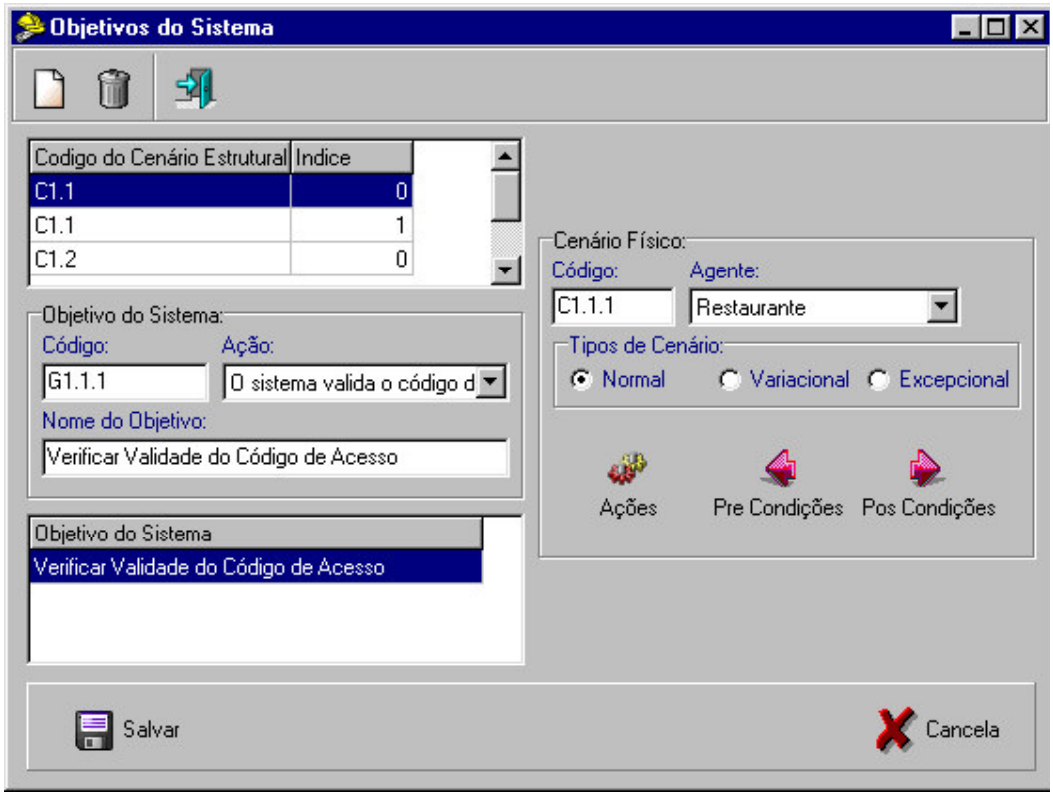

Figura. 5.11 – Tela para elicitação dos objetivos do sistema e especificação dos cenários físicos.

Na tela para cadastramento das ações do cenário físico, deve ser digitado o número de seqüência da ação dentro do cenário, o nome da ação, suas entradas e modificações (vide figura. 5.12).

