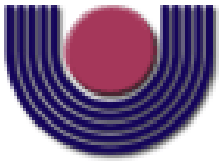


UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Integrando i* ao Processo de Medição de Software

Allan Roger Bello

CASCAVEL

2011

ALLAN ROGER BELLO

INTEGRANDO I* AO PROCESSO DE MEDIÇÃO DE SOFTWARE

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientador: Prof. Dr. Victor Francisco Araya Santander

CASCAVEL

2011

ALLAN ROGER BELLO

INTEGRANDO I* AO PROCESSO DE MEDIÇÃO DE SOFTWARE

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Ciência da Computação*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Victor Francisco Araya Santander
(Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Carlos José Maria Olguin
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Sandra Mara Stocker Lago
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 27 de Outubro de 2011.

DEDICATÓRIA

Aos meus pais, que sempre me incentivaram, me apoiaram, e me deram forças para continuar lutando sempre. Também a minha namorada que teve toda a paciência do mundo e me apoiou sempre. Muito obrigado!

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ES	Engenharia de Software
APF	Análise de Pontos de Função
DFD	Diagrama de fluxo de Dados
DD	Dicionário de Dados
DER	Diagrama Entidade Relacionamento
SD	Dependências Estratégicas
SR	Razões Estratégicas
PF	Pontos por Função
IFPUG	*International Function Point Users Group*
ALI	Arquivo Lógico Interno
AIE	Arquivo de Interface Externa
ETD	Elemento do Tipo Dado
ETR	Elemento do Tipo Registro
AR	Arquivo Referenciado
VFA	Valor do Fator de Ajuste
GIT	Grau de Influência Total
PFNA	Pontos de Função Não Ajustados
PFA	Pontos por Função Ajustados
DC	Diagrama de Classe
RS	Requisitos do Sistema

Sumário

Resumo

Atualmente, um dos maiores problemas em discussão pelos profissionais da área de computação e principalmente desafio para profissionais da Engenharia de Software (ES) tem sido conseguir apresentar métodos de construção de aplicações cada vez mais rápidas, curtos e de maior confiabilidade, além disto, tentar incrementar novos conceitos nos métodos já existentes, com o intuito de melhorar o processo. Neste contexto é importante iniciar o desenvolvimento de software entendendo o ambiente organizacional no qual o mesmo atuará e para este fim pode-se utilizar técnicas de modelagem organizacional. Neste sentido destaca-se o *framework i**, pela capacidade de modelar as capacidades estratégicas importantes para avaliar os relacionamentos no ambiente organizacional.

As técnicas de medição de software também são técnicas interessantes a serem utilizadas no planejamento e gerenciamento de aplicações, pois quando se consegue medir um sistema antes de seu desenvolvimento isso possibilita que seu gerente ou equipe responsável pelo gerenciamento consiga com o tempo e experiência prever uma relação de custo e tempo que será necessário para a construção do sistema.

É evidente que ambas as técnicas contribuem para o desenvolvimento de sistemas, porém o problema hoje é que as técnicas de medição utilizadas requerem muita documentação e um bom grau de análise e/ou experiência do profissional, assim exigindo tempo, e isto para o mercado é crucial, o que prejudica na busca por um produto de qualidade com baixo custo e tempo. Dessa forma este trabalho propõe a integração do *framework i** ao processo de medição de *software*, uma vez que para casos de estimativas de software o mesmo pode se tornar um recurso de grande valia, pois pode beneficiar desde a redução da documentação necessária, até ao auxilio da análise e tomadas de decisões durante todo o processo.

Palavras-chave: Engenharia de Software, Modelagem Organizacional, Desenvolvimento de Software, Gerência de Projeto, Medição de Software.

Capítulo 1

Introdução

Este Capítulo tem como objetivo a apresentação do trabalho, indicando as motivações para o seu desenvolvimento, bem como a sua estruturação. Inicialmente, na Seção 1.1, a proposta é contextualizada no escopo da medição de software e *framework* i*, destacando a importância do processo da medição e uso do i* na obtenção de produtos de software de qualidade. Na Seção 1.2, são descritas as principais motivações que levaram a execução desta pesquisa. Na Seção 1.3, a proposta em si é explanada, bem como seus principais objetivos. Já na Seção 1.4, são relatadas as contribuições esperadas com o desenvolvimento do presente trabalho. Por fim, na Seção 1.5 a organização desse documento é apresentada.

1.1 Contexto

Com o passar dos anos o processo de globalização levou o computador a ser um bem de fácil acesso no mundo inteiro. Esse processo pode ser resumido em avanços tecnológicos e científicos que geraram um novo paradigma com um novo nicho de mercado, que é o desenvolvimento de sistemas a vários setores. Para tentar ocupar esse espaço aberto no mercado, muitos empreendedores se lançaram ao mesmo sem a devida preocupação com os processos de desenvolvimento. Por outro lado, a tendência desse mercado é de absorver apenas produtos de qualidade, e com produtos já dispostos e consolidados no mercado a concorrência se tornará muito mais difícil.

Logo no ano de 1994 o departamento de defesa dos Estados Unidos conduziu um estudo onde indicou que 75% de todos os grandes sistemas intensivos de software adaptados falham e sua causa principal é o fraco gerenciamento, e não o desempenho técnico.

Outro estudo realizado pela (Standish Group, 2001) chamado de relatório de “CHAOS” trás dois estudos realizados no norte dos Estados Unidos com cerca de 40000 projetos de aplicação de TI (Tecnologia da Informação).

O primeiro estudo da Standish Group em 1994 mostra que empresas dos Estados Unidos gastaram cerca de 81 milhões de dólares em projetos que foram cancelados onde 31% dos

projetos de software estudados foram cancelados antes mesmo de estarem concluídos, 53% dos projetos excederam mais do que 50% a sua estimativa de custo, e somente 9% dos projetos em grandes empresas foram entregues no tempo e orçamento previstos.

No segundo, estudo agora referente ao ano de 2001, foi encontrado um quadro de evolução se comparado com o anterior. O percentual de projetos entregues dentro do tempo, custo e especificações previstas subiu para 28%, projetos cancelados ou falidos antes de serem completados caiu para 23%, e a extrapolação de orçamento caiu para 45% e a de prazo caiu para 63%.

As principais causas de falhas nos projetos estão associadas a dificuldades com os seguintes temas: apoio da alta gerência, envolvimento do usuário, experiência do gerente do projeto, e definição clara das regras de negócio e escopo do projeto (Standish Group, 2001).

Essa evolução vista nos estudos de 1994 e 2001 demonstraram como houve uma evolução no desenvolvimento de soluções de software adotada, e como a gerência de projetos é grande responsável para a qualidade da aplicação em desenvolvimento. Demonstrou também como a corrida pela ocupação do espaço aberto para o mercado de desenvolvimento de sistemas foi feito sem qualquer planejamento.

Neste contexto, segundo o Guia PMBOK (2008, p12), ”o gerenciamento de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas as atividades do projeto a fim de atender aos seus requisitos. Para (Vargas, 2000) “o gerenciamento de projetos pode ser aplicado a qualquer situação onde exista um empreendimento que foge ao que é fixo e rotineiro na empresa”.

A gerência é dividida em algumas subáreas: integração, gerenciamento de escopo, gerenciamento de prazos, gerenciamento de custo, gerenciamento de qualidade, gerenciamento de recursos humanos, gerenciamento de comunicação, gerenciamento de riscos e gerenciamento de contratação. Neste trabalho a contribuição esperada é permitir uma medição viável de qualidade, para que as empresas possam com o tempo, experiência e históricos serem capazes de obter maior sucesso no gerenciamento de escopo, custos e prazos. A seguir descrevemos brevemente cada uma:

a) **Gerenciamento de Escopo:** têm o objetivo de definir e controlar o que deve e o que não deve estar incluído no projeto. Consiste da iniciação, planejamento, definição, verificação e controle de mudanças do escopo;

b) Gerenciamento de prazos: têm o objetivo de garantir o término do projeto no tempo certo. Consiste da definição, ordenação e estimativa de duração das atividades, e da elaboração e controle de prazo;

c) Gerenciamento de custo: têm o objetivo de garantir que o projeto seja executado dentro do orçamento aprovado, ou seja, planejamento de recursos, estimativa, orçamento e controle de custos.

Contudo é preciso levar em conta que realizar um planejamento e gerenciar um projeto de sistemas de software não é algo trivial, começando pelos requisitos que são itens que mudam a medida que o projeto está em andamento. O cliente geralmente, ao decorrer do processo, se atenta a alguns detalhes ou funcionalidades que esqueceu ou se deu conta de que precisará, e claro isso irá interferir no planejamento realizado inicialmente.

Também entra o fator tempo, pois algumas metodologias tradicionais de desenvolvimento de software impõem a necessidade de gerar certo número de documentação que muitas vezes acaba sendo inviável a realidade da maioria das empresas no mercado, essa inviabilidade por fim acaba prejudicando qualquer resultado de planejamento levantado e conseqüentemente correndo um maior risco de erros nas estimativas levantadas inicialmente.

(Braga, 1996, p.387) afirma que "não se pode gerenciar o que não se pode medir". E a medição do software é uma forma de se conseguir estimar prazos, custos e analisar a produtividade do desenvolvimento do software.

A medição de software sem dúvidas é uma alternativa interessante para contribuir com o gerenciamento de um projeto. Atualmente existe uma técnica muito utilizada que é a APF (Pontos de Função), (Tavares *et al*, 2005).

O grande problema dessa técnica de medição é a quantidade de documentação que deve ser levantada entre diagramas, requisitos e outros processos que são "demorados", e onde entra o fator tempo, mencionado anteriormente. Também existe a forte necessidade de conhecimento sobre as regras de negócio do "ambiente" em que o sistema atuará, pois durante o processo de medição este conhecimento é importante para a tomada de decisões.

Empresas também vêm desconsiderando na maioria das vezes o contexto organizacional e elementos intencionais existentes que devem ser considerados em um processo de medição. Assim utilizar modelos organizacionais pode auxiliar consideravelmente o nível de precisão de estimativas de software, e conseqüentemente tendem a melhorar a qualidade final do sistema.

Nesse contexto torna-se necessário impulsionar as pesquisas na Engenharia de Software (ES) em busca de formas e metodologias para se buscar uma medição de software juntamente com o *framework* i*, onde se consiga diminuir o esforço gasto na etapa de levantamento da documentação necessária para se aplicar as métricas de medição, e fazer com que o usuário/cliente participe mais dessa etapa.

Além disso, o uso desse modelo organizacional não traria apenas esse benefício, mas também permitiria entender melhor o contexto organizacional no qual o sistema estará inserido, além de auxiliar na avaliação de impactos de mudanças que podem afetar estimativas já realizadas. Na próxima seção, apresentamos as principais motivações para a realização deste trabalho.

1.2 Motivações

O desenvolvimento de software ainda continua sendo um desafio para a área da computação, assim ainda continua sendo motivo de pesquisa por profissionais da área de ES que procuram com o passar dos anos e com o surgimento de novas pesquisas encontrar algumas soluções para tentar circundar "velhos" problemas que são enfrentados pela comunidade acadêmica e empresarial desde que os primeiros sistemas começaram a serem produzidos.

A demanda por sistemas ainda tende a crescer e muito, e com isso quem irá ganhar o mercado é quem conseguir produzir um produto de qualidade. Algumas empresas que já conseguem enxergar essa necessidade começam a investir mais em processos de engenharia para buscar o melhor produto custo benefício possível, e assim conquistar uma qualidade aceitável pelos clientes (Vazquez *et al*, 2010), se consolidando no setor.

Nesse cenário o desenvolvimento de novas técnicas de suporte a ES é de fundamental importância para acompanhar essa evolução do mercado, e conseqüentemente ajudar na busca dessa qualidade desejada.

Neste contexto se vê a necessidade de realizar pesquisas na área de medição de software, as quais consideram fatores críticos: a necessidade de viabilizar uma medição com menor número de documentos para aplicar a técnica de medição APF (Pontos de Função) e o fato de basear a medição em metas organizacionais estratégicas da organização que requer o software, uma vez que a medição é dependente do entendimento da organização, porém não ampara seu entendimento durante o processo. Também cabe destacar que a maioria das

pequenas e médias empresas não dispõe de recursos e tempo para aplicar uma técnica de medição com toda a documentação exigida.

Neste contexto, e partindo da importância que a medição de sistemas de software tem para as empresas, de fato é importante encontrar caminhos onde se possam integrar novos recursos para melhorar este processo, e talvez deixá-lo menos "custoso", dessa forma tornando viável este tipo de prática nas mesmas. Dessa forma contribuiríamos para um planejamento mais completo e fácil para o desenvolvimento de sistemas de software.

1.3 Proposta

Neste trabalho, apresenta-se um estudo para buscar "medidas de tamanho" de software utilizando o *framework i** com o intuito de agilizar o processo de medição na APF, visando um caminho com maior facilidade de análise e tomadas de decisões necessárias durante o processo, também permitindo usufruir do *framework i** para o decorrer das etapas até o desenvolvimento do sistema de *software*.

Construir sistemas em tempo hábil para serem úteis aos negócios e com qualidade adequada é o desafio que as organizações que desenvolvem software estão enfrentando (Tavares *et al*, 2005) e é um benefício que pode ser alcançado quando se agrega conhecimento organizacional, agilidade na medição de software, conhecimento do tamanho funcional (medição de software) e qualidade na gerência de projetos (vantagens conquistadas quando se consegue medir o sistema).

O primeiro objetivo deste trabalho está focado em viabilizar uma medição de software de forma mais rápida e viável, partindo primeiramente do *framework i**. Empresas também vêm encontrando dificuldades na maioria das vezes sobre contexto organizacional e elementos intencionais existentes que devem ser considerados em um processo de medição. Assim utilizar modelos organizacionais integrados dentro do processo medição de *software* pode melhorar consideravelmente o nível de precisão de estimativas e agilizar o processo.

1.4 Contribuições Esperadas

Espera-se com este trabalho contribuir com a pesquisa na área de ES através de um resultado positivo permitindo uma medição de *software* de qualidade e em tempo hábil.

Partindo de uma avaliação sobre os resultados de medição alcançados pretende-se contribuir para o gerenciamento no desenvolvimento de sistemas de *software*, desde sua fase inicial ate à medida que o projeto é concretizado. Lembrando que a medição é uma técnica muito poderosa e serve como auxílio principalmente para as gerências de custo, prazos e escopo.

De maneira complementar, deseja-se que o estudo desenvolvido permita ao utilizar o *framework* i*, adquirir durante andamento do processo de medição uma maturidade organizacional, retirando todas as informações e vantagens possíveis dele para melhorar o processo de desenvolvimento juntamente com a medição realizada. Assim espera-se contribuir para o processo de produção de softwares.

1.5 Estrutura do Trabalho

Este trabalho divide-se em seis Capítulos, estruturados da maneira apresentada a seguir.

No Capítulo 2 são abordados os principais conceitos do *framework* i*, bem como os principais benefícios ao ser adotado, características que a diferem de outros modelos organizacionais e figuras exemplificando a utilização e interpretação do diagramas que compõem este *framework*.

O Capítulo 3 apresenta uma visão geral da importância que tem a medição de software no contexto empresarial, explica também detalhadamente a técnica de APF, com exemplo de uso para um determinado contexto.

O Capítulo 4 apresenta os passos e diretrizes a serem seguidos para integrar o *framework i** ao processo de medição de software.

O Capítulo 5 traz um estudo de caso onde é aplicada a medição de software integrada com o *framework i**, baseada nos passos e diretrizes estudados no Capítulo 4.

Por fim o Capítulo 6 apresenta as considerações finais deste trabalho.

Capítulo 2

Modelagem Organizacional: Framework i*

As atividades de análise são responsáveis por 50% a 60% de todos os erros do software. Com uma atenção maior no processo de análise é possível reduzir os custos de desenvolvimento e manutenção. Muitas técnicas tradicionais utilizadas hoje em dia no desenvolvimento de software tratam aspectos relacionados à funcionalidade do sistema, entradas que deverão ser transformadas, saídas que deverão ser produzidas, porém sem considerar aspectos mais amplos como: objetivos da organização, regras de negócio, restrições, aspectos não funcionais relacionados à qualidade, confiabilidade e usabilidade (Pádua e Cazarini). E é partindo desse contexto que será visto a importância de se adotar um modelo organizacional.

Diferentes tipos de modelos são utilizados em diversos campos científicos, refletindo-se sobre o assunto em questão, e os tipos de entendimento que se pretende em cada campo. No passado técnicas de modelagem conceituais de software e a engenharia de sistemas de informação, concentraram-se em descrever e analisar comportamentos e as "suas" estruturas, que são implementáveis em software. Contudo nesse contexto os sistemas de software evoluíram muito e com isso se tornaram cada vez mais complexos e densamente entrelaçados com o ambiente humano e social (Yu, 1995).

Na atualidade podemos perceber que as tecnologias de informação estão impactando a vida das pessoas de maneira mais atuante do que nunca, e que a partir dessa complexidade para se desenvolver sistemas de software, torna-se de fundamental importância modelar as organizações, não apenas internamente, mas também objetivando apontar estratégias alternativas que ajudem no entendimento da regra de negócio. Além do que, para se produzir um sistema de software de qualidade se torna imprescindível compreender seu ambiente organizacional no qual o software será implantado e estará em uso (Chichinelli, 2002) e (Santander, 2002).

Segundo (Alencar, 1999), o modelo organizacional é uma representação da estrutura, atividades, processos, informações, recursos, pessoal, comportamento, objetivos e restrições

das empresas comerciais, governamentais ou de outra natureza, a fim de ajudar a compreender as complexas interações entre organizações e pessoas.

As organizações definem alguns objetivos e metas que terão de ser satisfeitas pelo sistema de software. Isto implica em dizer que o grau de satisfação dos objetivos e estratégias organizacionais contribui como um grande fator para o sucesso de sistemas de software (Santander, 2002). Além de contribuir para um maior amadurecimento por parte da equipe responsável pelo gerenciamento e desenvolvimento, facilitando assim o trabalho durante o ciclo de construção do sistema.

Neste contexto podemos perceber que a modelagem organizacional realmente traz grande benefício no desenvolvimento de sistemas de software, e adotar uma modelagem para se conhecer a organização de uma empresa é o primeiro passo para o êxito de um bom planejamento de projeto.

O presente Capítulo fornece uma visão geral a respeito da importância e as vantagens da modelagem organizacional com ênfase ao *framework i**. Esta ênfase é apresentada na Seção 2.1. A Seção 2.2 é referente às considerações finais.

2.1 Framework I*

O i* é um *framework* de modelagem organizacional desenvolvido para modelagem e análise. É usado para representar relacionamentos entre atores estratégicos em uma rede social. Isso é feito sob uma visão estratégica e intencional de processos que envolvem vários participantes, ou seja, uma modelagem organizacional (YU, 1995).

O *framework* i* nasceu com o objetivo de introduzir alguns aspectos da modelação social e de raciocínio em métodos de engenharia de sistemas de informação, especialmente a nível de requisitos. Ao contrário dos tradicionais métodos de análise de sistemas que se esforçam para abstrair os aspectos humanos de sistemas, i* reconhece a primazia dos atores sociais. Atores são vistos como sendo intencionais, ou seja, eles têm metas, crenças, habilidades e compromissos (YU, 1995).

A análise de sistemas no seu início utilizava técnicas como Diagrama de fluxo de Dados (DFD), Dicionário de Dados (DD) e Diagramas de Entidade e Relacionamento (DER). O problema desses e de outros modelos é que os mesmos descrevem apenas fluxos de dados, entidades e estados do sistemas, não expressando as razões envolvidas no processo, por exemplo: o porquê de executar uma determinada ação em vez de outra.

Visando sanar algumas destas deficiências expressas por técnicas cujo foco está na modelagem funcional, no *framework* i*, há uma concentração em propriedades intencionais e relacionamentos ao invés de comportamento real. Ao não descrever o comportamento diretamente, uma descrição intencional oferece uma maneira de caracterizar os atores respeitando a premissa de autonomia. Modelagens convencionais de sistemas oferecem apenas ontologias estática e dinâmica levando a uma visão empobrecida e mecanicista do mundo, já a modelagem intencional fornece uma rica expressividade que é muito mais apropriada para a concepção social do mundo (Yu, 1995).

Para superar as limitações dessa visão mecanicista, a partir do framework i* houve uma mudança de nossa atenção para longe do foco habitual nas atividades e fluxos de informação. Em vez disso, nós perguntamos: o que cada ator quer? Como é que conseguem o que querem? E quem é que eles dependem para alcançar o que eles querem?. Seguindo essa ideia é possível: melhorar a compreensão sobre os relacionamentos da organização, entender razões envolvidas nos processos de decisões e reconhecer motivações, intenções e raciocínios sobre as características de um processo (Santander, 2002) e (Chichinelli, 2002).

Nesse contexto, colocando conceitos sociais selecionados para o núcleo da atividade diária dos análistas de sistemas, ou seja, através da adoção de uma ontologia social para os principais modelos de construções (Yu, 1995), podemos concluir que a análise social não é só um complemento para a análise técnica, mas sim a base para a condução do desenvolvimento do sistema inteiro, onde essa compreensão social é a chave para o sucesso do processo de desenvolvimento de um sistema.

Para realizar a análise organizacional, os engenheiros de requisitos modelam todos que tenham alguma relação no sistema como atores e suas intenções como metas. Os atores são entidades que possuem metas e podem depender de outros atores para conseguir alcançá-las (Yu, 1995). O conjunto de dependências criadas pelos atores forma uma rede social que representa o ambiente do sistema e o sistema em si (Santos, 2008). O i* é formado por dois modelos: o Modelo de Dependência Estratégica (SD) e o Modelo de Razão Estratégica (SR), os quais são descritos a seguir.

2.1.1 Modelo SD

O modelo SD é composto por nós e ligações. Os nós representam os atores e cada ligação representa uma dependência entre atores. No ambiente organizacional o ator é compreendido como sendo uma entidade que executa ações para obter objetivos (Chichinelli, 2002), (Santander, 2002) e (Santos. E.B, 2008).

O modelo SD tem como meta demonstrar as motivações e os desejos dos atores que fazem parte da organização além de apresentar os relacionamentos dos atores no contexto. A partir de um modelo SD pode-se verificar a viabilidade ou não das dependências e questionar a existência de novos relacionamentos entre atores (Santos, 2008).

No contexto do modelo SD o ator é classificado de acordo com sua dependência, ou seja: o ator que depende de alguma maneira de outro ator é chamado de ***Depender***, e o ator que supri a necessidade do mesmo é conhecido como ***Denpendee.*** Por exemplo, a vida de um proprietário de carro é facilitada pelos mecânicos que estão sempre aptos e dispostos a reparar seu carro mesmo o proprietário não sendo capaz de reparar, porém ao mesmo tempo em que o proprietário depende de alguém o mesmo corre o risco de não receber o que espera, ou seja, o proprietário é dependente dos mecânicos e está vulnerável ao "sucesso" de seus "serviços".