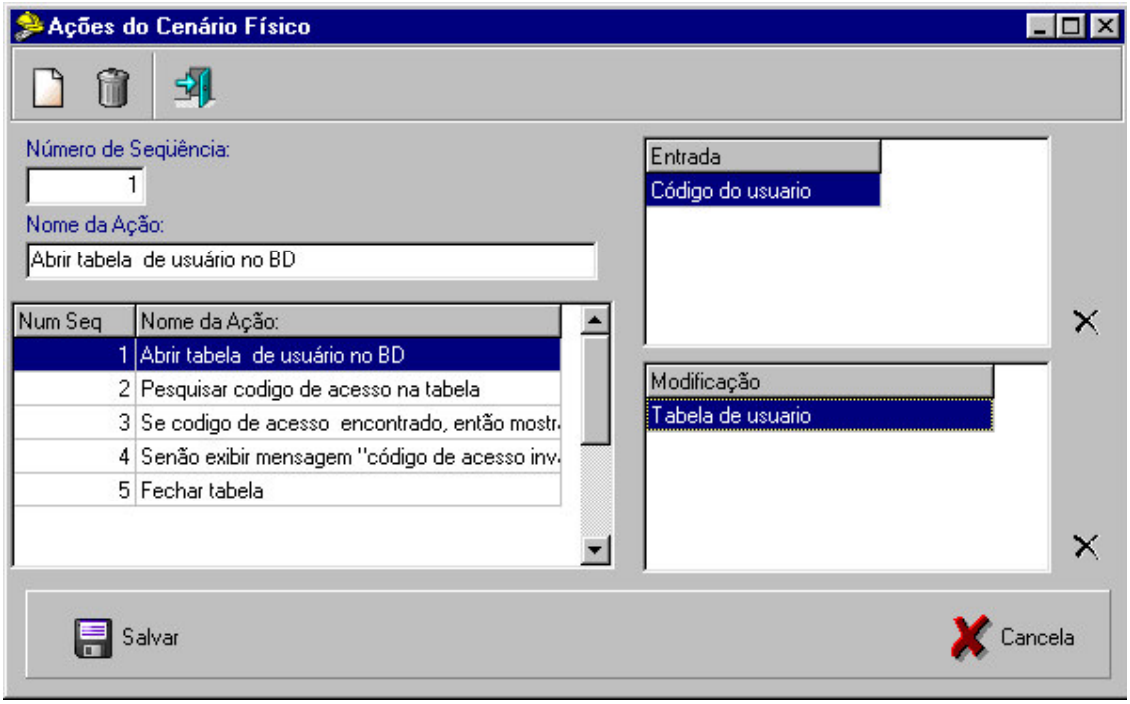

Figura. 5.12 - Tela para cadastramento das ações dos cenários físicos.

Na fase de manutenção de tabelas, podem ser criados novos RCs funcionais, (vide fig. 5.13) utilizando neste caso a estratégia dirigida a template.

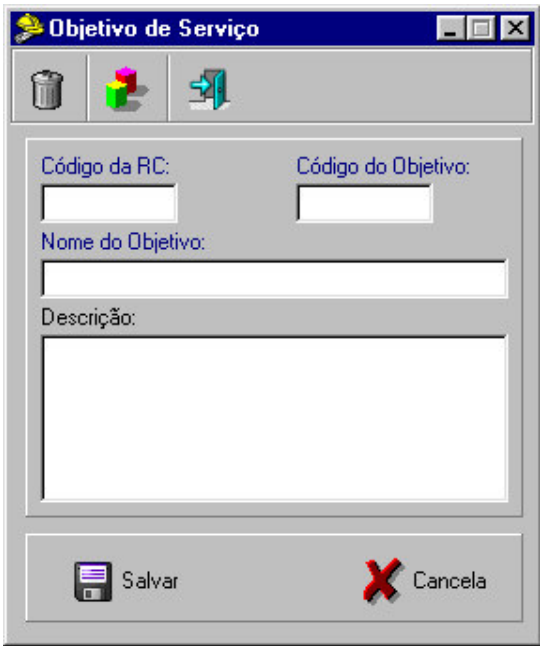

Figura. 5.13 – Tela para inclusão de um novo RC funcional pela estratégia dirigida a template.

Dentro de manutenção de tabelas, encontra-se também o formulário (vide figura. 5.14) para manutenção do cadastro dos atores e/ou agentes, responsáveis pela execução do caso de uso e/ou cenários.

Figura. 5.14 – Tela para manutenção do cadastro dos atores/agentes.

A próxima fase implementada pela ferramenta é a fase de reorganização de objetivos e cenários. Na figura. 5.15 e nas posteriores explicaremos como se processa essa fase utilizando a ferramenta de suporte ao método.

O reorganizador de objetivos e cenários monta as regras, dependendo do tipo de relação de dependência e dos objetivos escolhidos. Na fig. 5.15 é ilustrado um exemplo de dependência por precedência, já explicado no capítulo quatro. Dessa forma, "Cotar produtos" deve ocorrer antes de "Verificar cotação", o que é indicado pela notação G2.3 < G3.1, gerado no campo regra.

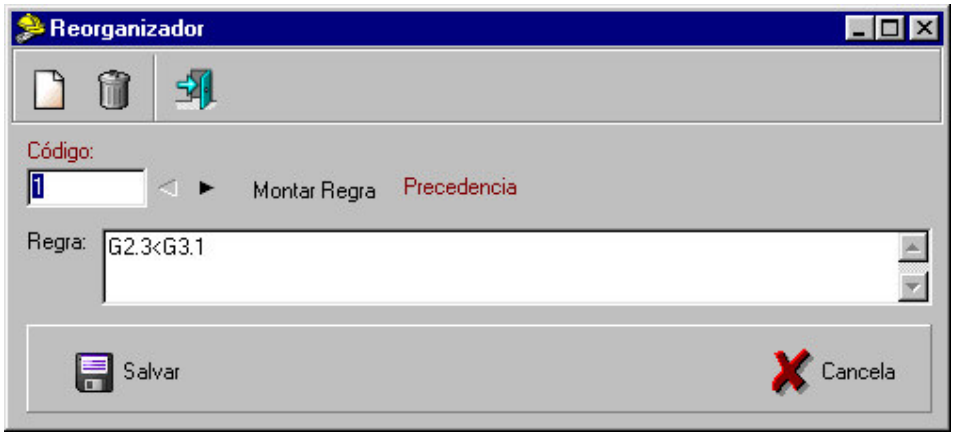

Figura 5.15 – Exemplo de dependência por precedência.

Na figura 5.16 é escolhido o tipo de relação de dependência e na figura 5.17 os objetivos envolvidos na montagem da regra.

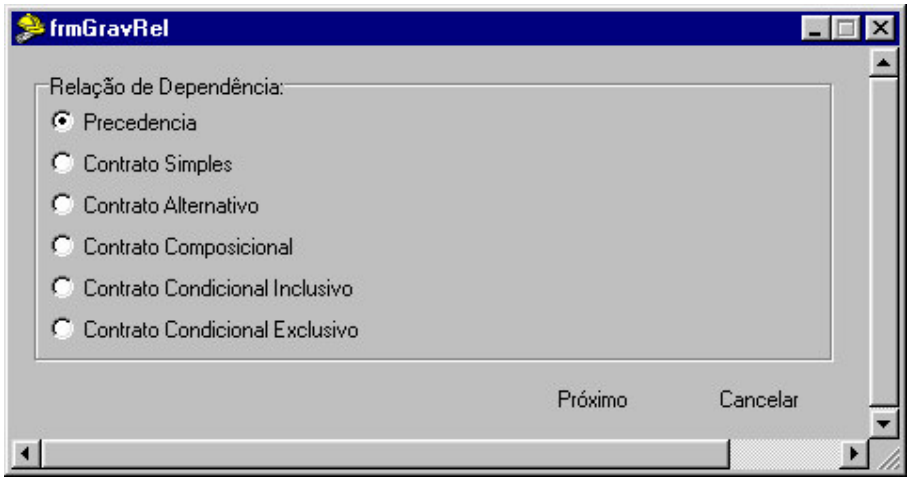

Figura 5.16 – Tela de escolha do tipo de relação de dependência.

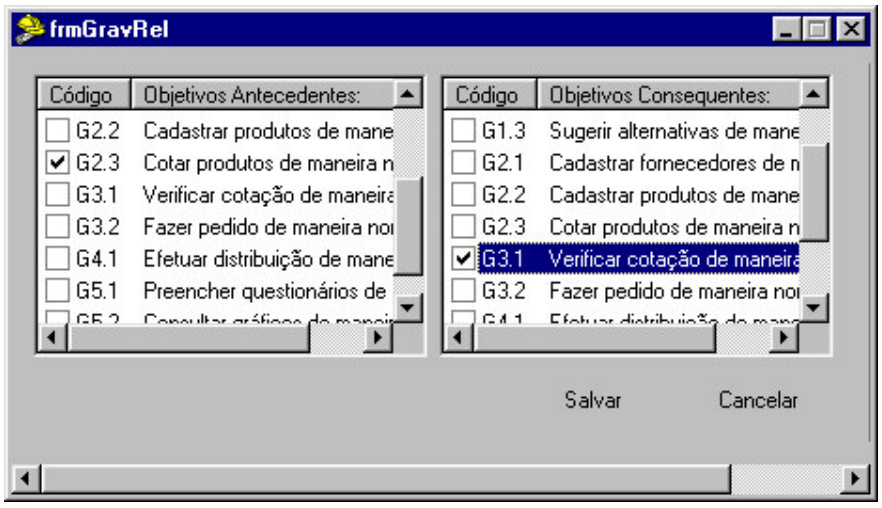

Figura 5.17 – Tela para escolha dos objetivos envolvidos na montagem da regra.