O objeto ou elemento de dependência entre *Depender* e *Dependee* é denominado de ***Dependum*.** Essas dependências são portanto estratégicas para os atores envolvidos, pois podem ser benéficas ou prejudiciais a eles (Santander, 2002) e (Chichinelli, 2002). Vejamos a figura 2.1 na qual está ilustrado as dependências e a seguir a definição de ambas .

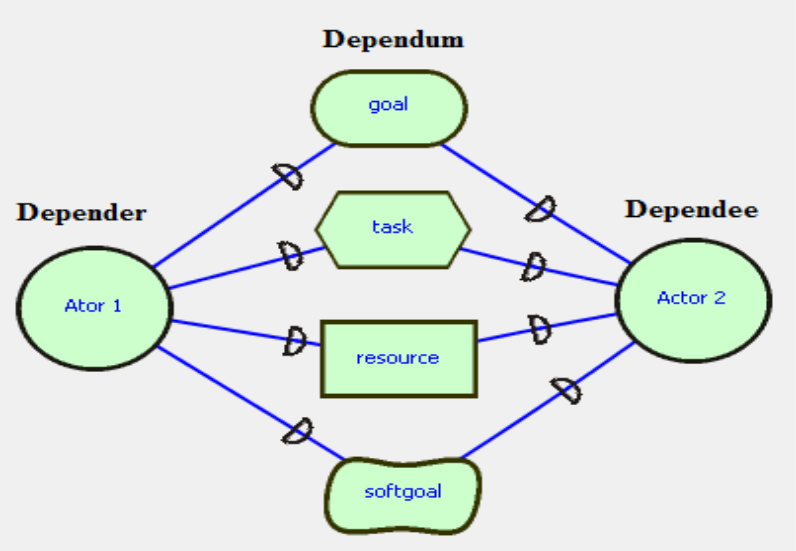


Figura 2.1: Dependências entre atores

No modelo SD podemos distinguir dependências em quatros tipos:

a) Dependência de **objetivo (*goal*):** Refere-se ao tipo de dependência onde o *Depender* depende do *Dependee* para modificar um estado do mundo, e ainda o *Dependee* tem a liberdade de tomar decisões necessárias para a obtenção do objetivo. Nesse contexto o *Depender* se torna vulnerável caso o *Dependee* não atinja o objetivo.
b) Dependência de **tarefa (*task*):** Refere-se ao tipo de dependência onde o *Depender* depende do *Dependee* para executar uma atividade, e cabe ao *Depender* mostrar o caminho como isto será feito, entretanto "o porque" da realização da tarefa não é fornecido ao *Dependee*. Nesse contexto o *Depender* se torna vulnerável caso o *Dependee* falhe na realização da tarefa por não ter seguido os passos indicados.
c) Dependência de **recurso (*resource*):** Refere-se ao tipo de dependência onde o *Depender* depende do *Dependee* referindo-se a disponibilidade de um recurso físico ou de informação. Neste contexto *Depender* se torna vulnerável caso não ocorra a disponibilização do recurso.
d) Dependência de ***softgoal:*** Refere-se ao tipo de dependência onde a avaliação de realização do objetivo é subjetiva, ou seja, seu real significado não é claramente conhecido. Neste contexto geralmente o *Dependum* é uma qualidade, tais como: rápido, confiável, seguro e assim por diante, portando são consideradas como metas onde não são bem definidas e passiveis de interpretação.

Visando aprofundar o entendimento sobre o *framework* i*, vejamos agora a ilustração de um exemplo sobre o ambiente organizacional de um hotel apresentado na Figura 2.2 Este exemplo foi retirado de (Girardello *et al*, 2009).

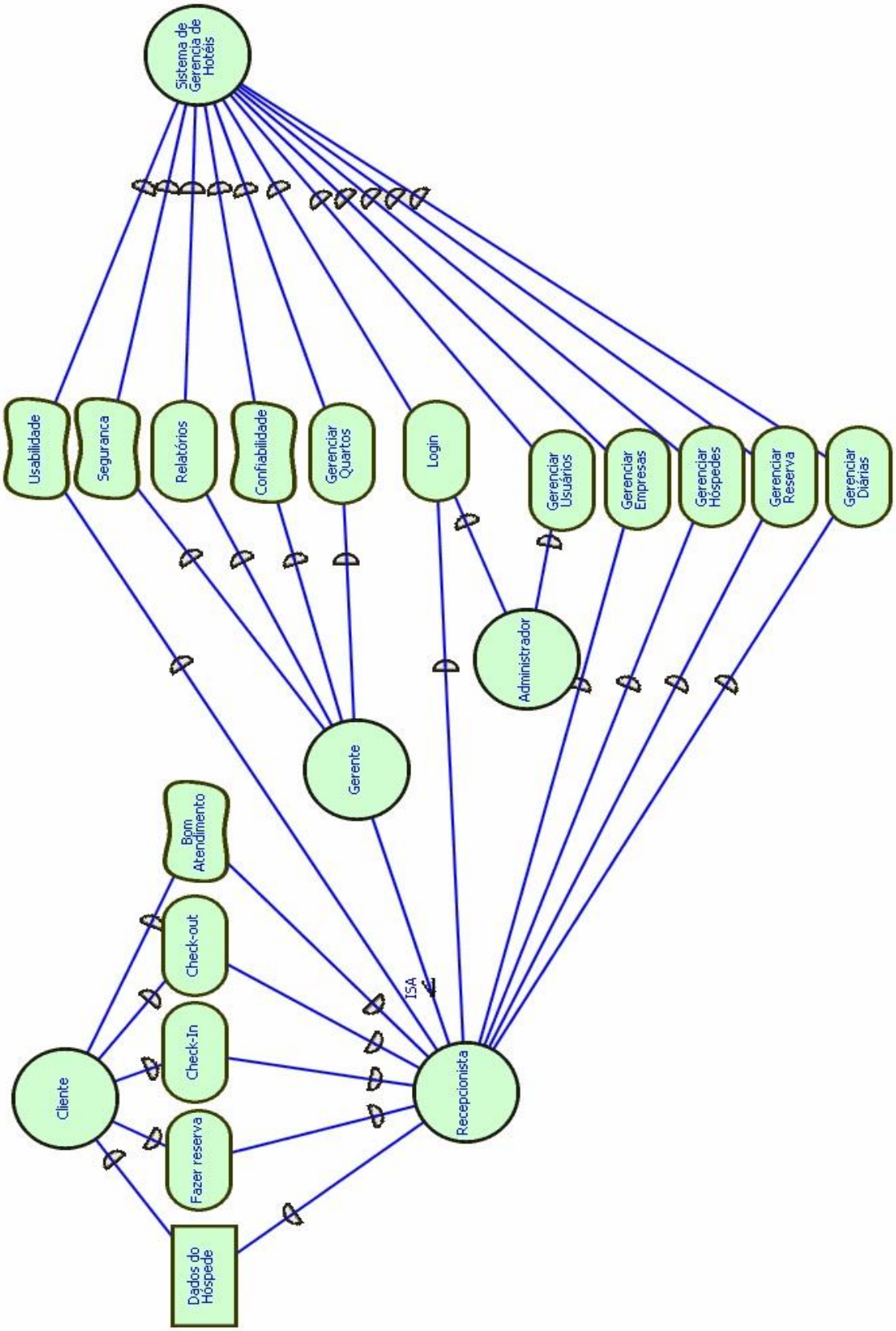


Figura 2.2: Modelo de Dependências Estratégicas (SD) Hotel Real

No exemplo apresentado na Figura 2.2, podemos notar, por exemplo, que na dependência de recurso **"Dados do Hóspede"** (*Dependum*) o ator "**Recepcionista"** (*Depender*) depende dos dados a serem fornecidos pelo ator **"Cliente"** (*Dependee*) para alcançar seu objetivo. No mesmo cenário entre os atores só que agora com o ator "**Cliente"** assumindo papel de *Depender* e o ator **"Recepcionista"** no papel de *Dependee* temos que o cliente é dependente do **"Recepcionista"** em relação aos objetivos: **"Fazer reserva", "*Check-In*"** e **"*Check-Out*"** e de um *softgoal* **"Bom Atendimento"** (onde todos são *Dependum*), Assim o "**Recepcionista"** é quem deve proporcionar condições para que o "**Cliente"** alcance seus objetivos, ou seja, o cliente está vulnerável ao recepcionista. Ainda analisando o ator "**Recepcionista"** notamos que ainda há dependências de objetivo: **"Gerenciar Empresas"**, **"Gerenciar Hóspedes", "Gerenciar Reservas"** e **"Gerenciar Diárias",** e dependências *softgoal: **"Usabilidade", "Segurança"*** deste ator em relação ao ator **"Sistema de Gerencia de Hotéis"** o qual quem tem o papel de satisfazer as mesmas.

No mesmo modelo só que no cenário envolvendo o ator **"Gerente"**, o mesmo é dependente de dependências objetivo: **"Relatórios"** e **"Gerenciar Quartos"** além de dependências do tipo *softgoal:* **"Usabilidade", "Segurança" e "Confiabilidade"**, todas estas sendo de responsabilidade do ator **"Sistema de Gerencia de Hotéis"**. Outro detalhe que ocorre com o ator **"Gerente"** é o surgimento da ligação *ISA*, o que significa que o ator gerente herda todas as dependências do ator recepcionista.

Já o ator **"Sistema de Gerencia de Hotéis",** em todos os casos deste modelo, assume o papel de *Dependee*, ou seja, o papel de fornecer meios para que os outros atores ligados a ele possam satisfazer suas metas. Assim já podemos notar que este ator tem grandes responsabilidades neste contexto organizacional.

O modelo SD como foi observado oferece uma compreensão aprofundada sobre os relacionamentos entre os atores, além de definir bem as dependências estratégicas no meio organizacional, assim possibilitando apontar as habilidades de cada ator e também as possibilidades de falha.

2.1.2 Modelo SR

Já no modelo SR é possível compreender e modelar de maneira mais detalhada as razões envolvidas nos processos. Este modelo possibilita um maior entendimento a respeito das

razões estratégicas de atores, e como os mesmos são expressos no ambiente organizacional. O SR também costuma ser utilizado na compreensão de como sistemas de software estão relacionados com as rotinas dos atores da modelagem para a geração de alternativas (Santander, 2002).

É importante para a construção do SR que se dê uma atenção maior para as razões que estão associadas a cada ator em relação aos relacionamentos de dependência com outros atores. Os *Dependee* praticamente têm como objetivo satisfazer os *dependums*, ou seja, partindo desde princípio podemos observar como os *Dependee* pretendem alcançar seus objetivos, assim decompondo as intenções e razões estratégicas como um todo (Santander, 2002) e (Yu, 1995).

O SR apresenta-se de nós e ligações permitindo expressar as razões envolvidas nos processos (Chichinelli, 2002). O SR é composto por quatro tipos de nós que se baseiam pelos diferentes tipos de *dependum* (objetivo, tarefa, recurso e *softgoal)*. O SR é também composto de ligações, conforme segue: **meio-fim (*means-end*)** e **ligação de decomposição de tarefa (*task decomposition*).**

A ligação **meio-fim** é responsável por representar um relacionamento entre um fim e um meio para atingir esse fim, onde este fim pode variar entre um objetivo, recurso, *softgoal*, ou uma tarefa e o meio geralmente se trata de uma tarefa a ser cumprida para que se chegue ao fim, podemos observar melhor na figura 2.3, retirada de (Chichinelli, 2002). Neste contexto a ligação meio-fim consegue proporcionar ao SR uma visão de processo orientado a objeto, ou seja, a ligação é apresentada de tal maneira que ao se fazer um questionamento do "como", um meio é procurado com o nó atual sendo seu fim desejado (Chichinelli, 2002), (Santander, 2002) e (Yu, 1995).

Já a ligação de **decomposição de tarefa** é responsável por expressar da melhor maneira possível como realizar uma determinada tarefa, e para isso a mesma pode ser decomposta em subconjuntos (figura 5): sub-objetivo, sub-tarefas e um recusoPara (recurso necessário para a realização da tarefa) , ou um *softgoalPara*, figura 2.4 (Chichinelli, 2002), ou seja, seguindo está abordagem de separar uma tarefa em "unidades menores" é possível obter uma representação das razões associadas com as tarefas de forma mais clara e detalhada.

Após estas definições vejamos agora na figura 2.5 um exemplo de modelagem SR para o mesmo sistema de hotel exibido anteriormente na figura 2.2 (modelo SD).

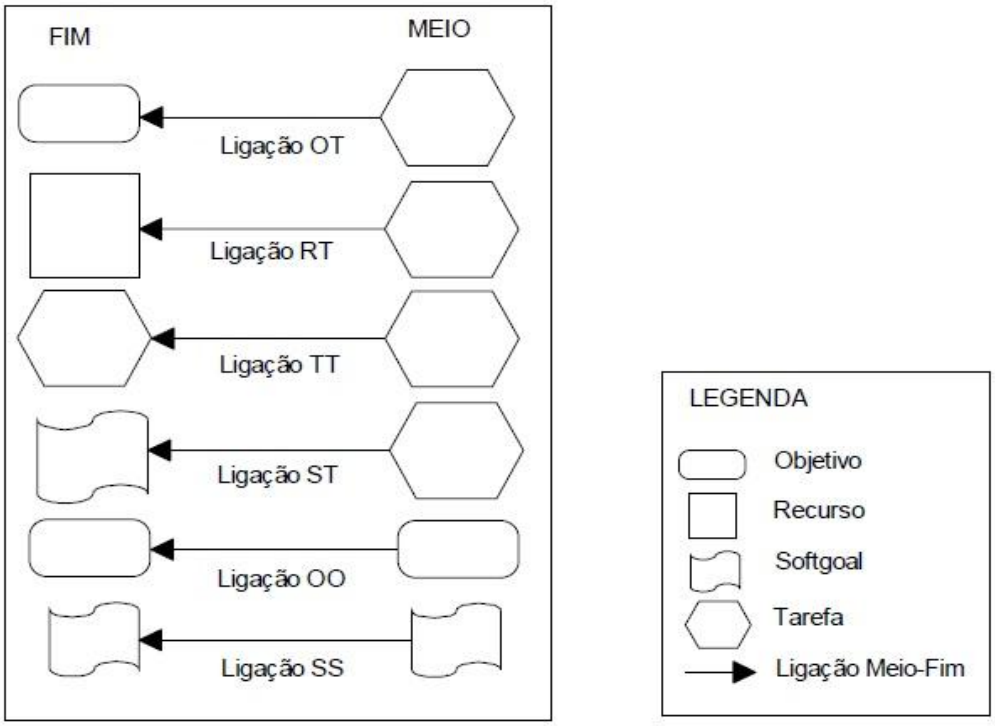


Figura 2.3: Ligação meio-fim

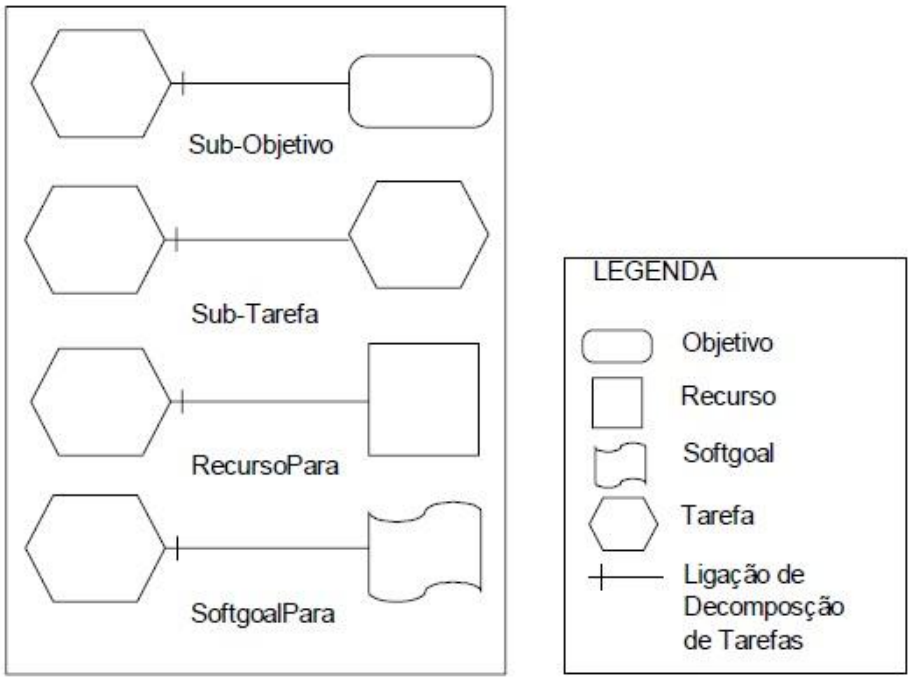


Figura 2.4: Ligação de decomposição de tarefa

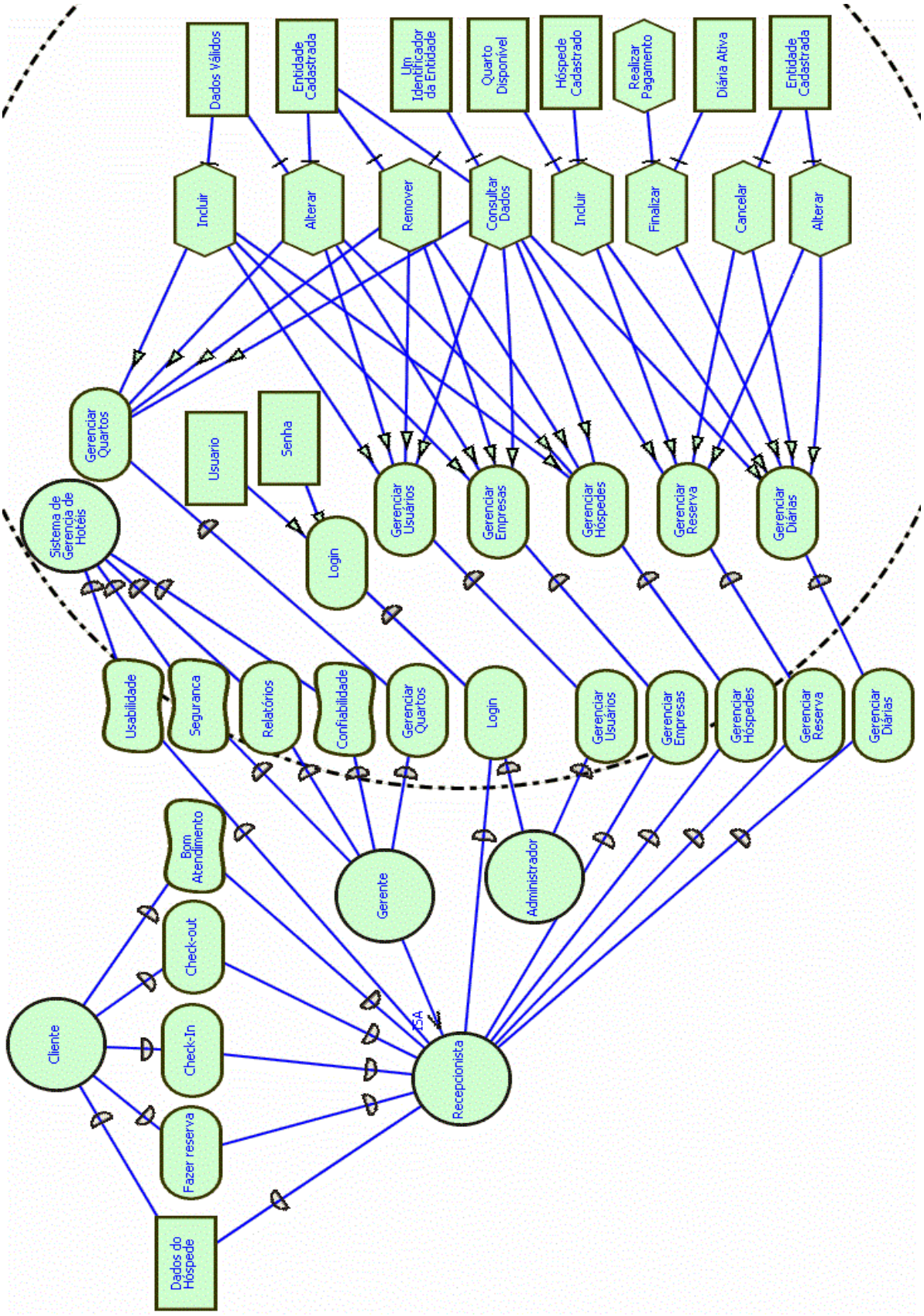


Figura 2.5**:** Modelo de Razões Estratégicas (SR) Hotel Real.

Podemos observar que no modelo SR (figura 2.5) construído para o hotel Real é feito um detalhamento maior do ator "**Sistema de Gerencias de Hotéis"** que no modelo SD, ou seja, este ator agora é subdividido em partes menores com o objetivo de descrever os relacionamentos intencionais internos através das ligações de decomposição de tarefas e meio-fim, dessa forma é possível explicitar as alternativas, motivos e razões dos atores. Neste contexto podemos observar, por exemplo, que a tarefa **"Consultar Dados"** é uma ligação de meio-fim onde o mesmo é "meio" para que se alcance seis "fins" que são na verdade objetivos: **"Gerenciar Quartos", "Gerenciar Usuários", "Gerenciar Empresas", "Gerenciar Hóspedes",** contudo esta tarefa é dependente de dois sub-recursos: **"Entidade Cadastrada" e "Um Identificador da Entidade"**, ou seja, existem vários objetivos que dependem dessa tarefa para que os mesmos sejam satisfeitos, entretanto para que essa tarefa seja satisfeita ela depende destes dois sub-recursos citados, para que assim seja capaz de cumprir seu papel no modelo. A mesma ideia é válida para as demais tarefas criadas no modelo SR: **"Incluir", "Alterar", "Remover", "Finalizar", "Cancelar"** e **"Alterar"**, seguindo a mesma lógica de decomposição de tarefas.

Assim o modelo SR é projetado permitindo modelar processos e razões associados a satisfação de dependências, explicitando assim as ações que o ator responsável por satisfazer essas dependências realizará.

2.2 Considerações Finais

Atualmente a realidade para o profissional da área de informática que entra no mercado de desenvolvimento de sistemas de computadores é a mesma para todos, ou seja, o fato é que esse profissional vem com uma formação técnica e científica específica para o domínio de desenvolvimento do sistema, porém apenas isso não é o bastante. Em muitos casos o profissional não conhece o domínio organizacional da aplicação em que está designado, e com isso vários aspectos relacionados com o ambiente organizacional, e que deveriam ser a base para a construção de sistemas, não são considerados.

Neste contexto, a modelagem organizacional é de grande importância para o "amadurecimento" das análises e decisões a serem tomadas no planejamento de sistemas,

além de direcionar para um levantamento de requisitos mais preciso e conseqüentemente para uma melhor qualidade final de produto.

Existe uma variedade de ferramentas para modelagem, contudo o *framework* i* pelas características já apresentadas e também pela fácil interpretação como foi visto, foi escolhido para fazer parte desta pesquisa, pois consideramos que essa riqueza de informação e facilidade de interpretação pode auxiliar fortemente na realização de estimativas de tamanho de software em sua etapa inicial. Mais especificamente, podemos observar que dependências associadas ao ator "Sistema de Gerencia de Hotel" podem ser exploradas para apoiar e orientar o processo de estimativas de software via técnica de análise de pontos por função.

Capítulo 3

Medição de Software

A medição de software, a partir da grande evolução da área de engenharia de software acabou se tornado cada vez mais importante, no contexto de entendimento e controle do desenvolvimento e gerenciamento de sistemas de software. A medição de software em alguns casos é medida para se ter noção da "qualidade", completude dos requisitos e qualidade de projetos, em outros casos se mede atributos do processo e do produto, possibilitando assim prever quando o software estará pronto para entrega ou ainda se o orçamento planejado será suficiente.

Neste capítulo serão apresentadas as técnicas de medição de software mais utilizadas no mercado, onde a Seção 3.1 irá apresentar Análise de Pontos por Função que é o foco da pesquisa. Por fim, as considerações finais são apresentadas na Seção 3.2.

3.1 Análise de Pontos de Função

A Análise de Pontos de Função (APF) é um método padrão para medir o tamanho funcional de um software em Pontos por Função (PF) a partir do ponto de vista do usuário. Está técnica foi originalmente concebida por Albrecht e ganhou popularidade a partir da criação do *International Function Point Users Group* (IFPUG), e em 2002 a mesma passou a ser reconhecida como padrão internacional (Vazquez *et al*, 2010).

O IFPUG é uma entidade composta por pessoas e empresas de diversos países, onde o objetivo da mesma é promover um melhor gerenciamento dos processos de desenvolvimento e manutenção de software com o uso de técnicas de medição de software (incluindo os PF). O IFPUG é promovido pelo trabalho voluntário de seus membros e não possui fins lucrativos (Vazquez *et al*, 2010).

Neste contexto, o IFPUG define como objetivo primário da técnica buscar a medição da funcionalidade que o usuário solicita e recebe, e do desenvolvimento e manutenção de maneira que a tecnologia, linguagem e arquitetura utilizada no desenvolvimento não

interfiram na medição. Além desses parâmetros, os processos devem ser suficientemente simples, assim minimizando ao máximo o trabalho.

Quando nos referimos ao ponto de vista do usuário e falamos que o método de medição APF é baseado neste mesmo ponto de vista, é importante esclarecer alguns pontos. O termo *usuário* tem um sentido mais amplo dentro da análise de medição, o **usuário aqui é qualquer pessoa ou coisa que interaja (envia ou receba dados) com a aplicação.** Por outro lado se considerássemos que a contagem de pontos por função fosse feita apenas no conceito com o usuário sendo uma pessoa, não seria possível medir sistemas que não tivessem interface com o usuário final.

A APF pode trazer vários benefícios como:

a) **Gerenciamento de escopo:** é possível determinar se os requisitos funcionais do sistema cresceram ou diminuíram, através da medição do projeto em cada fase de seu ciclo de vida.
b) **Gerenciamento de requisitos:** ao realizar o processo de contagem de pontos de função simultaneamente é feito uma análise sistemática da especificação dos requisitos, e com isso é realizado uma "revisão" dos mesmos que acaba contribuindo para a solidez e completeza dos requisitos.
c) **Estimar custo e recursos:** ao se realizar a APF no início do ciclo de vida do software é possível determinar seu tamanho funcional, esta medida pode ser utilizada como modelo de entrada para alguns modelos de estimativa de esforço, prazo e custo.
d) **Fundamentar a negociação de contratos:** a APF é utilizada por muitas empresas e órgãos públicos como meio de se gerar acordos e contratos sobre o software a ser desenvolvido.

Para realizar uma medição primeiramente precisa-se reunir toda documentação disponível sobre o sistema, caso não exista nada disponível (no caso de uma estimativa de um sistema), elaborar documentos a partir da perspectiva de negócio que facilitem e tornem mais precisas as estimativas. A documentação "padrão" a ser utilizada a princípio deve atender a seguinte exigência: descrever a funcionalidade entregue pelo usuário ou descrever a funcionalidade que é impactada pelo software medido e regras de negócio envolvidas. Alguns exemplos de documentação de grande valia são: diagramas de classe, diagramas de fluxo de dados, casos de uso, descrições procedurais, *layout* de relatórios e telas e manuais (Vazquez *et al*, 2010).

Devemos levar em consideração ao tentar realizar uma medição no ciclo de vida inicial do projeto (ainda não existe documentação), pois está pratica é considerada uma estimativa das funcionalidades do software que irá ser entregue ao final. Neste contexto conforme os requisitos vão ganhando forma é normal o aparecimento de funcionalidade que antes não haviam sido especificadas inicialmente, este problema em muitos casos se deve ao fato da "equipe" ainda não ter o conhecimento organizacional suficiente do ambiente para realizar tal estimativa, sendo assim é evidente que o resultado final será comprometido.

Uma etapa importante e que deve estar clara antes de iniciar o processo de contagem é a delimitação da fronteira de aplicação. Está delimitação é encarregada de separar o sistema de software do mundo exterior, que no caso seria os usuários, a definição de fronteira apesar de aparentar simples deve estar muito bem definida, pois caso ocorra o contrario as chances de ocorrer falhas no processo são enormes. O modo "mais recomendado" para tal etapa seria delinear esta fronteira a partir de uma perspectiva de negócio, evitando quaisquer considerações com técnicas envolvidas (Vazquez *et al*, 2010) e (Barcellos, 2007).

Após esses parâmetros estarem bem claros e definidos, o passo seguinte é a definição do escopo de contagem, o que significa dizer que é preciso identificar quais funções serão incluídas no processo de contagem. Contudo para melhor entendimento deste processo vamos especificar separadamente cada tipo de função.

3.1.1 Funções do Tipo Dado

São funções responsáveis por representar funcionalidades fornecidas pelo sistema ao usuário a fim de atender necessidades de armazenamento de dados. Essas funções podem ser classificadas em (Vazquez *et al*, 2010) e (Barcellos, 2007):

a) **Arquivo Lógico Interno (ALI):** grupo lógico de dados relacionados ou informação de controle identificado pelo usuário e mantido **dentro** da fronteira da aplicação. Sua principal intenção é manter os dados que sofrem manutenção através de processos elementares da aplicação que está sendo contada. Exemplo: tabelas de banco de dados atualizadas pela aplicação

b) **Arquivo de Interface Externa (AIE):** grupo lógico de dados relacionados ou informação de controle identificado pelo usuário e mantido **fora** da fronteira da aplicação. Sua principal intenção é armazenar dados referenciados através de uma ou

mais transações da aplicação sendo contada. Exemplo: tabelas de banco de dados lidas pela aplicação, porém atualizadas por outra aplicação.

Em APF quando nos referimos a "*Arquivos",* nos referimos na verdade a um conjunto de dados logicamente relacionados e reconhecidos pelo usuário, onde esse arquivo pode estar mapeado em um ou mais arquivos físicos do sistema ou tabelas de banco de dados. Assim podemos dizer que a forma como a aplicação implementa essas funcionalidades não são relevantes para determinação das funções do tipo dado.

Dessa forma os passos para a contagem das funções do tipo dado podem ser mapeados em quatros passos: descartar dados do código, identificar arquivos lógicos e classificar, identificar tipos de registro e por fim identificar tipos de dado.

Após a identificação dos arquivos em ALI ou AIE, cada arquivo deve ter sua complexidade determinada (**baixa, média, alta**) com base no número de Elemento do Tipo Dado (ETD) e número de Elemento do Tipo Registro (ETR). Já com a quantidade de ambas definidas, a classificação do arquivo é feita de acordo como é demonstrado na Tabela 3.1 (Barcellos, 2007).

Tabela 3.1: Complexidade de Arquivos Lógicos Internos e Arquivos de Interface Externa.

		Tipos de Elementos de Dados		
		1 a 19	***20 a 50***	***≥ 51***
Tipos de Elementos de Registros	***1***	*BAIXA*	*BAIXA*	*MÉDIA*
	2 a 5	*BAIXA*	*MÉDIA*	*ALTA*
	≥ 6	*MÉDIA*	*ALTA*	*ALTA*

Um ETD é definido como sendo um campo único reconhecido pelo usuário e não repetido, mantido ou recuperado de uma ALI ou AIE por meio de um processo, Exemplos (Vazquez *et al*, 2010):

a) Imagine um agendamento de um recebimento, onde a data de vencimento poderia estar da seguinte maneira dia, mês e ano (três campos), contudo mesmo para esses casos continua-se a considerar como um único tipo de dado (data, um ETD).

b) Imagine que para cada funcionário exista um controle do número de clientes (vários) que o mesmo é responsável, onde esse campo é calculado e armazenado em uma ALI, neste caso mesmo que possam existir vários clientes, é contado apenas um ETD (clientes).
c) Quando ocorre de um arquivo conter várias ocorrências do mesmo campo: valor janeiro, valor fevereiro e assim por diante, neste caso devem ser contados 2 ETD, ou seja, uma para mês e outro para valor.

Já um ETR é definido como um subgrupo de dados, com o mesmo sendo elemento de um ALI ou AIE. Estes subgrupos podem ser classificados como **opcionais:** que são aquelas em que o usuário tem a opção de não informar no processo que crie ou adicione dados ao arquivo, ou ainda como **obrigatórios:** que é o contrário do opcional, ou seja, são todos aqueles que sempre são requeridos pelo processo que cria ou adicione dados ao arquivo.

3.1.2 Funções do Tipo Transação

São funções responsáveis por representar os requisitos de processamento fornecidos pelo usuário. Essas funções podem ser classificadas em:

a) **Entrada Externa (EE):** transação responsável por processar dados ou informação procedentes de fora da fronteira de aplicação. Sua principal intenção é manter um ou mais arquivos lógicos (incluir, alterar e excluir cliente) e/ou modificar o comportamento do sistema.
b) **Saída Externa (SE):** transação responsável por enviar dados ou informação para fora da fronteira de aplicação. Identificada pela característica de apresentar informação ao usuário através de lógica de processamento que não seja apenas uma recuperação de dados, ou seja, seu processamento deve conter cálculos, ou criar dados derivados, ou manter um arquivo lógico interno, ou ainda alterar o comportamento do sistema. Exemplo relatório de totais de faturamento por cliente.
c) **Consulta Externa (CE):** transação responsável por enviar dados ou informação para fora da fronteira de aplicação. Identificada pela intenção de apresentar informações ao usuário pela simples recuperação de dados ou informação de controle de ALI e/ou AIE. Exemplos: consulta cadastro de clientes, verificação de senhas e recuperação de dados com base em parâmetros.

Para cada EE, SE e CE existem dois tipos de elementos que devem ser contados para cada função identificada:

a) Elemento do Tipo Dado (**ETD**)**:** considerado como sendo um campo único, não recursivo e não repetido.
b) Arquivo Referenciado **(AR**)**:** arquivos lógicos utilizados para processar a entrada e/ou saída (total de ALI e AIE utilizados pela transação).

Um ETD no contexto de funções do tipo transação pode ser contado da seguinte forma: conte um tipo de dado para cada atributo que atravesse a fronteira da aplicação (saindo e/ou entrando), que seja único, não repetido e reconhecido pelo usuário. Exemplos:

a) Nome do cliente e sua razão social, fornecidos pelo usuário ao adicionar o mesmo no sistema. Ao representar um gráfico do tipo pizza onde a legenda é contada como um tipo de dado e o valor numérico de cada fatia como sendo também outro tido de dado.
b) Contado um único tipo de dado para a capacidade do envio de alguma mensagem (erro, confirmação ou alerta) por parte do sistema. Exemplo: ao registrar a reserva de um quarto de um hotel em que o mesmo já estará ocupado na data prevista, onde neste caso o sistema enviará uma mensagem alertando o usuário do ocorrido.
c) Contado um único tipo de dado para a capacidade de especificar a ação tomada (mesmo que existam vários meios de ativar o mesmo processo deve-se conter apenas como sendo um único elemento tipo de dado). Exemplo: para salvar os dados na tela o usuário pode clicar no botão Salvar, ou usar a tecla de atalho CTRL + S (um ETD).

E um AR pode ser identificado e contado da seguinte forma:

a) Conte um arquivo referenciado para cada ALI mantido.
b) Conte um arquivo referenciado para cada ALI ou AIE lido durante o processamento.

Também é importante considerar na contagem que mesmo que haja vários registros dentro de um AIE ou ALI, devemos contar mesmo assim como sendo apenas um AR.

Assim como nas funções do tipo dado, as funções do tipo transação devem ter sua complexidade determinada (**baixa, média, alta**), com base no número de ETD e no número de AR. Já com a quantidade de ambas definidas a classificação deve ser feita de acordo como é demonstrado nas tabelas 3.2 e 3.3 (Barcellos, 2007) abaixo:

Tabela 3.2: Complexidade de **Entradas Externas (EE).**

Tipos de Arquivos Referenciados	*Tipos de Elementos de Dados*		
	1 a 4	*5 a 15*	*≥16*
0 a1	*BAIXA*	*BAIXA*	*MÉDIA*
2	*BAIXA*	*MÉDIA*	*ALTA*
≥3	*MÉDIA*	*ALTA*	*ALTA*

Tabela 3.3: Complexidade de **Saídas Externas (SE)** e **Consultas Externas (CE).**

Tipos de Arquivos Referenciados	*Tipos de Elementos de Dados*		
	1 a 5	*6 a 19*	*≥20*
0 a1	*BAIXA*	*BAIXA*	*MÉDIA*
2 a 3	*BAIXA*	*MÉDIA*	*ALTA*
≥4	*MÉDIA*	*ALTA*	*ALTA*

3.1.3 Cálculo de Tamanho Funcional

No momento de realizar o cálculo do tamanho funcional cada função (tipo dado ou transação) possui um peso em PF, este peso é determinado de acordo com a complexidade da função, que pode ser: baixa, média e alta.

A complexidade das funções do tipo **dado** é determinada por dois parâmetros: quantidade de ETD (campos) e quantidade de tipos de registro (subgrupos de dados dentro do arquivo). Já a complexidade das funções do tipo **transação** tem a complexidade definida através de dos parâmetros: ETD e quantidade de AR.

Com os arquivos lógicos internos e de interface externos já definidos em termos de complexidade, é hora de fazer a estimativa em PF para as funções **tipo dado**. A Tabela 3.4 (Barcellos, 2007) e (Vasquez *et al*, 2010) logo abaixo demonstra com é feita essa etapa do processo.

Tabela 3.4 Tabela de contribuição das funções do **tipo dado.**

Tipo de Função	**Complexidade Funcional**		**Totais por Tipo de Função**
ALI	Baixa Média Alta	x7 x10 x15	Soma de todas as ALI, vezes sua complexidade.
AIE	Baixa Média Alta	x5 x7 x10	Soma de todas as AIE, vezes sua complexidade

Seguindo a mesma ideia e com as transações de EE, SE e CE já definidas em termos de complexidade, é hora de fazer a estimativa em PF para as funções **tipo transação**. A Tabela 3.5 (Barcellos, 2007) e (Vasquez *et al*, 2010) logo abaixo demonstra como é feita esta etapa do processo.

Tabela 3.5: Tabela de contribuição das funções do **tipo transação.**

Tipo de Transação	**Complexidade Funcional**		**Totais por Tipo de Função**
EE	Baixa Média Alta	x3 x4 x6	Soma de todas as EE, vezes sua complexidade.
SE	Baixa Média Alta	x4 x5 x7	Soma de todas as SE, vezes sua complexidade
CE	Baixa Média Alta	x3 x4 x6	Soma de todas as CE, vezes sua complexidade.

3.1.4 Fator de Reajuste

A contagem de números de pontos de função demonstrada até agora, baseada apenas pelas funções de transação e funções do tipo dado, refletem apenas o tamanho funcional que o sistema fornecerá ao usuário final, sem considerar quaisquer singularidades do sistema. Isto ocorre, por exemplo, na implantação de um sistema, onde para um determinado cliente este sistema operando apenas em sua máquina (local) consegue satisfaz seus requisitos, já para outro cliente o mesmo deverá operar em arquitetura cliente servidor. É obvio que esta funcionalidade demonstrada que tende a mudar de cliente para cliente deve ser considerada de alguma forma na complexidade dos sistemas, sendo assim foi criado um "ajuste" com o intuito de melhor refletir esse tipo de característica singular na contagem do sistema.

Neste contexto para se ajustar os pontos de função encontrados após a etapa de cálculo do tamanho funcional, o sistema deve ser analisado referente a quatorze considerações: Comunicação de Dados, Processamento Distribuído, Desempenho, Configuração Altamente Utilizada, Taxa de Transações, Entrada de Dados On-Line, Eficiência do Usuário Final, Atualização *On-Line*, Processamento Complexo, Reutilização, Facilidade de Operação, Facilidade de Instalação, Múltiplos Locais e Modificações Facilitadas (Barcellos, 2007), (Vasquez *et al*, 2010) e (IFPUG, 2005).

Dessa forma para cada característica deve-se atribuir um nível de influência de zero a cinco, onde **zero** indica nenhuma influência, **um** influência mínima, **dois** influência moderada, **três** influência média, **quatro** influência significativa e **cinco** grande influência (IFPUG, 2005) e (Vazquez *et al*, 2010).

Para calcular o valor deste fator utiliza-se da seguinte conta: **VFA = (GIT * 0,01) + 0,65,** onde o Valor do Fator de Ajuste é o VFA e o Grau de Influência Total é o GIT, ou seja, a soma de todos os valores dos níveis de cada característica mencionada.

Agora com o valor de ajuste em calculado podemos ajustar os pontos de função, o processo é apenas a multiplicação dos pontos de função encontrados (não ajustados) com o fator de ajuste, utilizando a seguinte conta: **PFA = PFNA * VFA**, o resultado é o valor dos Pontos por Função Ajustados (PFA).

3.1.5 Exemplo de Contagem para um sistema de Hotel

Para melhor compreensão será apresentado um exemplo simples envolvendo uma estimativa do sistema de hotelaria "Hotel Real" (Girardello *et al*, 2009) (o mesmo do exemplo dos diagramas citados no Capítulo 2), porém não será feita uma contagem de todo o escopo, e sim de apenas de um contexto deste exemplo, pois o intuito neste caso é consolidar o conhecimento da contagem de pontos por função utilizando a APF. Dessa forma será feita a identificação e classificação de funções do tipo **dado** e **transação** dentro apenas deste contexto em específico do sistema.

O Hotel Real trabalha na área de hotelaria há mais de trinta anos. A empresa não possui nenhum sistema automatizado de informação implantado, todas as atividades de operação como cadastros de hóspedes e empresas, controle de diárias, entre outras, são realizadas manualmente com o auxílio de livros de registros, o **Apêndice A e B** (Girardello *et al*, 2009) trazem maiores informações sobre o hotel e suas necessidades. Desta forma a proposta de um sistema para a automação da maioria das atividades é bastante bem vinda pelos proprietários.

Dentro desta iniciativa de construção de um sistema é necessário primeiramente que se faça a contagem de pontos de função utilizando a APF, com o intuito de se obter uma estimativa do tamanho funcional. Como foi visto anteriormente existe a necessidade da elaboração de documentação, que será a fonte de nossas análises, e responsáveis por identificar futuras funções a serem consideradas no processo de contagem. Toda essa documentação que deve ser levantada se encontra nos A**pêndices A, B, C e D** (Girardello *et al*, 2009)**.**

Será iniciada a APF seguindo os passos de acordo com o que foi definido sobre a técnica neste trabalho, e ao longo dessa análise conforme for necessário a cada passo será analisando e demonstrando a documentação que se torna necessária para as devidas conclusões.

Primeiramente é preciso definir o tipo de contagem a ser realizado, mesmo este exemplo sendo retirado de um trabalho acadêmico onde o sistema já foi finalizado, o intuito desta pesquisa é analisar a APF no momento da estimativa, ou seja, quando o projeto está em sua parte inicial, verificando o levantamento de documentação necessária para a contagem. Assim o tipo de contagem a ser realizada refere-se hipoteticamente a estimativa deste software. Passo **(a)** analisado:

a) Determinação do tipo de contagem: A contagem a ser realizada neste contexto refere-se a um sistema a ser desenvolvido, ou seja, ainda não foi iniciado.

O passo seguinte, muito importante para o sucesso da contagem, é a delimitação da fronteira de aplicação, esta etapa irá explicitar o "que devemos medir", a definição é feita pela análise da documentação de requisitos **(Apêndice B)**, contexto onde o sistema atuará **(Apêndice A)**, e se há alguma interação com outra aplicação. Essa é a parte mais delicada do processo, pois não é nem um pouco intuitiva, e depende da interpretação do profissional sobre a documentação disponível "tentar" entender as regras de negócio envolvidas no meio, e as funcionalidades requeridas pelo cliente para então delimitar essa fronteira da melhor forma possível. Neste contexto e com os recursos disponíveis podemos concluir que a fronteira de aplicação neste sistema é de uma aplicação local (**Apêndice A**), focada na automatização de necessidades básicas de gerenciamento (**Apêndice A e B**) e que a princípio não terá nenhuma interação com aplicações externas (**apêndice A, B e C**).

Esta análise é de grande valia para a contagem do sistema como um "todo", apesar de que para o exemplo apresentado (fragmentos do escopo do sistema) não ser de "fundamental importância", uma vez que as funcionalidades a serem apresentadas são intuitivas no que se diz respeito a fronteira de aplicação "interna do sistema", ou seja, como citado no trabalho referem-se exatamente sobre a perspectiva do negócio. Passo **(b)** analisado:

b) Identificação da fronteira da aplicação: Consideramos que não exista interação com outros sistemas e que o mesmo tem seu escopo definido através do **apêndice A, B e C** (abstrato).

Com a fronteira de aplicação já definida o passo seguinte agora é identificar e classificar todas as funções do tipo **dado** e de **transação**. Primeiramente serão analisadas as funções do tipo dado, e para isto vamos especificar um contexto em específico do sistema "**Hotel Real"**, este contexto se refere apenas ao cadastramento de um hóspede, o Diagrama Entidade Relacionamento (DER) do sistema é necessário (**apêndice D**). Contudo para facilitar a análise deste contexto a figura 3.1 ilustra o mesmo dentro do DER (apenas a tabela hóspede).