Na figura 5.18, a dependência por contrato simples é exemplificada. Assim o objetivo funcional G1.2 “Verificar duplicidade de uma maneira normal” aciona automaticamente o objetivo funcional G1.3 “Sugerir alternativa de maneira normal”.

Figura 5.18 – Exemplo de dependência por contrato simples.

A última fase da ferramenta consiste na geração do produto final do método que é o documento de requisitos. O documento de requisitos gerado encontra-se na figura 5.19 e servirá de base para as demais fases do desenvolvimento do sistema.

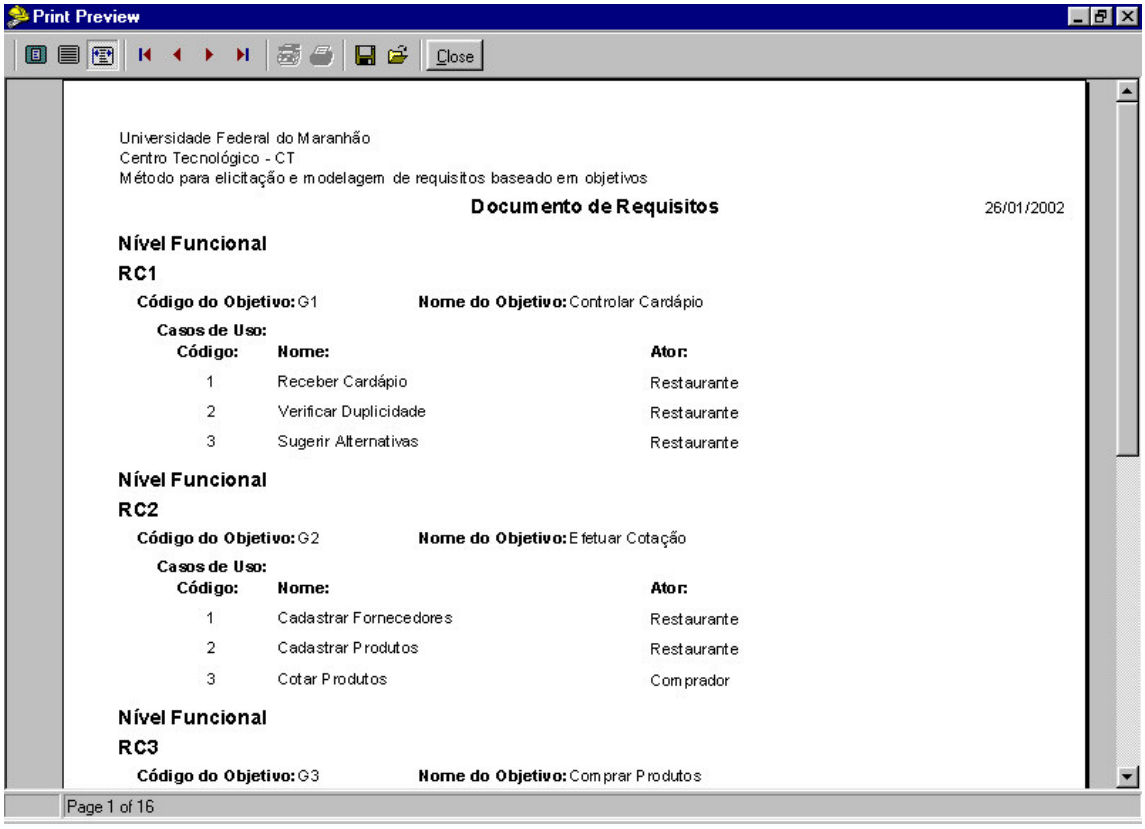

Figura. 5.19 – Tela para visualização do documento de requisitos gerado pela ferramenta.

6. CONCLUSÃO

6.1 Contribuições do trabalho

Esse trabalho teve como objetivos estudar algumas abordagens para Engenharia de Requisitos direcionadas a casos de uso, cenários, objetivos e apresentar uma proposta de um método para a fase de Engenharia de Requisitos. Também foi realizado um estudo comparativo entre as diversas abordagens, de maneira a se obter uma visão crítica das mesmas.

A abordagem criada tem como um dos seus principais aspectos a utilização de um acoplamento bidirecional entre objetivo e caso de uso. Nesse processo, à medida em que é descoberto um objetivo de serviço, este é operacionalizado por casos de uso. Na direção reversa, ou seja, de casos de uso para objetivos, a abordagem guia o processo de elicitação de requisitos descobrindo objetivos funcionais a partir dos casos de uso.

Os objetivos funcionais são operacionalizados por cenários estruturais. Na direção cenários-objetivos, as ações do sistema geram os objetivos do sistema concretizados pelos cenários internos do sistema.

A abordagem proposta propõe combinar objetivos, casos de uso e cenários como um mecanismo para elicitar os requisitos de um sistema. Ela é proveniente da integração da abordagem CREWS-L´Ecritoire [45] com o trabalho de Regnell et al [42,43] e o Método GBRAM [4,5], consistindo em um modelo de processo e um modelo de produto, aplicado em um estudo de caso para elicitar os requisitos de um sistema para restaurantes.

Com a integração desses métodos, observa-se que o modelo RC da abordagem CREWS-L`Ecritoire foi extendido para <objetivo, casos de uso>. Casos de uso são descritos a partir da visão de um único ator, permitindo que cenários normais, excepcionais e variacionais sejam integrados em um modelo de uso sintetizado. Dessa forma, a obtenção de cenários origina-se da aplicação de uma estratégia de refinamento sobre os casos de uso.

No que concerne ao Método GBRAM, foram extendidas as relações de dependência de contrato entre objetivos para dependência alternativa, dependência composicional, dependência condicional exclusiva e inclusiva, assim como foi introduzida a técnica de caso de uso, conforme descreve Regnell et al em [42,43] e extendida a noção de

cenários para caracterizar situações não só excepcionais, como as normais e variacionais dos objetivos.

Uma ferramenta de suporte ao desenvolvimento do método proposto foi desenvolvida e aplicada sobre o exemplo de restaurantes. Nessa ferramenta, foram implementadas as fases de elicitação de objetivos, especificação de casos de uso, especificação de cenários, reorganização de objetivos e cenários e operacionalização de objetivos e cenários, até a geração do documento de requisitos.

O método para elicitação e modelagem de requisitos apresentado nesta dissertação possui limitações. Uma delas consiste em não suportar ações e agentes concorrentes. As ações suportadas são do tipo seqüência, alternativa e iterativa.

Os casos de uso devem ser executados por somente um ator/agente. A recursividade, assim como as combinações das relações de dependência entre objetivos em expressões, tais como G1→ (G1 Λ G2) v G3 também não foram inseridas. A abordagem alcança somente requisitos funcionais; os requisitos não funcionais não foram considerados.

6.2 Trabalhos futuros

Sugerem-se como trabalhos futuros a extensão desse método para as fases de integração, formalização, verificação e validação dos objetivos e cenários. A fase de integração concentra-se em unificar os vários cenários escritos de acordo com os critérios selecionados, que podem ser atores, casos de uso, promovendo dessa forma uma visão geral do comportamento do sistema com uma visão única por ator. Formas de integração de cenários podem ser encontradas em [20,22].

A integração de cenários deve ser realizada somente no nível comportamental a partir de critérios estabelecidos no nível funcional. Assim, depois de elaborados os objetivos e cenários, estes podem ser formalizados ou integrados, seja através de uma estratégia de formalização ou integração a serem definidas futuramente. Posteriormente à formalização destes cenários, estes podem ser submetidos à verificação utilizando uma estratégia de verificação e correção para que possam assim serem validados**.**

A fase de validação consiste em, juntamente com o usuário, validar os objetivos e cenários produzidos de maneira à obtenção de um documento de requisitos de

software que satisfaça aos clientes, seja não ambíguo e consistente. Uma sugestão de extensão à abordagem proposta pode ser encontrada em [12].