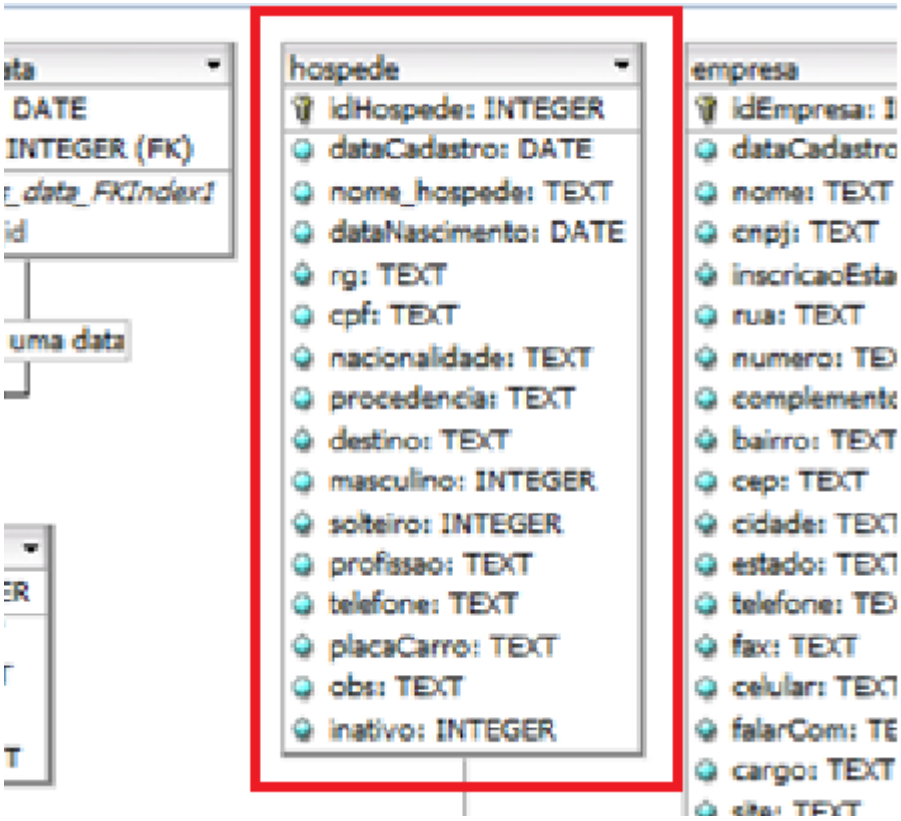


Figura 3.1: Tabela `hóspede` retirada do DER (**apêndice D).**

Inicialmente já é possível identificar que a tabela da figura 3.1 se trata de um ALI, pois do ponto de vista do usuário o armazenamento de um hóspede é satisfeito apenas pelos dados desta tabela, ou seja, não envolveria outras tabelas físicas em banco, nem tão pouco é dependente de informações externas. Analisando este ALI é possível identificar quinze ETD que são os próprios atributos da tabela, e identificar um único ETR, uma vez que para este contexto não existe nem um tipo de especialização, e sim apenas o subgrupo de dados referente à hóspede dentro deste ALI. Por fim, podemos afirmar que não existe nem uma AIE, uma vez que não ocorra nenhum tipo de interação por parte do sistema com outros sistemas (passo **b**). Passo **(c)** analisado.

c) Contabilizar as funções tipo dados – ALI e AIE: identificado que o arquivo/tabela "Hóspede" se trata de um ALI, e que o mesmo é o único neste exemplo. Este ALI possui 15 ETD e 1 ETR. Após a contagem das **funções do tipo dado** os resultados são agrupados como demonstrado na Tabela 3.6.

Tabela 3.6 Levantamento das **funções tipo dado** para o contexto de cadastramento de hóspede.

Arquivos Lógicos Internos	**Elementos Tipo Dado**	**Elementos Tipo Registro**
1 (Hóspede)	15 (Atributos de Hóspede)	1 (subgrupo de dados)
Arquivos de Interface Externa	**Elementos Tipo Dado**	**Elementos Tipo Registro**
0 (Não há interação com outros sistemas)	0	0

Com as funções do tipo dado definidas agora o próximo passo será a identificação das funções do tipo **transação**. A identificação deste tipo de função geralmente é a que requerer um maior número de informação para análise, além de exigir também um bom conhecimento do ambiente organizacional em que o sistema irá ser aplicado, pois qualquer funcionalidade não contada por falta de conhecimento das regras de negócio, ou até mesmo resultado de análise errônea compromete o resultado final, uma vez que estas funções são as de maior ocorrência dentro do processo de contagem.

Desta forma será feita a contagem e análise destas funções de maneira independente, separadas pelos passos **d** ao **f**. A documentação "disponível" para estás análises são: Requisitos do Sistema (RS) **(Apêndice B.1)** e Diagrama de Classe (DC) **(Apêndice C)**.

Podemos tirar boas conclusões ao analisarmos um DC (caso o mesmo esteja bem definido e avançado), em nosso caso especificamente na classe responsável por executar o processamento de dados de hóspede (Figura 3.2), é possível identificar algumas funções de caráter EE, que são: incluir, alterar e excluir, todas referente ao processamento de um hóspede.

Outra forma de identificar e/ou validar estas funções seria através da documentação de requisitos do sistema **(Apêndice A)**, onde é possível fazer uma busca aos requisitos referente à hóspede e validar algumas funções EE através dos seguintes requisitos: "Incluir Hóspedes" (**RF-01, Apêndice B.1**), "Alterar Hóspede" (**RF-02, Apêndice B.1**) e "Excluir Hóspedes" (**RF-03, Apêndice B.1**). Assim neste caso de contagem, tanto no diagrama de classes quanto nos requisitos do sistema foram possíveis identificarem funções EE existentes.

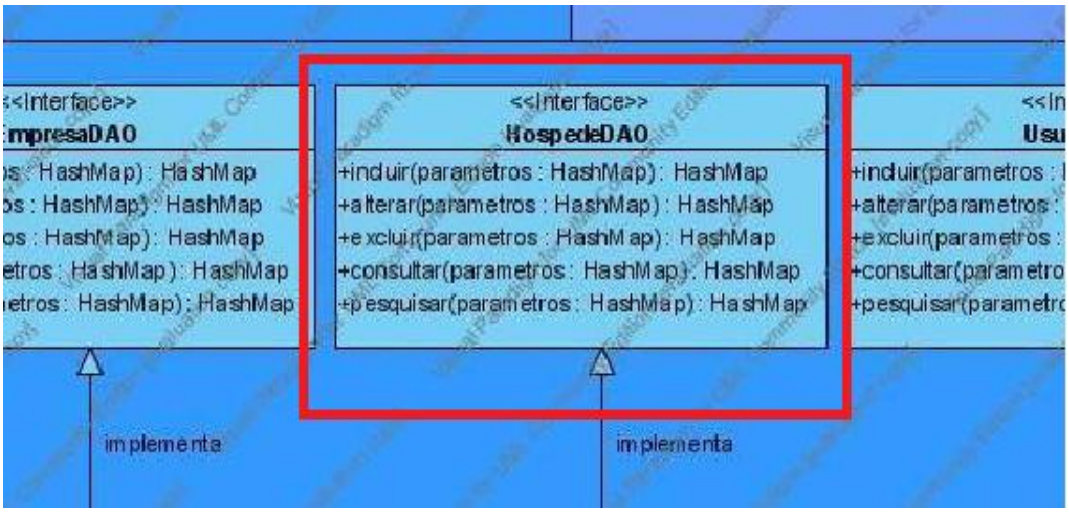


Figura 3.2: Classe HospedeDAO, retirada do Diagrama de Classes Apêndice C

Com as funções de EE identificadas, será contado o número de ETD e AR para cada uma delas. Para as EE de inclusão e alteração são consideradas como tendo cada uma 15 elementos do tipo dado (atributos que são inseridos e alterados), para a EE de exclusão é considerada como tendo apenas um único ETD (código identificador do hóspede), ambas as funções mencionadas podem ser identificados seja usando o DER (Figura 3.1), ou ainda o DC do sistema. Para todos os EE encontrados foi considerado um AR para cada, pois os mesmos referenciam apenas um único "arquivo" (Hóspede). Passo (**d**) analisado.

d) Contabilizar as funções tipo transação - Entradas Externas (EE): Identificado três EE referente ao contexto da aplicação.

Utilizando da mesma documentação (DER) podemos agora facilmente identificar a função do tipo CE: consultar. Considerando que esta consulta retorna todos os dados referente ao hóspede, e seguindo o mesmo raciocínio das funções encontradas anteriormente a consulta refere-se a 15 elementos ETD e um AR.

e) Contabilizar as funções tipo transação - Consultas Externas (CE): Identificado uma CE referente ao contexto da aplicação.

Para a análise de saídas externas consideremos o requisito "Gerar Relatório"(**RF-27, Apêndice B.1**), onde em sua descrição o mesmo releva a necessidade entre outras da geração de relatório dos hóspedes cadastrados em um determinado período, assim já podemos concluir uma SE para este relatório. Na documentação não é explicitado que informações dos hóspedes serão utilizadas no relatório, contudo consideremos neste caso que este relatório irá imprimir

as seguintes informações: identificador do hóspede, nome completo, data de cadastro, telefone e placa do carro. Dessa forma se pode concluir que para a SE consulta existam 5 ETD (as cinco saídas do relatório definidas) e 1 AR (Hóspede). Passo (**f**) analisado, os resultados de todas as análises das funções do tipo transação são demonstrados na Tabela 3.7.

f) Contabilizar as funções tipo transação - Saídas Externas (SE): Identificado apenas uma única SE.

Tabela 3.7: Levantamento das **funções tipo transação** para o sistema exemplo.

Entradas Externas	**Elementos Tipo Dado**	**Arquivos Referenciados**
1 (Inclusão) 1 (Alteração) 1 (Exclusão)	15 (dados do Hóspede) 15 (dados do Hóspede) 1 (identificador Hóspede)	1 (Hóspedes) 1 (Hóspedes) 1 (Hóspedes)
Consultas	**Elementos Tipo Dado**	**Arquivos Referenciados**
1 (Consultas aos dados cadastrais)	15 (dados do Hóspede)	1 (Hóspedes)
Saídas Externas	**Elementos Tipo Dado**	**Arquivos Referenciados**
1 (Relatório Gerencial)	5 (dados de saída do Relatório)	1 (Hóspede)

Seguindo esses passos conseguimos calcular os pontos de função não ajustados. Tabela 3.8:

Ao observar a Tabela 3.8, podemos facilmente fazer o cálculo dos pontos de função não ajustados, isto claro quando já temos a complexidade e o número de funções de cada tipo (coluna **"Itens Contados por Complexidade")**. Vale lembrar que esta complexidade é obtida da análise do levantamento de funções (seja ela dado ou transação) em relação a sua tabela de complexidade. Exemplo: vamos analisar a função "ALI" da Tabela 3.6, podemos notar que há ocorrência de uma única função "Baixa", uma vez que ela é a única encontrada na análise, ao analisarmos está tabela podemos notar que foram atribuídas a está "ALI" 15 ETD e apenas 1 ETR, e é com esses dados que conseguimos classificar está função de acordo com sua tabela de classificação (Tabela 3.1). Esta mesma ideia segue para as demais funções, sendo é claro que cada uma é classificada de acordo com sua tabela de classificação, as quais já foram ilustradas neste capítulo.

Tabela 3.8: Cálculo dos pontos de função **não** ajustados.

Função	Itens Contados por Complexidade	Contribuição	Total por Complexidade	Total de PFNA da Função
ALI	1 Baixa 0 Média 0 Alta	x 7 x 10 x 15	7 0 0	7
AIE	0 Baixa 0 Média 0 Alta	x 5 x 7 x 10	0 0 0	0
EE	3 Baixa 0 Média 0 Alta	x 3 x 4 x 6	9 0 0	9
CE	1 Baixa 0 Média 0 Alta	x 3 x 4 x 6	3 0 0	3
SE	1 Baixa 0 Média 0 Alta	x 4 x 5 x 7	4 0 0	4
TOTAL	**23 pontos de função não ajustados (PFNA)**			

O próximo passo após ter em mãos a contagem dos PFNA é calcular o fator de ajuste de acordo com aquelas características mencionadas anteriormente (Seção 3.1.4). Para analisarmos quais destas características nosso sistema abrange e qual o nível de influência de cada uma, será utilizado os requisitos não funcionais do sistema **(Apêndice A** e **B.1.2)**. O resultado está na Tabela 3.9, logo abaixo.

Tabela 3.9: Valores das características e justificativas

Características Gerais do Sistema	**Nível de Influência**	**Justificativa**
Comunicação de Dados	0	O Sistema trata-se de uma aplicação *stand alone*, portando sem comunicação de dados (Apêndice A).
Processamento Distribuído	0	Opera em micro *stand alone.*
Performance	1	Ao longo dos requisitos há alguns indícios da necessidade de uma *performance* aceitável.
Configuração Altamente utilizada	2	Descrito no requisito não funcional ”Configuração do Computador” (RNF-07 Apêndice B.1.2).
Volume de Transações	0	Nem um período de pico de transações esperada, pois são processamentos simples.
Entradas de dados *on-line*	0	Sem entradas (não há necessidade de acesso a internet no micro).
Eficiência do usuário final	3	Descrito no requisito não funcional ”Deve ser fácil de usar” (RNF-04 Apêndice B.1.2).
Atualização *on-line*	0	Sem ocorrências
Processamento Complexo	0	Processamento simples envolvendo a base de dados.
Reusabilidade	2	Descrito no requisito não funcional ”Fácil de Atualizar” (RNF-09 Apêndice B.1.2).
Facilidade de Instalação	0	Sem especificação
Facilidade de Operação	0	Sem especificação
Múltiplos locais	0	Hotel tem apenas uma sede.
Modificação facilitada	2	Descrito no requisito não funcional ”Fácil de Atualizar” (RNF-09 Apêndice B.1.2).
Total	**Grau de Influência Total = 10**	
VFA	**= (10 * 0,01) + 0,65 = 0,75**	

Com o VFA encontrado, basta multiplicar o mesmo pelos PFNA encontrados na Tabela 3.8. Assim temos que o resultado da contagem, **17** pontos de função ajustados para a aplicação.

3.2 Considerações Finais

É importante estar claro que pontos de função não medem diretamente esforço, produtividade ou custo, pois o mesmo é exclusivamente uma medida de tamanho funcional do software. Contudo este tamanho em conjuntos com outras variáveis pode sim e deve ser utilizado para medir esses parâmetros.

Também é evidente que esses parâmetros são de vital importância para o gerenciamento de um projeto de sistema de software, e para isso a utilização de técnicas de medição de software (com freqüência) dentro de uma empresa é um caminho promissor para viabilizar e melhorar o processo de gerenciamento dentro da mesma.

Por outro lado é evidente o alto grau de risco para possíveis "ambigüidades" de interpretação, e como são baseadas em possíveis análises corretas se torna muito vulnerável, uma vez que existe a necessidade de um alto grau de abstração, que dependeria também de uma boa documentação em mãos.

Capítulo 4

Integrando i* ao Processo de Medição de Software

Neste capítulo será apresentado o processo de integração do *framework i** (modelagem organizacional) estudada no Capítulo 2 ao processo de medição de software, estudada no Capítulo 3. Por fim, as considerações finais são apresentadas na Seção 4.2.

4.1 Integrando I* ao processo de APF.

A análise de pontos por função como já foi apresentado anteriormente é uma técnica que tem como objetivo medir o tamanho funcional de um software do ponto de vista do usuário, sendo assim o processo de medição deve ser baseado nas funcionalidades que o usuário solicita e recebe, e é esse fator que viabiliza a APF de ser considerada independente da tecnologia utilizada no desenvolvimento do sistema.

Neste contexto em que essas funcionalidades serão retiradas do ponto de vista do usuário, é fundamental extrair os mesmos da melhor forma possível. No processo de APF o IFPUG orienta que na identificação das funções durante o processo se leve em consideração a visão do usuário em relação a funcionalidades e as **regras de negócio**. As funcionalidades podem ser consultadas no documento de requisitos, ou outro qualquer, ou ainda diretamente com o cliente/usuário. Contudo quando procuramos algum "amparo" dentro das referências bibliográficas sobre APF não encontramos nenhum tipo de técnica ou ferramenta referente ao entendimento das regras de negócio utilizadas durante o processo de medição, até mesmo o próprio IFPUG ao relatar a importância das regras de negócio dentro do processo de medição não cita qualquer orientação/sugestão, técnica ou ferramenta para facilitar este tipo de entendimento.

A estimativa de software realizada antes do desenvolvimento do sistema pode ser feita por vários motivos, contudo supondo que o usuário necessita de um orçamento de um sistema, isso acarretaria para a "empresa" tempo necessário para o levantamento da documentação e

aplicação da técnica. Porém é claro que o usuário/cliente ao requerer um orçamento não está disposto a esperar um longo tempo pela resposta. Assim se essa estimativa baseada em funcionalidades do usuário e regras de negócio pudesse ser realizada de forma mais objetiva, o resultado teria uma melhor qualidade e na maioria dos casos exigiria um tempo menor, uma vez que o IFPUG afirma que para casos onde ainda não se encontre toda a documentação suficiente disponível é preciso buscar acesso aos "especialistas" no negócio para complementar essas falhas.

Dessa forma se a base do processo realmente são as funcionalidades do usuário e as **regras de negócio**, por que não ao invés de confeccionar toda uma gama de documentação e ao longo do processo e das necessidades tentar "descobrir" as regras de negócio e identificar as funções não partimos pelo caminho inverso, ou seja, definimos bem o ambiente organizacional tal como as regras de negócio, já pensando em possíveis funções para então após isso confeccionar documentações específicas apenas para suprir informações que faltam para complementar, identificar e classificar as funções. Dessa forma já seria viável ter uma visão geral de possíveis funções, ambiente organizacional e regras de negócio.

Neste contexto ao realizar a estimativa de sistemas de software a serem desenvolvidos a mesma abordagem citada poderia ser seguida, de maneira que o primeiro passo do processo fosse o entendimento do ambiente organizacional, obtendo também o maior conhecimento possível das regras de negócio envolvidas. Ainda se possível executar esse passo com auxilio do usuário, além de ao mesmo tempo já identificar as possíveis funções que farão parte do processo. Dessa forma esta etapa funcionaria como um meio norteador tanto para o levantamento da documentação quanto para o andamento da contagem e futuras conclusões.

Para o levantamento da documentação (que para este caso ainda seria feita a partir das necessidades), onde agora com o conhecimento organizacional, das regras de negócio e das possíveis funções identificadas, a confecção destes documentos deve se tornar menos custosa, mais ágil e partindo de uma perspectiva que atenda as dependências de informação e funções identificadas inicialmente.

Imaginando ainda o mesmo cenário onde o usuário/cliente solicita um orçamento de um sistema, observa-se também o outro lado, ou seja, o da empresa que para realizar todo o processo de medição e levantamento de documentação apenas para realizar um orçamento "sem compromisso" está associado a um alto risco. Neste contexto, é fato que a demora e o esforço acaba se tornando ponto negativo para ambas as partes envolvidas.

Entretanto, se utilizarmos o *framework* organizacional i* para a realização da etapa inicial de entendimento das regras de negócio e intencionalidades dos atores envolvidos no processo organizacional, incluindo as expectativas em relação ao sistema computacional a ser desenvolvido, poderíamos nortear mais consistentemente o processo de medição bem como tornar este processo mais ágil e viável, uma vez que o conhecimento modelado pelo *framework* permitiria::

- **Uma estimativa em um tempo menor:** considerando que com o conhecimento do ambiente organizacional, regras de negócio e identificação de possíveis funções facilita-se a elaboração da documentação necessária e/ou em alguns casos torna-se possível eliminar grande parte desta documentação, além de facilitar/agilizar a análise e classificação das funções durante o processo.
- **Qualidade aceitável:** considerando que o IFPUG afirma que a APF é restritamente uma medição das funcionalidades e regras de negócio envolvidas para o usuário, sem qualquer ligação com a tecnologia e linguagem adotada, ao especificarmos bem as funcionalidades, regras de negócio e o ambiente vivenciado pelo usuário estaríamos identificando funções **estratégicas e essenciais** que devem ser consideradas em um processo de APF.
- **Conhecimento para o andamento do projeto:** caso este orçamento venha a se tornar um projeto de desenvolvimento de fato, o conhecimento organizacional levantado nesta etapa é de grande valia para a construção do sistema, como foi visto capitulo 2.

É evidente que as regras de negócio mudam de sistema para sistema, bem como as necessidades iniciais requisitadas pelo usuário, que podem mudar com o decorrer do projeto. Neste contexto, também teríamos mais um beneficio ao utilizar o *framework i** para "visualizar" de uma melhor maneira as regras de negócio envolvidas no ambiente organizacional, que facilitaria identificar e alterar possíveis mudanças nas necessidades do usuário durante o processo, e ao longo do desenvolvimento do sistema.

Para demonstrar como pode ser realizada a integração do *framework i** ao processo de medição, aplicaremos nossa proposta a um estudo de caso baseado no exemplo de medição realizado na Seção 3.1.5. Primeiramente vejamos que já existe a modelagem organizacional do sistema, conforme ilustrado na figura 2.2 do capitulo 2 (Modelo SD), e na figura

2.5(Modelo SR). Contudo, serão descartadas para confeccionar um novo SD e SR partindo dos passos da proposta a serem apresentados.

Também é de nosso conhecimento que no modelo organizacional as dependências de *soft-goal* são informações importantes indicadores de alguns requisitos de qualidade esperados do sistema. Nesse contexto, outra vantagem da utilização da modelagem no auxilio ao processo de medição seria aproveitar as dependências *soft-goals* para já ir pensando em possíveis caracterizas do sistema (quatorze segundo IFPUG) que podem ser levantadas com o intuito de auxiliar na etapa de ajuste aos PFNA encontrados.

Assim como no exemplo da Seção 3.1.5, no qual foi proposta a medição usando o processo tradicional de APF, para o contexto hóspede, mais especificamente para as funcionalidades envolvidas no cadastro de um hóspede, o mesmo contexto será utilizado para realizar o processo agora com a integração do *framework i**. No estudo de caso a ser apresentado estes modelos já existem, contudo serão construídos novamente para possibilitar a realização do processo de medição via APF. A seguir apresenta-se passo a passo o processo de integração do *framework i** com a medição via APF. Cabe ressaltar que em cada passo, são apresentadas diretrizes para auxiliar nesta integração.

1º Passo: Construção do Modelo SD

Diretriz 1: realizar este passo com a presença do cliente/usuário, buscando extrair do mesmo as funcionalidades e regras de negócio envolvidas no contexto.

Diretriz 2: inserir no modelo um ator com o objetivo de representar o sistema para qual está sendo feita a medição.

Diretriz 3: identificar os atores que tem alguma dependência com esse ator "sistema", e encontrar dependências do tipo objetivo, *soft-goal,* tarefa e recurso entre estes atores e o sistema.

Diretriz 4: identificar *soft-goals* complementares para o modelo, buscando encontrar as mesmas considerando as quatorze características do fator de ajuste previstos pela IFPUG (ver Capítulo 3, seção 3.1.4).

Diretriz 5: Complementar o modelo SD incluindo demais atores e dependências do tipo objetivo, tarefa, *soft-goal* e recurso entre os mesmos, visando representar todas as intencionalidades neste ambiente.

Baseado nas diretrizes do 1º passo foi construído o modelo SD para o exemplo de gerenciamento de hóspedes, já identificando os possíveis *soft-goals* complementares. A figura 4.1 ilustra este modelo.

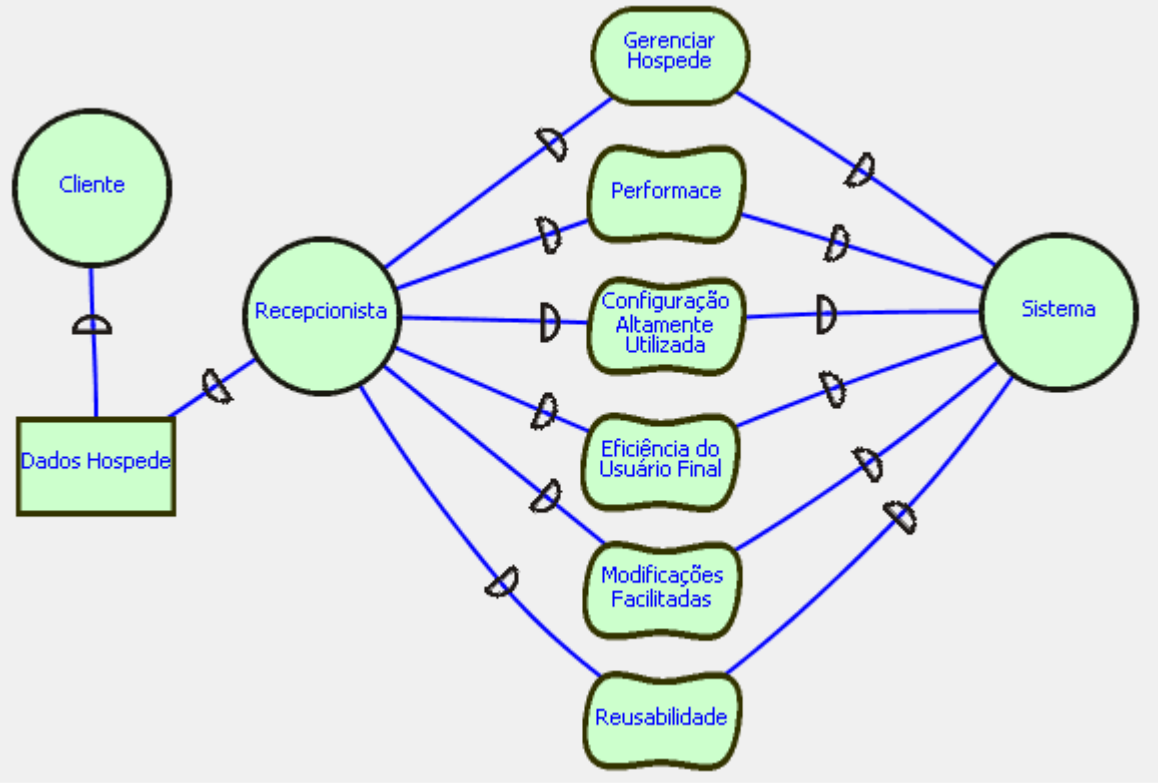


Figura 4.1: Modelo SD (contexto gerenciamento de Hóspedes) utilizando dos *soft-goal* para identificar características de ajuste.

A figura 4.1 traz o modelo SD apenas com os atores e dependências que farão parte do processo de medição, além de agora aparecerem cinco novas dependências do tipo *soft-goal* (modelo SR tende a explorar melhor essas dependências), que representam as necessidades de que o sistema terá que satisfazer.

As descobertas dessas novas dependências foram baseadas nos mesmos requisitos não funcionais utilizados para identificação dos mesmos na secção 3.1.5. Contudo, em um processo onde ainda não se tenha essas informações á proposta é que o profissional realize a modelagem com o auxílio do cliente/usuário (**Diretriz 1**).

Seguindo a ideia do *framework i** o próximo passo a ser desenvolvido seria o modelo SR, para demonstrar de maneira mais detalhada as razões estratégicas envolvidas no contexto. Esse detalhamento para o processo de medição de software pode e deve ser aproveitado, ou

seja, agora que é preciso realizar essa modelagem considerando funcionalidades esperadas pelo cliente/usuário, e abstrair as regras de negócio necessárias para o andamento da contagem.

2º Passo: Construção do Modelo SR

Diretriz 6: realizar esse passo com a presença do cliente/usuário abstraindo as funcionalidades e regras de negócio envolvidas no meio. Detalhar com auxílio do mesmo as razões estratégicas associadas ao ator definido da diretriz 2. Figura 4.2.

Diretriz 7: delimitar a fronteira de aplicação, representado-a no modelo pelo círculo de expansão do ator. Será incluso dentro da fronteira apenas as dependências (funcionalidades/regras) a serem de responsabilidade do sistema.

Diretriz 8: classificar e validar as dependências *soft-goals* encontradas no modelo SD (construído no 1º passo), de acordo com o peso (0-5) definido juntamente com o usuário.

Diretriz 9: se possível identificar a partir do círculo de expansão as prováveis funções do tipo dado.

Diretriz 10: identificar possíveis funções de transação (EE, CE e SE) e dado (ALI e AIE) durante o processo de modelagem, identificando as dependências e nomeando-as da seguinte maneira, {nome da dependência(tipo de função)[número de dados manipulados]}. Figura 4.3.

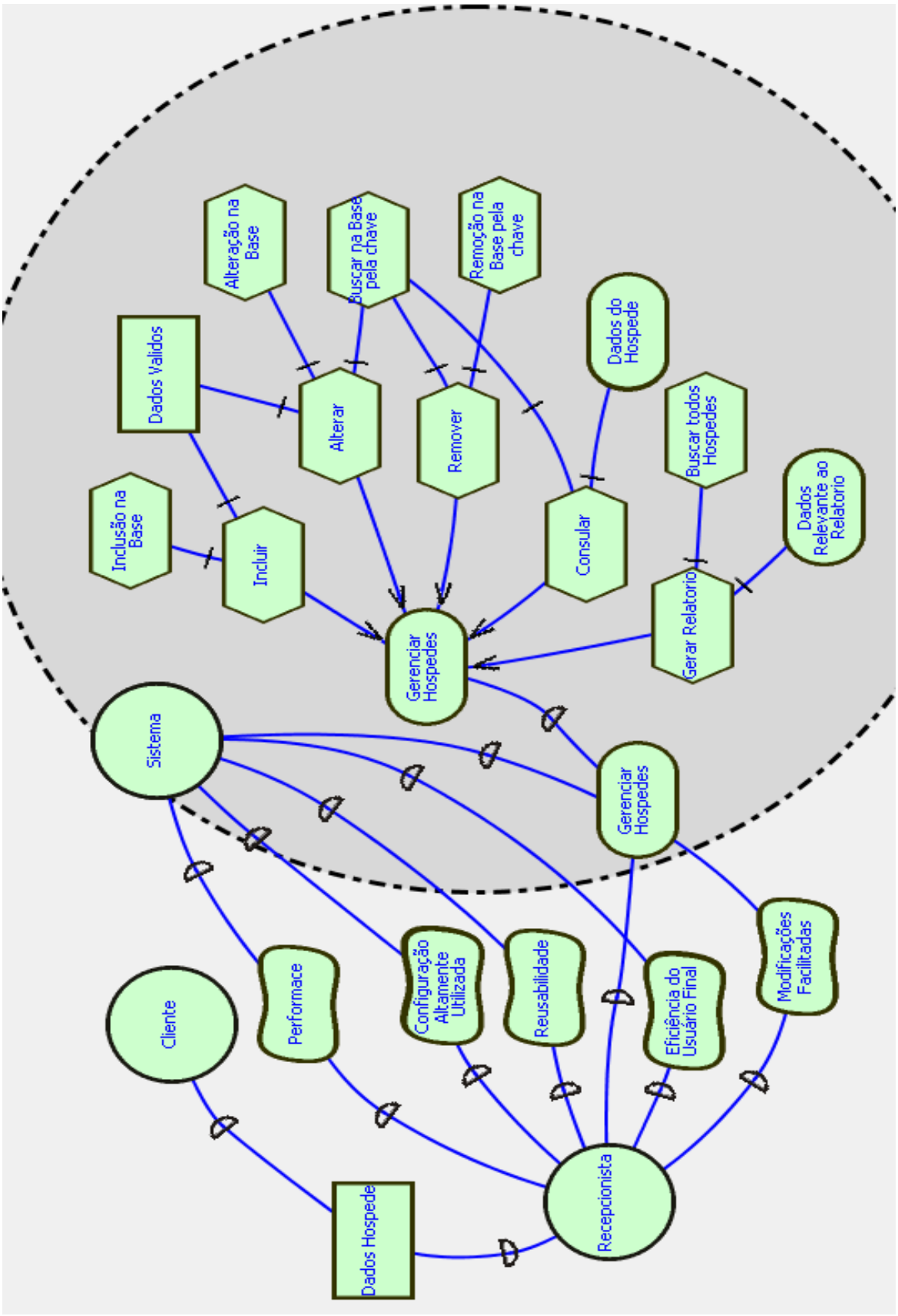


Figura 4.2: Modelo SR levantado.

A partir do SR é possível identificar a **fronteira de aplicação** (algo que foi bem mais "abstrato" na contagem sem o auxílio do *framework*, do capitulo anterior), uma vez que foi delimitada a mesma como uma equivalência ao "círculo" de expansão do ator dentro do SR (onde o ator é o sistema), representado pela **Diretriz 7**. Assim torna a fronteira algo visível para o decorrer do processo, permitindo análises mais corretas e fáceis na identificação de funções. A figura 4.3 mostra como ficaria a expansão do ator simulando a fronteira do ator "Sistema", onde no caso é possível delimitar o mesmo e perceber ainda mais nitidamente que os dados de hóspede não são de responsabilidade de outra aplicação, e sim do próprio usuário/cliente (ALI).

Ainda no modelo SR, também vamos dar a devida atenção as dependências *soft-goals* previamente encontradas no modelo SD da figura 4.1. Primeiramente é o momento de validar essas dependências e/ou encontrar outras juntamente com o usuário/cliente quando possível, e também para facilitar a análise e o cálculo dessas características no momento do ajuste está sendo proposta a seguinte alteração. Será colocando junto com o nome do *soft-goal* o peso de cada um seguindo os valores mencionados na Seção 3.1.4. Assim, já teríamos as características que devem ser consideradas no fator de reajuste juntamente com seu peso (determinando o mesmo junto com o usuário/cliente), facilitando e agilizando o processo.

A figura 4.3ilustra o modelo SR com as alterações de dependências, inclusão dos pesos e a "fronteira de aplicação", além de trazer as já conhecidas características da modelagem SR.

Analisando a Figura 4.3podemos notar que houve um maior volume de informação incluída no SR em relação ao SD. Na figura 4.3foram previamente estipulados cinco objetivos *soft-goal,* os quais a partir do SR foram levantados seus pesos. Essa etapa ocorreu pela necessidade de buscar maiores informações (preferencialmente com o usuário/cliente) sobre os *soft-goal* inicialmente previstos. Assim essa etapa permitiu que os *soft-goals* fossem caracterizados e classificados já se pensando em atender os padrões do fator de reajuste determinado pelo IFPUG (visto na Seção 3.1.4).

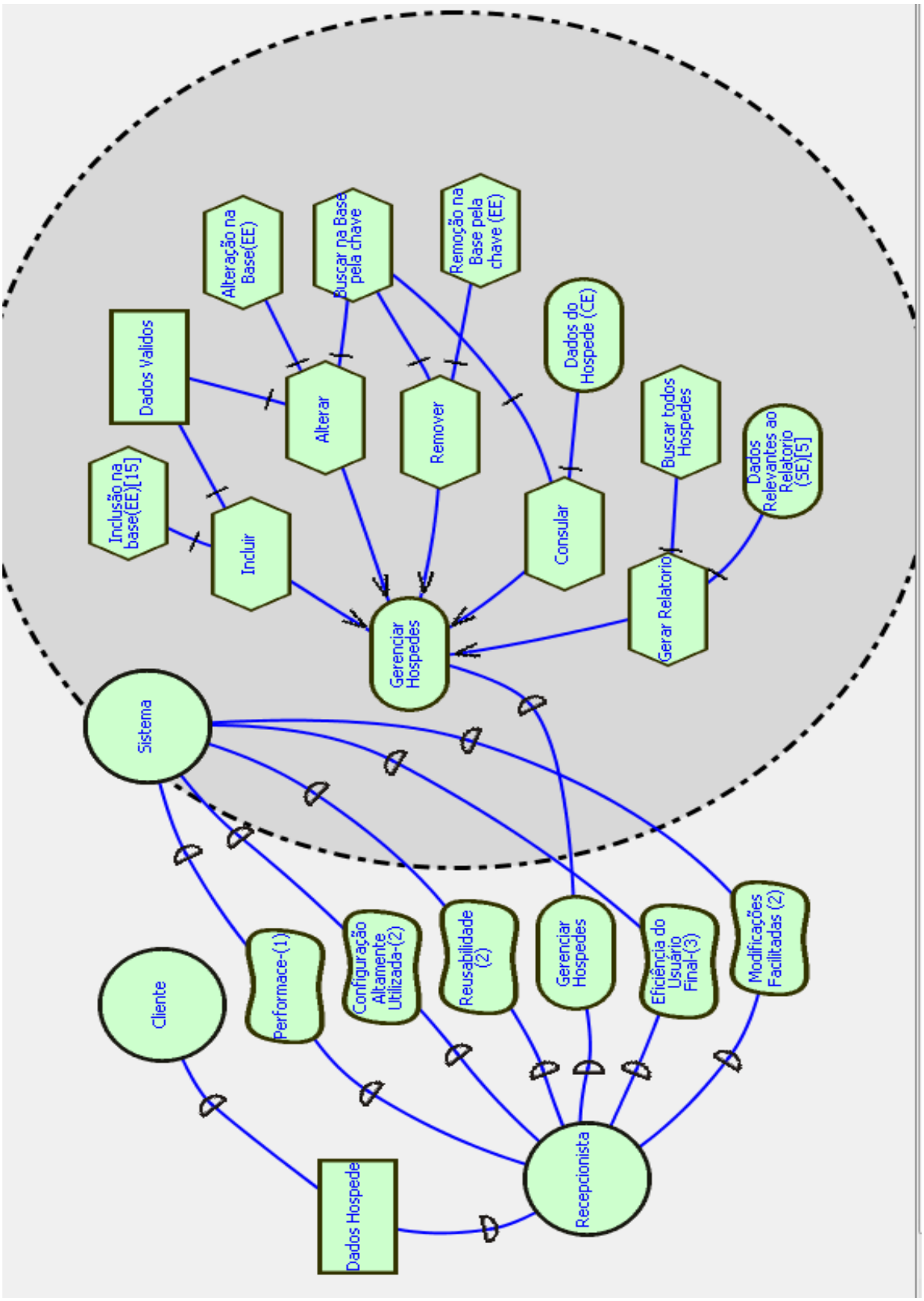


Figura 4.3: Modelo SR com informações importantes para o processo de medição.

Assim foi seguindo essa ideia que o *soft-goal* inicial "**Performance**" após o levantamento do peso que o mesmo terá no sistema (características do fator de reajuste), foi definido entre parênteses, seguindo a mesma ideia de atribuição de peso utilizada anteriormente nos exemplos. O mesmo ocorre para as demais características do fator de reajuste (*soft-goal*) (**Diretriz 8**).

O círculo de expansão envolvido no ator "Sistema" agora é uma forma de visão sobre a fronteira de aplicação, que quando bem definida proporciona qualidade "visual" sobre a fronteira e sobre o comportamento da mesma no contexto, assim contribuindo também para uma boa qualidade na medição de software pela APF, uma vez que sem a devida delimitação do escopo de contagem o resultado final encontrado tende a se distanciar do real tamanho final. Podemos fazer uma analogia do modelo SR construído com a "forma ideal" de visão e separação da fronteira proposta pelo IFPUG, como demonstra na Figura 4.4.

Neste contexto podemos notar que é possível distinguir a **aplicação a ser contada** e que a manipulação dos dados não tem influência com nenhuma aplicação externa, assim se tratando apenas do usuário externo, lembrando que a identificação da fronteira de aplicação é a etapa mais importante a ser realizada, uma vez que ao definir a mesma de forma incorreta todo o processo de medição estará comprometido, além é claro de ser parâmetro para identificar as funções ALI e AIE durante o processo. O usuário externo é quem ira realizar a manipulação do sistema, e conseqüentemente quem irá fornecer as informações para a aplicação (**Diretriz 9**).

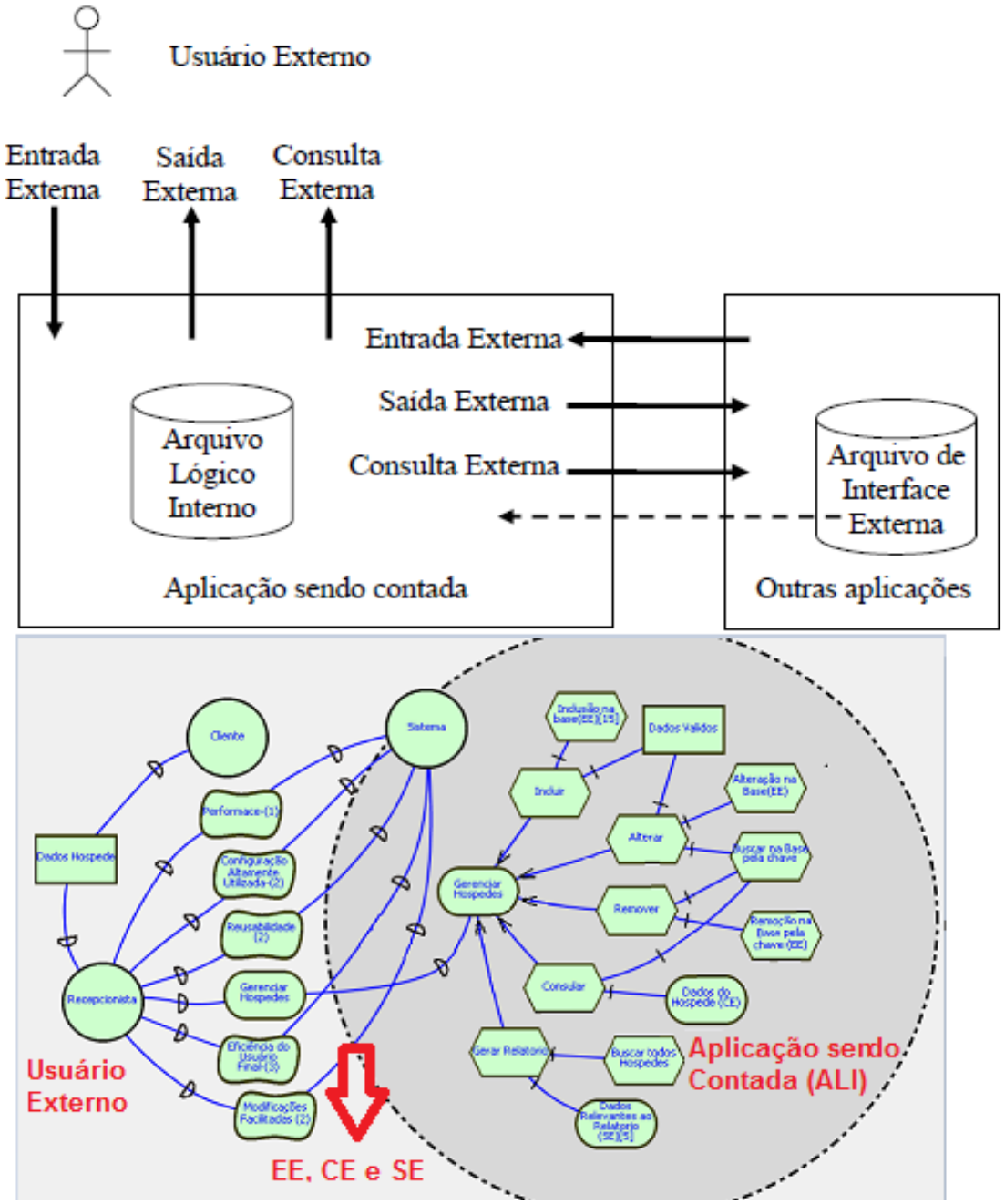


Figura 4.4: Analogia entre o modelo SR realizado com a divisão ideal da aplicação recomendada pela IFPUG.

Já as **entradas**, **consultas** e **saídas** externas, estão centralizadas na dependência de objetivo "Gerenciar Hospede", todas com a participação apenas do usuário externo "Recepcionista". Dessa forma o círculo de expansão permite uma delimitação adequada para o processo de contagem.

Ainda observando as entradas, consultas e saídas, é possível notar que na modelagem da figura 4.4existem algumas dependências já com possível identificação de tipo de função que cada uma trata, como é o caso da dependência "Incluir", onde quando o mesmo foi subdividido visando o detalhamento no modelo SR foi levantada juntamente com o cliente/usuário que uma das divisões se trata de uma entrada externa de dados, e que para isso seriam consideradas como entradas 15 dados referentes ao hóspede para serem armazenados no sistema. Assim no momento do detalhamento além de colocar na descrição que se trata de uma inclusão na base de dados, também é adicionado o tipo de função EE e também é adicionado o número de dados a serem inseridos quando se tem essa informação, ou seja, a descrição ficaria como a demonstrada na figura 4.4 ("**Inclusão na base (EE) [15]**"), **(Diretriz 10)**.

Neste mesmo contexto foi identificada uma possível **CE** ("Dados de Hospede(CE)") e uma **EE** ("Remoção na Base pela chave(EE)"). Contudo nestes dois casos o número de dados manipulados não foram apresentados (como ocorreu no caso anterior), como uma forma de demonstração de casos onde não havendo informação necessária para tomar essa decisão ou o cliente/usuário por algum motivo não tem esse conhecimento, é feito a descrição da dependência sem o número de dados manipulados. Posteriormente essas informações deveram ser levantadas aplicando a diretriz 11 que ainda será apresentada no 3º passo.

Para o caso da **SE** representada pela dependência "Dados relevantes ao Relatório (SE)[5]" foi levantado o mesmo a partir de uma possível necessidade de geração de um relatório gerencial a pedido do cliente/usuário, onde dentre os quinze dados a serem inclusos e conseqüentemente armazenados apenas cinco são relevantes estarem impressos neste relatório. Assim seguindo a mesma ideia de nomenclatura das dependências apresentadas foi criada e nomeada essa dependência.

3º Passo: Validar pendências do 2º passo e Aplicar a APF

Diretriz 11: procurar informação/recursos que validem as funções com alguma(s) pendência(s) vindas do 2º passo. Essa validação pode ocorrer por meio de geração de documentação ou ainda em uma nova consulta ao usuário/cliente. Atualizar as pendências no modelo SD e SR.

Diretriz 12: utilizar as informações levantadas nas modelagens para a análise durante o processo de APF. Realizar os passos da APF (**a-f**).