Com relação a fase de reorganização de objetivos e cenários, foram definidas as relações de dependência de contrato alternativa, composicional, condicional exclusiva e inclusiva. O ideal é que sejam acrescentadas ao método todas as combinações possíveis de relações entre objetivos, criando assim novas regras, ainda específicas, mas que posteriormente possam ser generalizadas e inseridas em uma base de regras.

Uma outra área interessante no qual esse trabalho poderia ser extendido seria a de requisitos não funcionais. Assim, o método englobaria aspectos funcionais e não funcionais dos requisitos. Em [15] pode ser encontrado um exemplo para requisitos não funcionais utilizando objetivos de forma gráfica.

A transição da engenharia de requisitos para a fase de análise é pouco comentada na literatura. Dessa forma, o mapeamento da fase de engenharia de requisitos para métodos orientados a objetos (UML, Fusion, Booch, entre outros) seria de muita utilidade. Em [37,38] foi realizado um modelo de requisitos para o Método Fusion. Em [10] são apresentadas heurísticas de transformação que auxiliam na passagem do modelo de requisitos de software baseado em cenários para os modelos de análise definidos pela UML (Linguagem de Modelagem Unificada).

Dentro desse contexto, em [16] é proposta uma estratégia para lidar com requisitos não funcionais desde as primeiras etapas do processo de desenvolvimento de software, que possibilita não apenas a elicitação dos requisitos não funcionais (RNFs), mas também a plena integração destes aos modelos conceituais que representam o software.

Tornar a abordagem proposta reutilizável seria recomendável, pois usufruiria dos principais benefícios da reutilização que são economia de tempo, redução de custos e minimização dos esforços da equipe desenvolvedora.

Como foi desenvolvido apenas um protótipo da ferramenta de suporte ao método, a extensão da implementação constitui uma outra sugestão de trabalhos futuros.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ACHOUR, C. Ben, Tawbi, Mustapha, SOUVEYET, Carine. *Bridging the gap between users and requirements engineering: the scenario-based approach.* Submitted to Computer Systems Journal, 1999.

[2] ACHOUR, C. Ben. *Guiding scenario authoring*. Submitted to the Euro-Japanese conference, 1998.

[3] ACHOUR, C. Ben, ROLLAND, Collete, SOUVEYET, Carine**.** *A proposal for improving the quality of scenario collections.* Proceedings of the Fourth International Workshop on Requirements Engineering: Foundations of Software Quality, REFSQ´98, Pisa, Italy Presses Universitaires de Namur (eds, E. Dubois, A. L. Opdhal, K. Pohl), pp. 29-42, 1998.

[4] ÁNTON, Annie. *Goal Identification and Refinament in the specification of Information Systems.* Ph.D. Thesis. Georgia Institute of Technology, June 1997.

[5] ÁNTON, Annie. *Goal Based Requirements Analysis*. Proceedings of the 2nd International Conference on Requirements Engineering' 96. Pp. 136-144, 1996.

[6] ÁNTON, Annie, E. Liang, and, and R.A. Rodenstein. *A Web-Based Requirements Analysis Tool.* In Fifth Workshops on enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'96), pages 238-243, Stanford, California, USA, June 1996.

[7] BOMAN, Tomas, SIGERUD, Katarina. *Tool for Requirements Elicitation and Documentation.* Available at http://www..cs.umu.se/~record/T-red/master.html

[8] BREITMAN, Karin, Leite J.C.S.P. *Evolução de cenários*, Tese de doutorado submetida na PUC-Rio em maio de 2000.

[9] CALDAS, J. *Uma ferramenta baseada em Cenários para Elicitação e Modelagem de requisitos*. Dissertação de Mestrado, ICMC – USP, Maio 1998.

[10] CALDAS, João and MASIERO, Paulo. *Obtenção de Modelos de Análise-OO a partir de um Modelo de Requisitos Baseado em Cenários.* In Workshop Iberoamericano de Engenharia de requisitos e Ambientes de Software. Alajuela, Costa Rica.pp 24-35.1999.

[11] CANTÚ, Marco. *Dominando o Delphi*. São Paulo: Makron Books, 1997.

[12] CARVALHO, M. C. F., ABDELOUAHAB, Zair. *On combining use cases and scenarios in Requirements Engineering*. XIII International Conference on Software and Systems Engineering and their Applications (ICSSE 2000), Paris, França, 2000.