Neste contexto temos a etapa da utilização do *framework* i* concluída (2º passo), onde agora o passo seguinte (3º passo) seria a contagem pela APF, como visto na Seção 3.1.5. Contudo, agora como muitas das dificuldades encontradas na medição anterior já sanadas nesta etapa, e as que não foram sanadas se tornaram bem mais fáceis de serem analisadas, tornando fácil de definir corretamente as funções da APF, uma vez que toda a funcionalidade e regra de negócio agora estão visíveis e compreendidas para o processo. Para exemplificar essa maior facilidade serão apresentados todos os passos seguidos e analisados na Seção 3.1.5 (**a-f**) **(Diretriz 12)**, sendo que agora os mesmos serão realizados e analisados com o auxilio dos resultados obtidos nos modelos organizacionais SD e SR construídos da diretriz 1 a diretriz 10.

Passos do exemplo da APF, Seção 3.1.5:

a) Determinação do tipo de contagem: Para esse passo ainda nada mudaria. O tipo de contagem refere-se a um sistema a ser desenvolvido (estimativa).

b) Identificação da fronteira da aplicação: Certamente é o passo mais beneficiado com a integração do *framework* i*, pois na contagem realizada anteriormente este passo foi o mais abstrato e menos intuitivo de todo o processo, onde o responsável pela medição deveria, sem qualquer orientação, procurar em todas as documentações disponíveis informações "fragmentadas" para assim concluir da melhor forma possível a fronteira de aplicação. Assim, agora observando a modelagem SR para este exemplo já fica claro o escopo de contagem, assim facilitando e muito a análise desse passo e diminuindo as chances de erro nessa análise.

c) Contabilizar as funções tipo dados - ALI e AIE: No caso anterior a definição do ALI foi realizada considerando a fronteira de aplicação, que caso tivesse sido definida de forma errada já teria comprometido essa análise. Contudo, agora observando a modelagem da figura 4.4, é correto afirmar que se trata de um ALI, uma vez que toda a manipulação de informação é realizada dentro da fronteira de aplicação. Já para a identificação de 15 ETD que seriam os dados a serem armazenados, neste contexto foram extraídos hipoteticamente do cliente/usuário e definidos na dependência "**Inclusão na base (EE) [15]**", onde 15 representam o número de dados manipulados.

d) Contabilizar as funções tipo transação – Entradas Externas (EE): A partir da modelagem organizacional (SD e SR) já foram identificadas inicialmente três funções

EE. Os atores da figura 4.4identificados com “(EE)”, respectivamente serão as funções “Inclusão”, “Alteração” e “Exclusão”. Para a função de inclusão já temos no ator o número de dados manipulados, que para a medição é o número de ETD. Já para as funções de alteração e exclusão como não foram informadas a quantidade de dados manipulados pelas mesmas fica a pendência de buscar essa informação com o usuário novamente ou em documentação, porém como essa ausência foi realizada de propósito é claro que esses dados serão os mesmos da medição realizada no capitulo anterior. Assim se existe documentação complementar que permita obter estes dados, a mesma deve ser utilizada. Caso contrário, o cliente usuário deve ser consultado posteriormente complementando os modelos organizacionais gerados.

e) **Contabilizar as funções tipo transação – Consultas Externas (CE):** Uma função do tipo CE, respectivamente representada pelo ator “Dados do Hóspede (CE)”. Como o número de dados manipulados referente ao hóspede são 15, o número de ETD, para essa CE também serão 15. E um único ETR já que o hóspede não possui nenhum tipo de especialização.

f) **Contabilizar as funções tipo transação – Saídas Externas (SE):** Identificamos na modelagem uma única função do tipo SE, referente ao ator “Dados Relevantes ao Relatório (SE)[5]”. O número de ETD para essa função foi definido 5, uma vez que este é o número de dados que deverá estar presente neste relatório gerencial.

Neste contexto, conseguimos realizar a análise das funções referentes ao processo de contagem de forma estratégica e mais facilitada que a realizada na Seção 3.1.5, além de já ter adiantado a análise das características necessárias para o ajuste da contagem, assim deixando este processo praticamente concluído, cabendo apenas a aplicação da fórmula de ajuste.

Outro benéficio já citado seria a utilização do *framework i** para auxilio a um futuro processo de desenvolvimento do sistema, uma vez que o mesmo contém rica informação organizacional para a equipe de desenvolvimento.

4.2 Considerações Finais.

Este capítulo apresentou a possibilidade de integração do *framework* organizacional *i** ao processo de medição de software, ambas técnicas estudadas nos capítulos 2 e 3 respectivamente.

Para demonstração dessa integração foi utilizado o exemplo apresentado na Seção 3.1.5, sobre o contexto de cadastro de hóspedes, com o intuito de consolidar essa integração de forma que as vantagens do mesmo se tornassem mais clara.

Neste contexto foi realizada a medição junto com o *framework i**, deixando clara a grande importância que tem para o processo de medição o conhecimento prévio sobre as funcionalidades e regras de negócio envolvidas, assim evidenciando a importância que tem a participação do cliente/usuário durante o processo, e a utilização de uma modelagem organizacional para ilustrar o ambiente. Também é possível notar que a modelagem realizada durante o processo deve ser aproveitada para o passo seguinte que seria o de desenvolvimento da aplicação contada.

Capítulo 5

Aplicação em um Estudo de Caso.

Neste capítulo será apresentado um estudo de caso exemplificando a integração do *framework i** (modelagem organizacional) estudada no Capítulo 2 ao processo de medição de software, estudada no Capítulo 3, a outro contexto na Seção 5.1. Por fim, as considerações finais são apresentadas na Seção 5.2.

5.1 Exemplo da Integração sobre o Contexto "Gerenciar Reservas".

O primeiro passo como já sabemos é a criação da modelagem SD sobre o contexto "Gerenciar Reservas" aplicando as **diretrizes do 1º passo** (1 ao 5), apresentadas no capítulo anterior, da mesma forma como foi utilizado para o contexto referente à "Hóspede". A figura 5.1 ilustra o modelo SD.

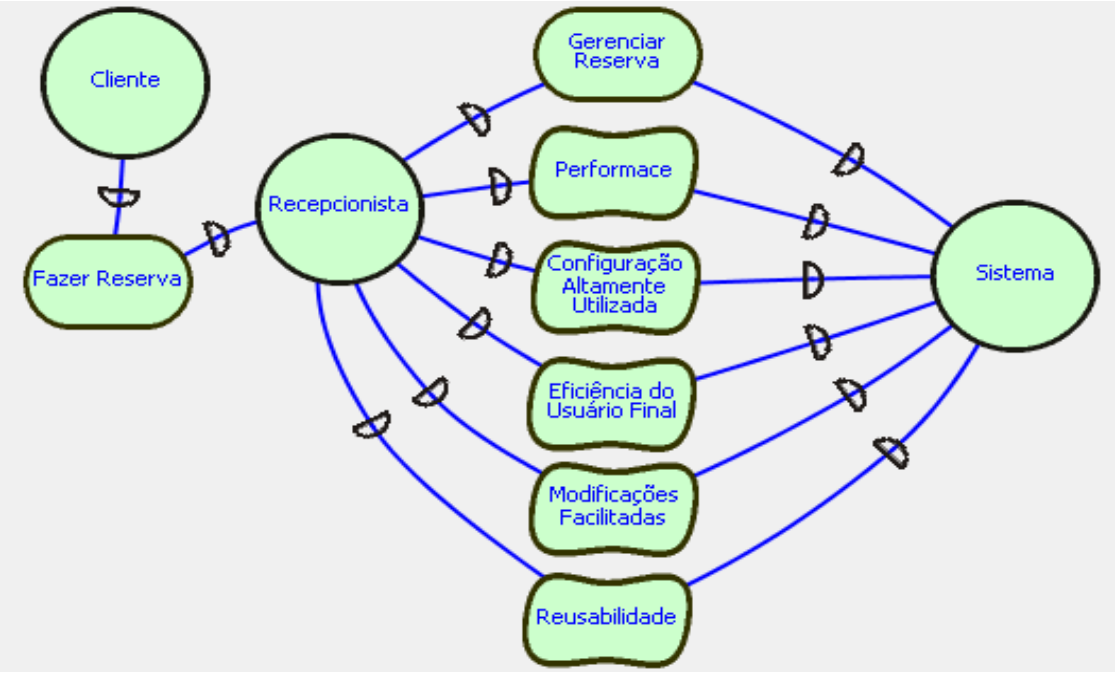


Figura 5.1: Modelo SD (contexto gerenciamento de Reserva) utilizando os *soft-goals* para identificar características de ajuste.

O modelo SD ilustrado foi baseado nas funcionalidades e regras de negócio que devem ser levantadas junto com cliente/usuário. Para as características de ajuste representadas pelos *soft-goals* foram identificadas com base nas características do IFPUG e validadas pelo cliente/usuário, como sugerido na diretriz 4. Dessa forma considerando que ambos os contextos representados se tratam do mesmo sistema essas características serão as mesmas.

Agora o próximo passo é o detalhamento estratégico através da modelagem SR, onde a partir desse modelo é preciso realizar o levantamento dos pesos que as características *soft-goal* referente ao ajuste terão sobre o sistema, bem como já identificar possíveis funções para o processo de medição. Assim é necessário que essa modelagem seja também validada pelo cliente/usuário, proporcionando qualidade para o andamento do processo, ou seja é necessário a elaboração do modelo seguindo as **diretrizes do 2º** passo (6-10). A Figura 5.2 ilustra o modelo SR.

O modelo SR ilustrado traz assim como no exemplo referente ao gerenciamento de hóspede o detalhamento das razões estratégicas envolvidas no ambiente, dependências *soft-goals* para a representação das características de ajuste e todas as informações relevantes ao processo de medição.

Para as dependências de *soft-goals* não há nenhuma mudança, seguem as mesmas utilizadas para a modelagem do SR utilizada no capítulo anterior referente á "hóspede", uma vez que seguem os mesmos parâmetros utilizados para definir-los e definir seus pesos (usuário/cliente).

Já para a questão de identificação de funções ocorreram algumas alterações interessantes e que não ocorreram na modelagem do "hóspede". A priori já é possível identificar que a aplicação responsável pelo gerenciamento de reservas (ator **Sistema Reserva**) é dependente de informações que são de responsabilidade da nossa hipotética aplicação gerenciar hóspede (ator **Sistema Hóspede**), assim levantando ao contexto a certeza da comunicação entre as aplicações, resultando na identificação de um AIE.

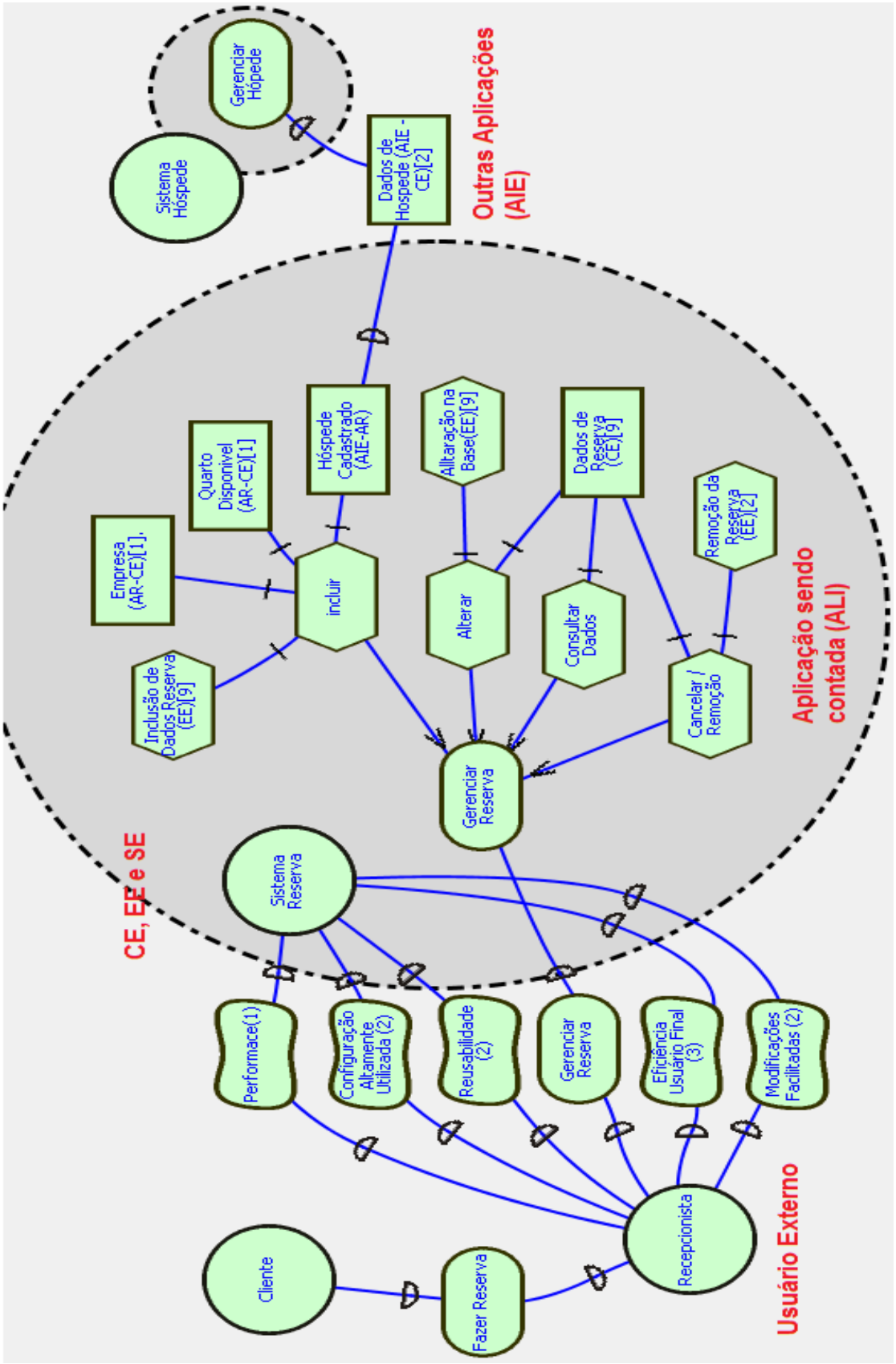


Figura 5.2: Modelo SR para Gerenciamento de Reserva, com informações importantes para o processo de medição.

Esse tipo de conclusão junto com o cliente/usuário é fundamental para este caso em especifico, tanto para as funções identificadas com as *tags* EE, CE, SE, ALI e AIE quanto para a identificação da regra de negócio, onde uma reserva se trata de apenas um único ALI, pois mesmo sendo compostos por dados de "Quartos" e "Empresa" (armazenamento separado em banco) sob a ótica de negócio não faz sentido que esses dados estejam armazenados separadamente, e caso os dados de Hóspede não fosse uma AIE ele também iria compor essa mesma ideia. Logo a nível estratégico e de negócio "reserva" mesmo sendo composto por vários dados fragmentados é considerado apenas um único ALI, e hóspede um AIE, algo que sem a modelagem orientada com ajuda do cliente/usuário (especialista do negócio) seria bem mais difícil de identificar.

Como destacamos no Capítulo 4, referente à integração do *framework i**, temos a possibilidade de validar alguma pendência sobre alguma informação a partir da criação de documentação mais objetiva ou consultar novamente o cliente usuário à medida que for preciso. Assim foi criada uma diretriz para essa possibilidade que seria a diretriz 11. Dessa forma, agora existe a possibilidade de criarmos a "documentação" de forma mais objetiva e direta, norteadas pelo *framework i**.

Vamos simular que ocorre a dificuldade de identificação e validação de algumas informações, como o número de dados manipulados durante o processo (realizar uma reserva), uma abordagem que o IFPUG considera na identificação das funções durante o processo é a validação através de layout de tela, neste contexto para esse caso onde existe a dificuldade na validação de informação seria um bom recurso a ser utilizado para validação junto com o usuário. A figura 5.3 ilustra o layout de tela levantado e "validado" pelo cliente/usuário **(Diretriz 11)**.

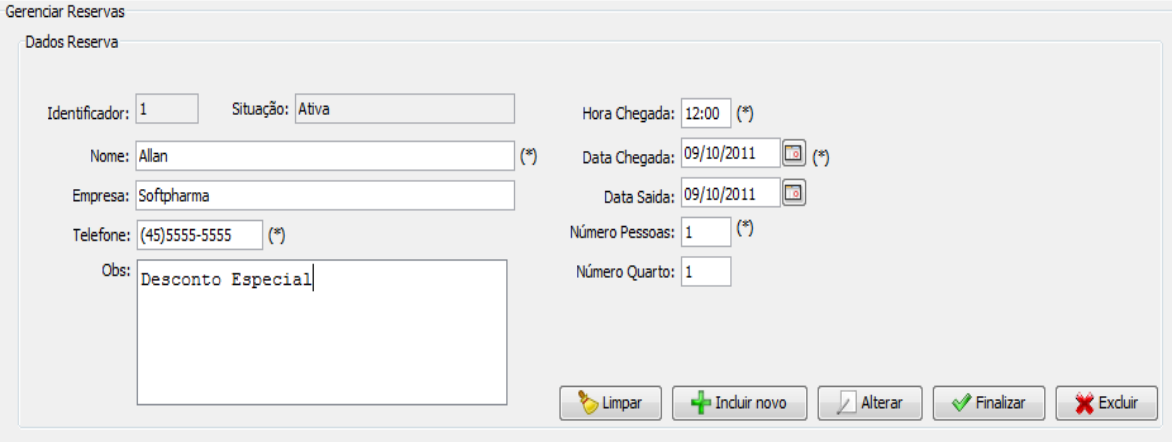


Figura 5.3: *Layout* de Tela , gerenciamento de Reservas.

Com o *layout* de tela (demonstrado na ilustração acima) validado pelo usuário, fica fácil de identificar e quantificar o número de dados manipulados para cada caso. O campo "Nome:" e "Telefone" refere-se ao hóspede responsável pela reserva, e não existe mais nenhuma informação referente a hóspede, ou seja, existe 2 dados manipulados dentro do ALI reserva referente ao AIE hóspede, além de se tratar de um AR para este ALI, portanto representado pelo ator "Dados Hóspede (AIE-AR) [2]".

Também observando o campo "Número Quarto:" é possível identificar que se trata de um AR (Quarto) para o ALI e uma CE referente ao seu retorno, o mesmo ocorre com o campo "Empresa", onde é identificado mais um AR-CE referente à Empresa. Já para os campos "Hora Chegada", "Data Chegada", "Data Saída", "Número Pessoas" e "Obs" são considerados EE específicas da própria reserva, ou seja, também serão contabilizadas como ETD para a ALI. Seguindo esse mesmo raciocínio para as demais funções teremos no fim o modelo SR completo, com um maior número de informações.

Neste contexto já temos uma boa modelagem para iniciarmos o processo de medição. Se ainda não tivéssemos alcançado êxito na modelagem poderíamos ainda recorrer ao levantamento de maiores informações/documentação conforme diretriz 10, contudo como em nosso exemplo isso já está claro, então é a hora de aplicar a APF com o auxilio da modelagem **(Diretriz 12)**:

a) **Determinação do tipo de contagem:** A contagem a ser realizada neste contexto refere-se a um sistema a ser desenvolvido, ou seja, ainda não foi iniciado.

b) Identificação da fronteira da aplicação: Observando a modelagem SR da figura 5.2 é possível identificar facilmente a fronteira de aplicação, uma vez que a mesma é representada pelo círculo de expansão do ator "Sistema Reserva".

c) Contabilizar as funções tipo dados – ALI e AIE: É possível identificar "Reserva" como sendo um ALI, e que o mesmo possui 9 ETD que são os dados referentes a reserva, onde os mesmos foram quantificados no ator "Dados de Reserva[9]", o número de subgrupos de dados identificados foram 4 (Reserva, Hóspede, Quarto, Empresa), portanto esta ALI contém 4 ETR. Também é possível identificar um AIE que seria a aplicação externa representada pelo ator "Sistema Hóspede", uma vez que a aplicação a ser contada depende de informação desta aplicação. O AIE tem 2 ETD que é referente ao dado "nome" e "telefone", que são necessários para a aplicação de gerenciamento de reserva, e um único ETR. Essas informação foi representada na modelagem pelo ator "Dados Hóspede (AIE-CE)[2]". A Tabela 5.1 ilustra o levantamento das **funções tipo dados.**

Tabela 5.1: Levantamento das **funções tipo dado** para o contexto de gerenciamento de reserva.

Arquivos Lógicos Internos	**Elementos Tipo Dado**	**Elementos Tipo Registro**
1 (Reserva)	9 (Dados de Reserva)	4 (subgrupo de dados)
Arquivos de Interface Externa	**Elementos Tipo Dado**	**Elementos Tipo Registro**
1 (aplicação responsável pelas informações de Hóspede).	2 (Dado relevante a aplicação sobre o AIE)	1 (subgrupo de dados)

d) Contabilizar as funções tipo transação - Entradas Externas (EE): Analisando os atores com a *tag* EE do modelo SR são possíveis identificar 3 funções referente a EE. Ambas foram inicialmente levantadas e representadas pelos atores "Inclusão de Dados Reserva(EE)[9]", "Alteração na Base(EE)[9]" e "Remoção da Reserva(EE)[2]". A Inclusão de Dados Reserva(EE)[9] é referente aos dados que são manipulados durante a reserva, "Alteração na Base(EE)[9]" se refere a entrada para futuras alterações, e "Remoção da Reserva(EE)[2]" que para este caso é considerada como 2 dados de manipulação pelo motivo de a exclusão ser realizada pelo nome do hóspede juntamente

com a data de reserva. Lembrando que todo o processo de modelagem foi e deve ser realizado com auxilio do cliente/usuário, pois ao extrair o conhecimento de negócio e abstrair ao máximo torna essa etapa mais rápida e fácil de analisar.

e) **Contabilizar as funções tipo transação - Consultas Externas (CE):** Analisando os atores com a *tag* CE do modelo SR são possíveis identificar 4 funções referente a CE, que foram inicialmente levantadas e representadas pelos atores "Empresa (AR-CE)[1]", "Quarto Disponível(AR-CE)[1]", "Dados Hóspede (AIE-CE)[2]" e "Dados de Reserva(CE)[9]". A CE representada pela Empresa (AR-CE)[1], é referente a empresa para que o hóspede trabalha, e essa informação é providenciada por uma consulta que traz apenas o nome da empresa. A mesma situação ocorre com "Quarto Disponível(AR-CE)[1]" onde o retorno é o número do quarto disponível, para Dados Hóspede (AIE-CE)[2] o retorno é referente a informações de hóspede (nome e telefone). Já para "Dados de Reserva(CE)[9]" o retorno da consulta é todos os dados referentes ao gerenciamento de reserva.

f) **Contabilizar as funções tipo transação - Saídas Externas (SE):** Sem SE para esta aplicação.

A Tabela 5.2 abaixo ilustra as funções tipo transação analisadas para a aplicação.

Tabela 5.2: Levantamento das **funções tipo transação** para a aplicação de gerenciamento de reserva.

Entradas Externas	**Elementos Tipo Dado**	**Arquivos Referenciados**
1 (Inclusão de Dados Reservas)	9 (todos de Reserva)	4 (Hóspede, Empresa, Quarto, Reserva,)
1 (Alteração na Base)	9 (todos de Reserva)	4 (Hóspede, Empresa, Quarto, Reserva,)
1 (Remoção de Reserva)	2 (data e nome)	1 (Hóspedes)
Consultas	**Elementos Tipo Dado**	**Arquivos Referenciados**
1 (Empresa)	1 (nome Empresa)	1 (Empresa)
1 (Quarto)	1(quarto disponível)	1(Quarto)
1(Hospede)	2(nome e telefone)	1(Hospede)
1(Dados Reserva)	9(todos os dados)	4 (Hóspede, Empresa, Quarto, Reserva,)

Com as funções do tipo **dado** e **transação** já identificadas o próximo passo é classificar as mesmas de acordo com suas Tabelas de classificação apresentadas no Capítulo 3, dessa forma consideramos que essa etapa já esteja clara. Neste contexto temos os seguintes resultados, para funções EE: "Alteração na Base" e "Inclusão de Dados Reservas" com complexidade **alta** e "Remoção de Reserva" com **baixa,** para as funções CE: "Empresa", "Quarto" e "Hospede" complexidade **baixa** e "Dados Reserva" **alta,** para o ALI: "Reserva" **baixa** e para o AIE: "aplicação responsável pelas informações de Hóspede" também **baixa.** A Tabela 5.3 ilustra o resultado encontrado para a classificação das funções encontradas juntamente com o PFNA calculado.

Tabela 5.3: Cálculo dos pontos de função **não** ajustados.

Função	**Itens Contados por Complexidade**	**Contribuição**	**Total por Complexidade**	**Total de PFNA da Função**
ALI	1 Baixa 0 Média 0 Alta	x 7 x 10 x 15	7 0 0	7
AIE	1 Baixa 0 Média 0 Alta	x 5 x 7 x 10	0 0 0	5
EE	1 Baixa 0 Média 2 Alta	x 3 x 4 x 6	9 0 0	15
CE	3 Baixa 0 Média 1 Alta	x 3 x 4 x 6	3 0 0	15
TOTAL	**42 pontos de função não ajustados (PFNA)**			

O próximo passo após ter em mãos a contagem dos PFNA é calcular o fator de ajuste de acordo com as características determinadas pela IFPUG mencionadas (Seção 3.1.4). Como já construímos as modelagens utilizadas levando em consideração essas características, e

representando-as em forma de *soft-goal*, facilita o cálculo do fator de ajuste. A Tabela 5.4 ilustra o VFA calculado.

Tabela 5.4: Características retiradas dos *soft-goal*.

Características Gerais do Sistema (*soft-goal*)	**Nível de Influência**
Performance	1
Configuração Altamente utilizada	2
Eficiência do usuário final	3
Reusabilidade	2
Modificação facilitada	2
Total	**Grau de Influência Total = 10**
VFA	**= (10 * 0,01) + 0,65 = 0,75**

Por fim com o VFA encontrado, basta multiplicar o mesmo pelos PFNA resultantes da Tabela 5.4. Assim temos que o resultado da contagem é **32** pontos de função ajustados para a aplicação de gerenciamento de reserva.

5.2 Considerações Finais.

Este capítulo apresentou um estudo de caso demonstrando a execução da proposta de integração do *framework* organizacional *i** ao processo de medição de software. Para demonstração dessa integração foi utilizada outro contexto, gerenciamento de reserva, onde o mesmo trouxe situações novas, não encontradas no exemplo apresentado no capítulo anterior. Neste contexto, permitindo uma maior clareza da aplicação da técnica durante o processo de medição.

Capítulo 6

Considerações Finais e Trabalhos Futuros.

6.1 Conclusões

Iniciamos este trabalho com a proposta de integrar os modelos organizacionais desenvolvidos pelo *framework i** ao processo de medição de *software,* mais especificamente para a técnica de Análise de Pontos por Função. Ao longo do trabalho descrevemos as características das duas técnicas separadamente, focando os benefícios que as mesmas trazem quando utilizadas.

Após o conhecimento dessas técnicas foi proposta a integração, onde foram criados passos e diretrizes para auxiliar no entendimento e execução da proposta. A execução da integração ocorreu, em um primeiro momento, aproveitando do mesmo estudo de caso utilizado para exemplificar o processo de APF, com o intuito de tornar visíveis os benefícios que a integração trouxe ao processo. No segundo momento, a proposta foi aplicada em um novo estudo de caso sobre um contexto diferente, onde com o auxilio dos passos e diretrizes definidos foi apresentada a APF juntamente com o *framework i*.*

Todo esse processo realizado conseguiu demonstrar que a APF é uma técnica realmente dependente de informações referente às funcionalidades requeridas pelo usuário e regras de negócio envolvidas no meio. O processo tradicional de APF não dá qualquer amparo ou recurso para que quem aplique a técnica possa entender melhor esses parâmetros, a única informação que as referências sobre a técnica dão sobre o assunto é que as mesmas são fundamentais para o processo e devem estar claras e dominadas.

Neste contexto este trabalho mostrou que o entendimento desses parâmetros não só podem ser dominadas previamente pela modelagem organizacional, como também é possível utilizar dessa modelagem para o processo de medição. Agora com este tipo de integração perguntas como: “qual a fronteira de aplicação”, como delimitar a mesmas, quais as funcionalidades do usuário, existe uma aplicação externa, como é feita a entrada no sistema, entre outras envolvendo funcionalidades e regras de negócio podem ser esclarecidas previamente no

modelo SD e SR. Sem contar a qualidade visual que a modelagem trouxe para quem está aplicando a medição, onde muitas das informações estão expressas nelas, como foi apresentado neste trabalho, alem é claro dessa modelagem trazer todos os benefícios já conhecidos da ferramenta para o processo seguinte, que seria o de desenvolvimento do sistema.

Contudo é evidente que para o sucesso da integração é indispensável à participação do usuário/cliente durante o processo, ou seja, sem um especialista do negócio envolvido ativamente nas etapas apresentadas o resultado resultante será falho.

6.2 Contribuições

A seguir serão apresentadas as principais contribuições deste trabalho para o processo de medição de *software* (APF).

- **Uma estimativa em um tempo menor:** considerando que com o conhecimento do ambiente organizacional, regras de negócio e identificação de possíveis funções facilita-se a elaboração da documentação necessária e/ou, em alguns casos, torna possível eliminar grande parte desta documentação, além de facilitar/agilizar a análise e classificação das funções durante o processo.
- **Qualidade aceitável:** considerando que o IFPUG afirma que a APF é restritamente uma medição das funcionalidades e regras de negócio envolvidas para o usuário, sem qualquer ligação com a tecnologia e linguagem adotada, ao especificarmos bem as funcionalidades, regras de negócio e o ambiente vivenciado pelo usuário estaríamos identificando funções **estratégicas e essenciais** que devem ser consideradas em um processo de APF.
- **Representação visual:** Com as modificações visuais realizadas no modelo SD e SR, onde os mesmos agora estão com informações relevantes agregadas para o processo de medição, temos ilustrado o ambiente organizacional juntamente com as informações da medição. Neste contexto consegue-se uma qualidade visual interessante e facilitada para tomadas de decisões e análises, substituindo aquele contexto abstrato utilizado anteriormente.
- **Qualidade de Software:** A técnica de APF permite quantificar o tamanho funcional do sistema, de forma que com históricos e experiência consegue-se

utilizar do tamanho encontrado para a aplicação dentro do gerenciamento de escopo, custos e prazos. Já a modelagem organizacional é fundamental para o entendimento organizacional e conseqüentemente para a construção do sistema. Neste contexto a proposta de agilizar o processo de medição dando a opção de uma medição mais ágil juntamente com uma modelagem organizacional, permitiria que empresas de fato utilizassem dessa abordagem, e que assim agregassem qualidade a seu produto.

- **Conhecimento para o andamento do projeto:** caso a medição realizada venha a se tornar um projeto de desenvolvimento de fato, o conhecimento organizacional levantado nesta etapa é de grande valia na construção do mesmo.

6.3 Trabalhos Futuros

A proposta apresentada foi aplicada sobre estudos de casos acadêmicos com o intuito de apresentação da mesma, dessa forma como trabalhos futuros seria interessante que fossem realizados estudos de casos reais, com projetos completos em alguma empresa de desenvolvimento de *software*. Assim os resultados obtidos poderiam ser utilizados para melhorias e adaptações que aperfeiçoariam a proposta apresentada.

Também se pretende realizar estudos sobre a técnica de Pontos por Casos de Uso, uma vez que a mesma se trata de uma derivação da APF, assim o *framework i** poderia agregar algum beneficio a está técnica. Neste contexto, se torna um desafio futuro para que se realizem maiores pesquisas com o intuito de integrar o *framework i** para está técnica, até porque a mesma assim como a APF é muito utilizada no mercado.

Apêndice A

Hotel Real

O Hotel Real atualmente é dirigido pelas famílias Girardello e Vacari, tornando o hotel um ambiente familiar. Por se tratar de duas famílias, a divisão de serviços é feita da seguinte forma: em um determinado dia uma família cuida do hotel, e no outro dia é a outra família que deve cuidar, e assim sucessivamente.

Para ser efetuada a entrada no hotel é realizado um cadastro, caso o hóspede já esteja cadastrado é realizado apenas o registro no livro de diárias. Caso o hóspede não tenha bagagens o pagamento deve ser feito adiantado.

Atualmente todos os dados e serviços são controlados manualmente em livros de registros e cadernos de anotações, como: livro de cadastros, livro de diárias, caderno de reservas e caderno de gastos das empresas. Isso acaba tornando o sistema muito lento e impreciso.

Sendo assim, propõe-se a implantação de um sistema gerenciador hoteleiro que aperfeiçoe todas as operações que envolvam informações/dados do hotel. O sistema proposto será implementado na plataforma Windows com as seguintes características: os usuários farão uso de um computador localizado na recepção para cadastros de hóspedes e empresas, verificação de disponibilidade de quartos, gerenciamento de diárias entre outras ações. O sistema será utilizado por diferentes funcionários, desde recepcionistas até a gerência, mas com níveis de privilégios diferenciados através de *login* e senha.

Apêndice B

Requisitos do Sistema

Para a classificação dos requisitos funcionais e não-funcionais quanto à sua prioridade, foi feita a divisão em três categorias: essenciais, importantes e desejáveis.

a) **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
b) **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
c) **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

B.1 Requisitos Funcionais

Os requisitos funcionais de um sistema são as capacidades que ele irá fornecer aos usuários. A seguir são apresentados os requisitos funcionais do sistema, assim como uma breve descrição dos mesmos.

[RF-01] Incluir Hóspede.
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir incluir um novo hóspede, a partir dos seguintes dados: nome, nacionalidade, CPF, estado civil, sexo, profissão, data de nascimento, data do cadastro, procedência, destino, número do RG, telefone, placa do carro e observações.
[RF-02] Alterar Hóspede
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve permitir alterar os dados cadastrais de um hóspede, através de uma consulta por nome ou CPF.

[RF-03] Consultar Hóspede

Prioridade: Essencial
Solicitante: Recepcionista

Descrição: O sistema deve permitir fazer a consulta de um hóspede através do nome ou CPF do mesmo. Sendo que o resultado da consulta será exibido em tela.

[RF-04] Excluir Hóspede
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve permitir excluir um hóspede a partir de uma consulta por nome ou CPF.

[RF-05] Incluir Empresa
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir incluir uma nova empresa, a partir dos seguintes dados: nome/razão social, endereço, CNPJ, telefones, falar com (responsável), o cargo do mesmo (responsável), inscrição estadual, e-mail, site, e observações.

[RF-06] Alterar Empresa
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve permitir alterar os dados cadastrais de uma empresa, através de uma consulta por nome ou CNPJ.

[RF-07] Consultar Empresa
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir fazer consulta de uma empresa através de nome ou CNPJ. Sendo que o resultado da consulta será exibido em tela.

[RF-08] Excluir Empresa
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve permitir excluir uma empresa a partir de uma consulta por nome ou CNPJ.

[RF-09] Incluir Quarto
Prioridade: Essencial
Solicitante: Gerente
Descrição: O sistema deve permitir incluir um novo quarto, a partir dos seguintes dados: número/identificador, tipo do quarto, número de camas e observações.

[RF-10] Alterar Quarto
Prioridade: Importante
Solicitante: Gerente
Descrição: O sistema deve permitir alterar os dados cadastrais de um quarto, através de uma consulta por número/identificador.

[RF-11] Consultar Quarto
Prioridade: Essencial
Solicitante: Gerente

Descrição: O sistema deve permitir fazer consulta de um quarto através do numero/identificador. Sendo que o resultado da consulta será exibido em tela.
[RF-12] Excluir Quarto
Prioridade: Importante
Solicitante: Gerente
Descrição: O sistema deve permitir excluir um quarto a partir de uma consulta por número/identificador.
[RF-13] Incluir Reserva
Prioridade: Desejável
Solicitante: Recepcionista
Descrição: O sistema deve permitir incluir uma nova reserva, a partir dos seguintes dados: nome, empresa, data prevista para chegada, telefone, hora prevista para chegada, número de pessoas e observações.
[RF-14] Alterar Reserva
Prioridade: Desejável
Solicitante: Recepcionista
Descrição: O sistema deve permitir alterar os dados de uma reserva, através de uma consulta por nome ou data prevista para chegada.
[RF-15] Consultar Reserva
Prioridade: Desejável
Solicitante: Recepcionista
Descrição: O sistema deve permitir fazer consulta de uma reserva através do nome ou data prevista para chegada. Sendo que o resultado da consulta será exibido em tela.
[RF-16] Cancelar Reserva
Prioridade: Desejável
Solicitante: Recepcionista
Descrição: O sistema deve permitir cancelar uma reserva a partir de uma consulta por nome ou data prevista para chegada.
[RF-17] Incluir Diária
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir incluir uma nova diária a partir dos seguintes dados: identificador de uma pessoa já cadastrada data de entrada, data prevista de saída, valor da diária e observações.
[RF-18] Alterar Diária
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir alterar os dados de uma diária.
[RF-19] Consultar Diária
Prioridade: Essencial
Solicitante: Recepcionista

Descrição: O sistema deve permitir fazer consulta de uma diária através da data da mesma. Sendo que o resultado da consulta será exibido em tela.

[RF-20] Cancelar Diária
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir cancelar uma diária.

[RF-21] Finalizar Diária
Prioridade: Essencial
Solicitante: Recepcionista
Descrição: O sistema deve permitir a finalização de uma diária, sendo está, total dos gastos do hóspede, tanto a soma dos valores das diárias em que ele permaneceu no hotel, como os gastos extras na lanchonete.

[RF-22] Incluir Usuário
Prioridade: Essencial
Solicitante: Administrador
Descrição: O sistema deve permitir incluir um novo usuário, a partir dos seguintes dados: nome, *login* nome, senha e a prioridade.

[RF-23] Alterar Usuário
Prioridade: Importante
Solicitante: Administrador
Descrição: O sistema deve permitir alterar os dados cadastrais de um usuário, através de uma consulta por nome.

[RF-24] Consultar Usuário
Prioridade: Importante
Solicitante: Administrador
Descrição: O sistema deve permitir fazer consulta de um usuário através do nome do usuário. Sendo que o resultado da consulta será exibido em tela.

[RF-25] Excluir Usuário
Prioridade: Importante
Solicitante: Administrador
Descrição: O sistema deve permitir excluir um usuário a partir de uma consulta por nome do usuário.

[RF-26] Logar no Sistema
Prioridade: Essencial
Solicitante: Usuário
Descrição: O sistema deve permitir que o usuário faça login no mesmo, através de um *login* e uma senha, sendo que as funcionalidades do sistema serão acessíveis aos usuários de acordo com o seu nível de privilégios.

[RF-27] Gerar Relatório
Prioridade: Desejável
Solicitante: Gerente

Descrição: O sistema deve permitir gerar relatórios específicos, como de novos cadastros de hóspedes realizados em um determinado período, relatórios de gastos de uma empresa já cadastrada em um determinado período, entre outros.

B.2 Requisitos Não-Funcionais

Requisitos não funcionais, ao contrário dos funcionais, não expressam nenhuma função (transformação) a ser implementada em um sistema, eles expressam condições de comportamento e restrições nos serviços do sistema, tais como restrições de tempo, restrições no processo de desenvolvimento, padrões, etc, que devem prevalecer. Segue abaixo os requisitos não funcionais do sistema e uma breve descrição dos mesmos.

Quanto a Segurança:

[RNF-01] Confidencialidade dos dados
Prioridade: Essencial
Solicitante: Gerente
Descrição: O sistema deve garantir a confidencialidade dos dados. Operacionalização: A confidencialidade dos dados será implementada através de uma política de *login* e senha, em que cada usuário poderá acessar dados conforme seu nível de privilégio.
[RNF-02] Integridade dos dados
Prioridade: Essencial
Solicitante: Gerente
Descrição: O sistema deve garantir a integridade dos dados. Operacionalização: A integridade dos dados será mantida através de uma política de armazenamento de dados removidos, que consistirá em ao invés de apagar os dados, apenas desativá-los, marcando-os como inativos.
[RNF-03] Disponibilidade dos dados
Prioridade: Desejável
Solicitante: Gerente
Descrição: O sistema deve garantir a disponibilidade dos dados. Operacionalização: A disponibilidade dos dados será realizada através de backups feitos em tempos programados, além da política de armazenamento de dados removidos.

Quanto a Usabilidade:

[RNF-04] Deve ser fácil de utilizar
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve ser fácil de ser usado e de localizar as operações desejadas. Operacionalização: Para ter agilidade ao acessar as funcionalidades mais utilizadas, haverá teclas de atalho, que também facilitarão o alcance dos menus e funções do sistema ao usuário.

Bem como, as interfaces terão o padrão definido, com o objetivo de uma melhor visualização e entendimento do software. E também terá um manual de ajuda, descrevendo as funcionalidades do sistema.

Quanto a Confiabilidade:

[RNF-05] Confiabilidade dos dados
Prioridade: Importante
Solicitante: Recepcionista
Descrição: O sistema deve garantir a confiabilidade dos dados. Operacionalização: O sistema emitirá mensagens de confirmação da operação realizada, e também fará de tempos em tempos o backup dos dados.

Quanto ao Custo:

[RNF-06] Deve ter um custo baixo
Prioridade: Desejável
Solicitante: Gerente
Descrição: O sistema deve ter um custo baixo de desenvolvimento. Operacionalização: Para o custo de desenvolvimento ser baixo, o sistema irá utilizar apenas ferramentas gratuitas.

Quanto a Performance:

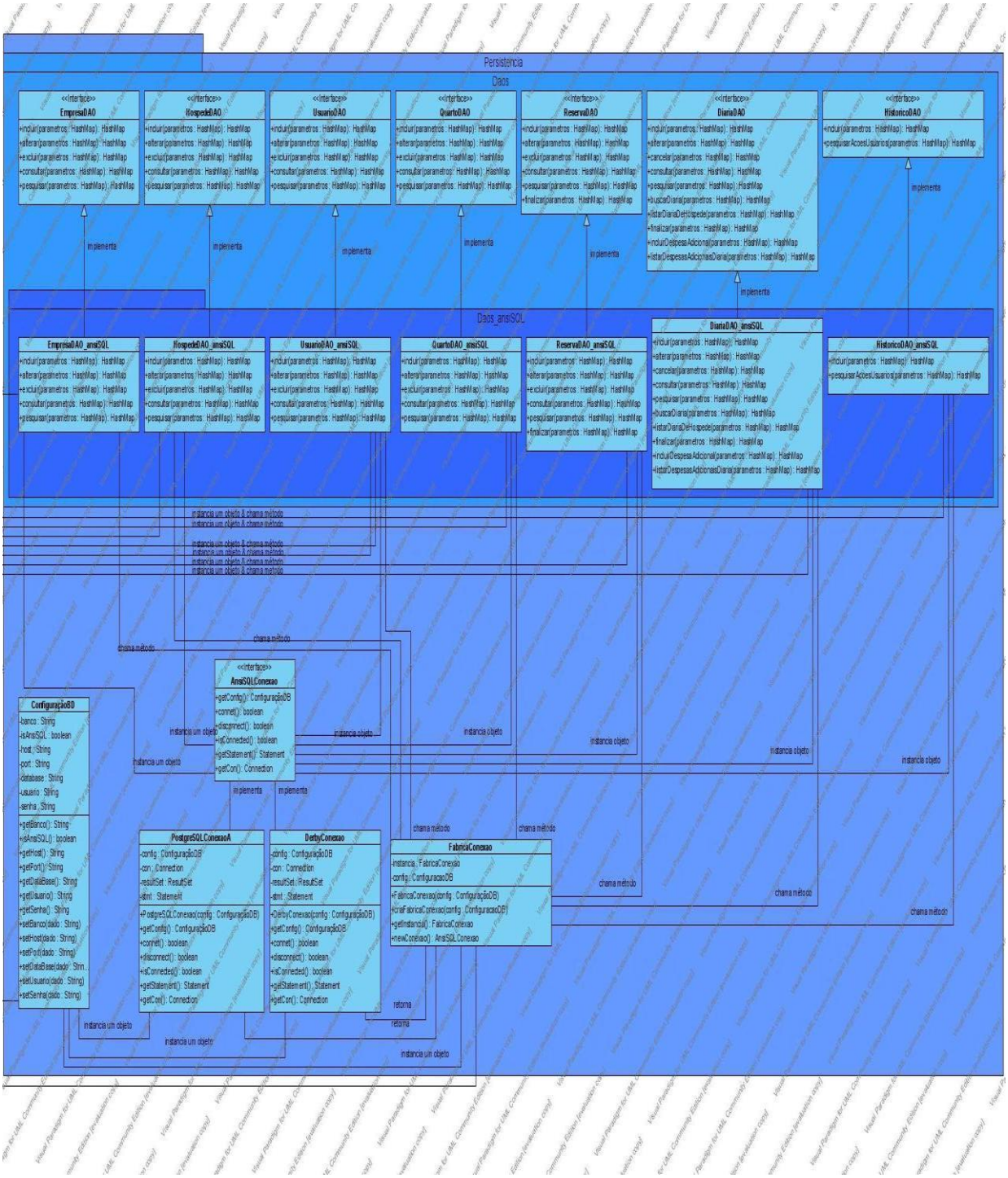
[RNF-07] Configuração do Computador
Prioridade: Importante
Solicitante: Gerente
Descrição: O sistema deve funcionar em uma configuração especifica de máquina. Operacionalização: Para ter uma boa *performance* o sistema deverá rodar em uma máquina com a seguinte configuração: CPU Intel 2,0GHz, 1GB de memória RAM, 40GB de HD, Teclado, Mouse, Monitor 15".