[13] CARVALHO, M. C. F., ABDELOUAHAB, Zair. *Um Método para elicitação e Modelagem de Requisitos.* IV Workshop em Engenharia de Requisitos (WER´**2001)**, Buenos Aires, Argentina, 2001.

[14] COCKBURN, A. *Structuring use cases with goals*. Technical report. Human and Technology, 7691 Dell Rd, Salt Lake City, UT 84121, HaT. TR.95.1.Available at http://members.aol.com/acockburn/papers/usecases.htm, 1995.

[15] CHUNG, Lawrence, NIXON, Brian and YU, Eric. *Using Non-Functional Requirements to Systematically Support Change.* In Proceedings 2nd International Symposium on Requirements Engineering (RE´95), pages 132-139, York, UK, March 1995.

[16] CYSNEIROS, L. M., LEITE, J.C.S.P. *Requisitos não funcionais: Da Elicitação ao Modelo Conceitual*. Tese de Doutorado submetida na PUC-Rio em fevereiro de 2001.

[17] DANO, B., BRIAND, H., BARBIER, F. *A use case driven requirements engineering process.* Third IEEE International Symposium on Requirements Engineering ' 97, Antapolis, Maryland, IEEE Computer Society Press, 1997.

[18] DARDENNE, A., Lamsweerde, Van A.. and S. Fickas. *Goal-directed Requirements Acquisition***.** Science of Computer Programming, 20 (1-2): 3-50, April 1993.

[19] DARIMONT, R., DELOR, E., MASSONET, P. and LAMSWEERDE, A. Van , *GRAIL/KAOS : An environment for Goal-Driven Requirements Engineering*, Proc ICSE'98: 20th International Conference on software Engineering, Kyoto, Japan, April 1998, Vol. 2, 58-62.

[20] DESHARNAIS, J. , Frappier M, Khedri R., Mili A. *Integration of Sequential Scenarios.* IEEE Transactions Software Engineering 1998; 24(09): 695 – 708.

[21] ENGO, Frank. *Como Programar em Delphi 3*. Tradução e Revisão Técnica: Álvaro Antunes. São Paulo: Makron Books, 1996.

[22] GLINZ, Martin. *An Integrated Formal Model of Scenarios based on Statecharts*. In : Proceedings of the 5th ESEC´95, Lecture Notes on Computer Science 989. Springer, Berlin, pp 254-271, 1995.

[23] GOGUEN, Joseph and LINDE, Charlotte. Techniques for requirements elicitation. In Fickas and Finkelstein, editors. Requirements Engineering'93, 152-164, IEEE, 1993.

[24] GREEN, S. *Goal-Driven Approaches to Requirements Engineering*. Technical Report DOC TR 93-42 1994, Imperial College of Science, Technology and Medicine, Department of Computing Technical Report, London, UK, 1994.

[25] HSIA, P, SAMUEL, J., GAO, J. et al. *Formal approach to scenario analysis*. IEEE Software, pp. 33-41, 1994.

[26] HURLBUT, Russell. *A survey of Approaches for describing and formalizing use cases.* . Technical Report. XPT-TR-97-03. Available at http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html

[27] JACOBSON, CHRISTERSON, M, JOHSSON. *Object Oriented Software Engineering: a Use Case Driven Approach,* Addison-Wesley Publishing Company, 1992.

[28] KULAK, Daryl and GUINEY, Eamonn. *Use Cases – Requirements in Context*. ACM Press. New York. 2000.

[29] LAMSWEERDE, A. Van., *Divergent Views in Goal-Driven Requirements Engineering*, Proc Viewpoints'96 – ACM SIGSOFT Workshop Viewpoints in Software Development, Oct. 1996.

[30] LAMSWEERDE, A. Van., *Goal-Oriented Requirements Engineering: A Guided Tour*, Invited Minitutorial, Proc. RE'01 – International Joint Conference on Requirements Engineering, Toronto, IEEE, August 2001, pp. 249-263.

[31] LAMSWEERDE, A. Van and LETIER, E., *Integrating Obstacles in Goal-Driven Requirements Engineering,* Proc ICSE'98: 20th International Conference on software Engineering, Kyoto, Japan, April 1998, Vol. 1, 53-63.

[32] LAMSWEERDE, A. Van, DARIMONT, R., LETIER, E., *Managing Conflicts in Goal-Driven Requirements Engineering*, Special Issue on Inconsistency Management in Software Development, Vol. 24 Nº 11,November 1998, 908-926.

[33] LAMSWEERDE, A. Van and LETIER, E., *Handling Obstacles in Goal-Driven Requirements Engineering*, IEEE Trans. on Software Engineering, Special Issue on Exception Handling, Vol. 26 Nº 10, October 2000, pp.978-1005.

[34] LEITE, Júlio César Sampaio do Prado et al. *Enhancing a Requirements Baseline with scenarios.* In Third IEEE International Symposium on Requirement Engineering RE' 97, antapolis, Maryland, IEEE Computer Society Press, pp. 44-53, 1997.

[35] LEITE, Júlio César Sampaio do Prado et al. *A Scenario Construction Process. In on Requirement Engineering* , Antapolis, Maryland, IEEE Computer Society Press, pp. 38-61. 2000.

[36] MARTIN, James *Engenharia da Informação: Introdução*. Rio de Janeiro.Campus. 1991, 191 p.

[37] MISAKA, Antonio Paulo. *Um Modelo de requisitos para o Método FUSION*. Dissertação de Mestrado, USP – São Carlos. 1996.

[38] MISAKA, Paulo, MASIERO, Paulo. *Proposta de um Modelo de Requisitos para o Método Fusion* . In Workshop iberoamericano de engenharia de requisitos e ambientes de software. Torres, RS. Pp 1-13. 1998.

[39] POTTS, C.,Takahashi, K., Anton, A . *Inquiry-Based Requirements Analysis*. IEEE Software, 11(2): 21-32, March 1994.

[40] PRESSMAN, Roger. *Engenharia de Software*. 3ª ed. Editora McGrawHill, 1995.

[41] RALYTÉ, Jolita, ROLLAND, Collete, PLIHON, Véronique. *Method enhancement with scenario based techniques.* To appear in: Proceedings of CAISE´99, 11th Conference on Advanced Information Systems Engineering Heidelberg, Germany June 14-18, 1999.

[42] REGNELL, B., KIMBLER, Kristofer, WESSLÉN, A. *Improving the use case driven approach to Requirements Engineering.* Proceedings of Second International Symposium on Requirements Engineering,, York, UK March 1995.

[43] REGNELL. B., ANDERSSON, M., BERGSTRAND, JOHN. *A Hierarchical Use Case Model with Graphical representation.* IEEE International Symposium and Workshop on Engineering of Computer-Based Systems, Germany, March 1996.

[44] REGNELL, Bjorn. *Requirements Engineering with use cases – a Basis for Software Development.* Ph.D. Thesis. Lund Institute of Technology, 1999.

[45] ROLLAND, C., SOUVEYET, C. ACHOUR, C. Ben. *Guiding goal modelling using scenarios.* IEEE Transactions on Software Engineering, special issue on Scenario Management, 1998.

[46] ROLLAND, C., Grosz, G and Kla, R.. *Experience with Goal-Scenario Coupling in Requirements Engineering.* CREWS Report Series 98- 32. Fourth IEEE International Symposium on Requirements Engineering (RE´97). University of Limerick, Ireland. 7-11 June 1999.

[47] SOMMERVILLE, Ian and KOTONYA, Gerald. *Requirements Engineering: Processes and Techniques.* John Wiley & Sons, 1997.

[48] SOMMERVILLE, Ian and SAWYER, Peter. *Requirements Engineering: a good practice guide*. John Wiley & Sons, 1998.

[49] SUTCLIFFE, A G. and N.A.M. Maiden. *Bridging the Requirements Gap: Policies, Goals and Domains*. In Proceedings of the Seventh International Workshop on Software Specification and Design, Redondo Beach, Califórnia, December 1993.

[50] TAWBI, M., Souveyet, C. and Rolland, C. *L`Ecritoire a tool to support a goal-scenario based approach to requirements engineering.* CREWS Report Series. Submitted to Information and Software Technology Journal (ed. M. Shepperd) Elsevier Science Publishers. 1998.