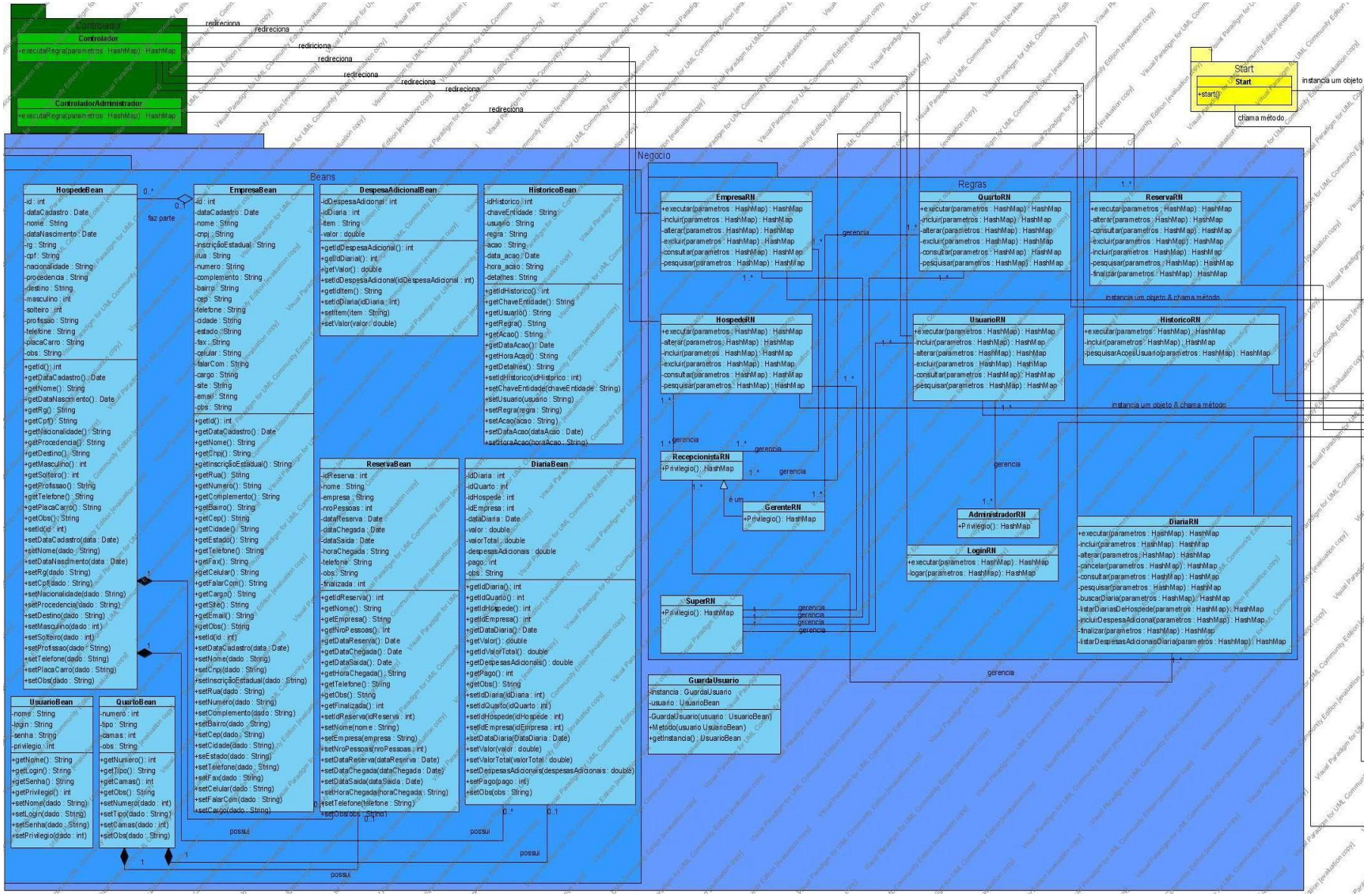
Quanto a Evolução:

[RNF-08] Fácil de Atualizar
Prioridade: Essencial
Solicitante: Gerente
Descrição: O sistema deve garantir que futuramente possam ser atualizado. Operacionalização: O sistema irá utilizar o padrão de desenvolvimento MVC (Modelo, Visão e Controle), para obter uma maior modularidade do software, garantindo que alterações e evoluções no sistema sejam possíveis com uma maior facilidade. E também, utilizará Orientação a Objetos para uma melhor organização e entendimento do código fonte.

Apêndice C

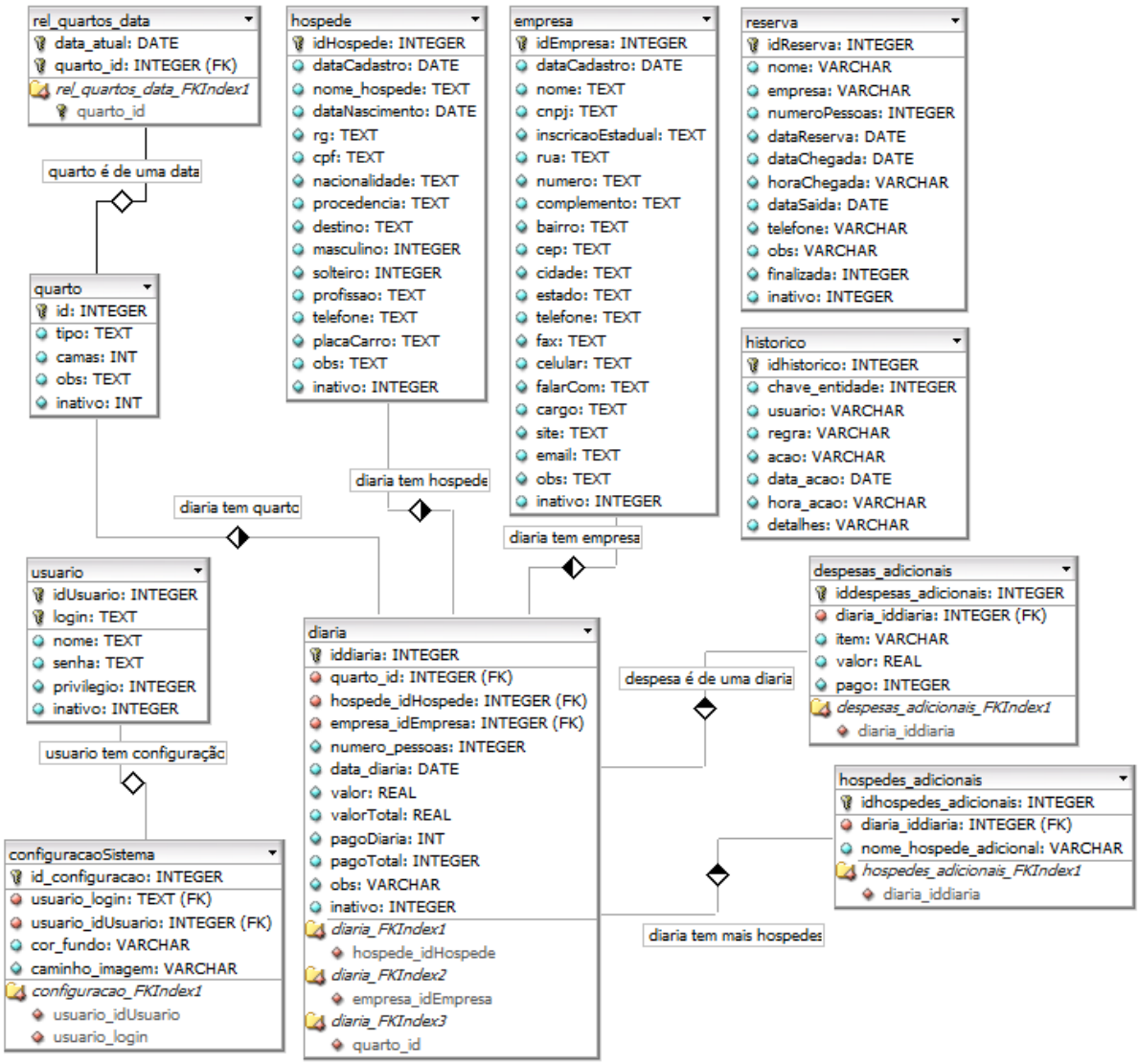
Diagrama de Classes




Controlador
Controlador
+executaRegra(parametros : HashMap) : HashMap
ControladorAdministrador
+executaRegra(parametros : HashMap) : HashMap
redireciona
Start
Start
+start()
instancia um objeto
chama método
Negocio
Beans
HospedeBean
-id : int
-dataCadastro : Date
-nome : String
-dataNascimento : Date
-rg : String
-cpf : String
-nacionalidade : String
-procedencia : String
-destino : String
-masculino : int
-solteiro : int
-profissao : String
-telefone : String
-placaCarro : String
-obs : String
+getId() : int
+getDataCadastro() : Date
+getNome() : String
+getDataNascimento() : Date
+getRg() : String
+getCpf() : String
+getNacionalidade() : String
+getProcedencia() : String
+getDestino() : String
+getMasculino() : int
+getSolteiro() : int
+getProfissao() : String
+getTelefone() : String
+getPlacaCarro() : String
+getObs() : String
+setId(id : int)
+setDataCadastro(data : Date)
+setNome(dado : String)
+setDataNascimento(data : Date)
+setRg(dado : String)
+setCpf(dado : String)
+setNacionalidade(dado : String)
+setProcedencia(dado : String)
+setDestino(dado : String)
+setMasculino(dado : int)
+setSolteiro(dado : int)
+setProfissao(dado : String)
+setTelefone(dado : String)
+setPlacaCarro(dado : String)
+setObs(dado : String)
faz parte
EmpresaBean
-id : int
-dataCadastro : Date
-nome : String
-cnpj : String
-inscriçãoEstadual : String
-rua : String
-numero : String
-complemento : String
-bairro : String
-cep : String
-telefone : String
-cidade : String
-estado : String
-fax : String
-celular : String
-falarCom : String
-cargo : String
-site : String
-email : String
-obs : String
+getId() : int
+getDataCadastro() : Date
+getNome() : String
+getCnpj() : String
+getInscriçãoEstadual() : String
+getRua() : String
+getNumero() : String
+getComplemento() : String
+getBairro() : String
+getCep() : String
+getCidade() : String
+getEstado() : String
+getTelefone() : String
+getFax() : String
+getCelular() : String
+getFalarCom() : String
+getCargo() : String
+getSite() : String
+getEmail() : String
+getObs() : String
+setId(id : int)
+setDataCadastro(data : Date)
+setNome(dado : String)
+setCnpj(dado : String)
+setInscriçãoEstadual(dado : String)
+setRua(dado : String)
+setNumero(dado : String)
+setComplemento(dado : String)
+setBairro(dado : String)
+setCep(dado : String)
+setCidade(dado : String)
+setEstado(dado : String)
+setTelefone(dado : String)
+setFax(dado : String)
+setCelular(dado : String)
+setFalarCom(dado : String)
+setCargo(dado : String)
DespesaAdicionalBean
-idDespesaAdicional : int
-idDiaria : int
-item : String
-valor : double
+getIdDespesaAdicional() : int
+getIdDiaria() : int
+getValor() : double
+setIdDespesaAdicional(idDespesaAdicional : int)
+getItem() : String
+setIdDiaria(idDiaria : int)
+setItem(item : String)
+setValor(valor : double)
HistoricoBean
-idHistorico : int
-chaveEntidade : String
-usuario : String
-regra : String
-acao : String
-data_acao : Date
-hora_acao : String
-detalhes : String
+getIdHistorico() : int
+getChaveEntidade() : String
+getUsuario() : String
+getRegra() : String
+getAcao() : String
+getDataAcao() : Date
+getHoraAcao() : String
+getDetalhes() : String
+setIdHistorico(idHistorico : int)
+setChaveEntidade(chaveEntidade : String)
+setUsuario(usuario : String)
+setRegra(regra : String)
+setAcao(acao : String)
+setDataAcao(dataAcao : Date)
+setHoraAcao(horaAcao : String)
ReservaBean
-idReserva : int
-nome : String
-empresa : String
-nroPessoas : int
-dataReserva : Date
-dataChegada : Date
-dataSaida : Date
-horaChegada : String
-telefone : String
-obs : String
-finalizada : int
+getIdReserva() : int
+getNome() : String
+getEmpresa() : String
+getNroPessoas() : int
+getDataReserva() : Date
+getDataChegada() : Date
+getDataSaida() : Date
+getHoraChegada() : String
+getTelefone() : String
+getObs() : String
+getFinalizada() : int
+setIdReserva(idReserva : int)
+setNome(nome : String)
+setEmpresa(empresa : String)
+setNroPessoas(nroPessoas : int)
+setDataReserva(dataReserva : Date)
+setDataChegada(dataChegada : Date)
+setDataSaida(dataSaida : Date)
+setHoraChegada(horaChegada : String)
+setTelefone(telefone : String)
+setObs(obs : String)
DiariaBean
-idDiaria : int
-idQuarto : int
-idHospede : int
-idEmpresa : int
-dataDiaria : Date
-valor : double
-valorTotal : double
-despesasAdicionais : double
-pago : int
-obs : String
+getIdDiaria() : int
+getIdQuarto() : int
+getIdHospede() : int
+getIdEmpresa() : int
+getDataDiaria() : Date
+getValor() : double
+getIdValorTotal() : double
+getDespesasAdicionais() : double
+getPago() : int
+getObs() : String
+setIdDiaria(idDiaria : int)
+setIdQuarto(idQuarto : int)
+setIdHospede(idHospede : int)
+setIdEmpresa(idEmpresa : int)
+setDataDiaria(DataDiaria : Date)
+setValor(valor : double)
+setValorTotal(valorTotal : double)
+setDespesasAdicionais(despesasAdicionais : double)
+setPago(pago : int)
+setObs(obs : String)
UsuarioBean
-nome : String
-login : String
-senha : String
-privilegio : int
+getNome() : String
+getLogin() : String
+getSenha() : String
+getPrivilegio() : int
+setNome(dado : String)
+setLogin(dado : String)
+setSenha(dado : String)
+setPrivilegio(dado : int)
QuartoBean
-numero : int
-tipo : String
-camas : int
-obs : String
+getNumero() : int
+getTipo() : String
+getCamas() : int
+getObs() : String
+setNumero(dado : int)
+setTipo(dado : String)
+setCamas(dado : int)
+setObs(dado : String)
possui
Regras
EmpresaRN
+executar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-excluir(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
QuartoRN
+executar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-excluir(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
ReservaRN
+executar(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-excluir(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
-finalizar(parametros : HashMap) : HashMap
HospedeRN
+executar(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-excluir(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
UsuarioRN
+executar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-excluir(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
HistoricoRN
+executar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-pesquisarAcoesUsuario(parametros : HashMap) : HashMap
instancia um objeto & chama método
RecepcionistaRN
+Privilegio() : HashMap
é um
GerenteRN
+Privilegio() : HashMap
AdministradorRN
+Privilegio() : HashMap
LoginRN
+executar(parametros : HashMap) : HashMap
-logar(parametros : HashMap) : HashMap
DiariaRN
+executar(parametros : HashMap) : HashMap
-incluir(parametros : HashMap) : HashMap
-alterar(parametros : HashMap) : HashMap
-cancelar(parametros : HashMap) : HashMap
-consultar(parametros : HashMap) : HashMap
-pesquisar(parametros : HashMap) : HashMap
-buscarDiaria(parametros : HashMap) : HashMap
-listarDiariasDeHospede(parametros : HashMap) : HashMap
-incluirDespesaAdicional(parametros : HashMap) : HashMap
-finalizar(parametros : HashMap) : HashMap
-listarDespesasAdicionaisDiaria(parametros : HashMap) : HashMap
SuperRN
+Privilegio() : HashMap
gerencia
GuardaUsuario
-instancia : GuardaUsuario
-usuario : UsuarioBean
-GuardaUsuario(usuario : UsuarioBean)
+Metodo(usuario UsuarioBean)
+getInstancia() : UsuarioBean

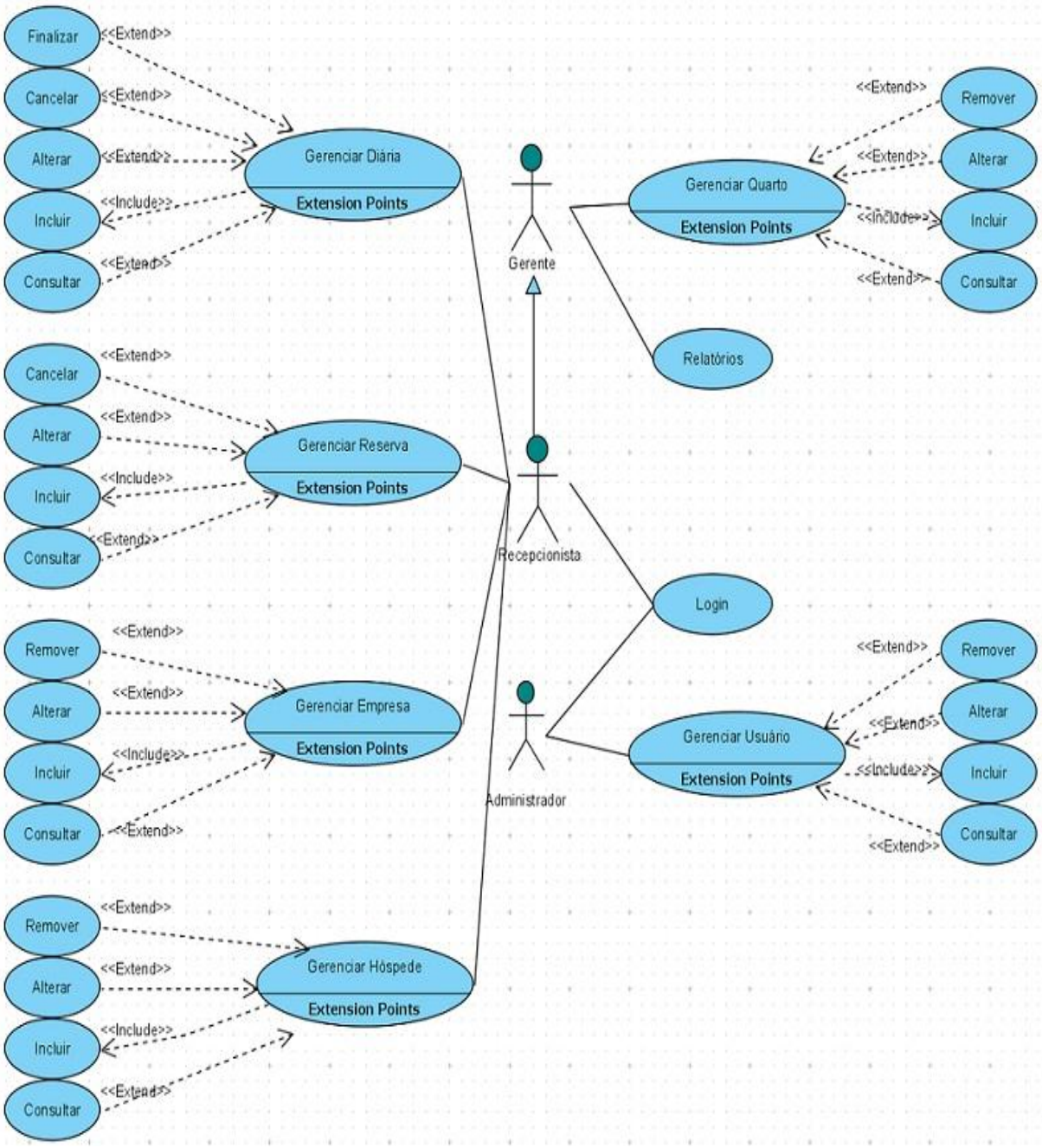
Apêndice D

Diagrama Entidade Relacionamento



Apêndice E

Diagrama de Casos de Uso



Referências Bibliográficas

ALENCAR, F. M. R; SOUZA, F.M; CASTRO, J.F.B. *Modelagem organizacional: análise comparativa das técnicas I*.* Workshop Iberoamericano de Engenharia de Requisitos de Ambiente de Software. San Jose, Costa Rica. 1999.

ANDRADE, E .L. P. *Pontos de Caso de Uso e Pontos de Função na gestão de estimativa de tamanho de projetos de software orientados a objetos*. Dissertação (Dissertação de Mestrado) – UCB Universidade Católica de Brasília. Brasília. 2004.

BARCELLOS, M. P. *Realização de Estimativas utilizando Análise de Pontos de Função.* Universidade Federal do Espírito Santo, Espírito Santo. 2007.

BRAGA. A. *Análise de Pontos de Função.* Infobook. Rio de Janeiro, 1996.

CHICHINELLI, M. *Contribuição da Técnica de Modelagem Organizacional I* ao Processo de Engenharia de Requisitos, com destaque aos Requisitos Não Funcionais*. Dissertação (Dissertação de Mestrado) – Engenharia de Software – Universidade de São Paulo, São Carlos, SP, 2002.

FRANCA, L. P. A; STAA, A. V; LUCENA, C. J. P. *Medição de Software: Uma Solução Baseada na Web.* PUC, Rio de Janeiro, 2005.

GIRARDELLO, A. D; FREDRICH, A. P; SIPPERT, T. A. S. *Sistema Gerenciador de Hotel.* Documentação de Engenharia de Software UNIOESTE Universidade Estadual do Oeste do Paraná, PR, Brasil, 2009.

IFPUG. Function Point Counting Practices Manual. International Function Point Users Group. Janeiro 2005.

MONTEIRO, T, C. *Pontos de Caso de Uso Técnicos (TUCP): Uma Extensão da UCP*. Dissertação (Dissertação de Mestrado) – UNIFOR – Universidade de Fortaleza, Fortaleza, CE, 2005.

PÁDUA, S. I. D, CAZARINI, E. W. *Modelagem Organizacional, Facilitador do Desenvolvimento de Sistemas de Informação*. USP- EESC - Escola de Engenharia de São Carlos, São Carlos, SP, Brasil.

PMBOK. *Um Guia do Conhecimento em Gerenciamento de Projetos.* 4º Edição Project *Managemend Institute* PMI, Atlanta, EUA. 2008.

SANTANDER, V. F. A. *Integrando Modelagem Organizacional com Modelagem Funcional*. Tese (Tese de Doutorado) – Engenharia de Software – Universidade Federal de Pernambuco, Recife, PE, 2002.

SANTOS, E. B. *Uma Proposta de Métricas para Avaliar Modelos i**. Dissertação (Dissertação de Mestrado) – UFPE – Universidade Federal de Pernambuco, Recife, PE, Brasil. Junho 2008.

STANDISH GROUP, CHAOS: *pesquisa sobre o desenvolvimento de software e o panorama da indústria de Tecnologia da Informação na atualidade*, 2001. Disponível em www.standishgroup.com/chaos.html. Consultado na Internet em: 30/03/2011

TAVARES, H. C. A. B.; CARVALHO, A. E. S.; CASTRO, J. F. B. *Medição de Pontos por Função a Partir da Especificação de Requisitos*. Artigo. Serpro – Empresa do Ministério da Fazenda, Universidade Federal de Pernambuco, Recife, PE, Brasil. 2005.

VAZQUEZ, C. E; SIMÕES G. S; ALBERT, R. M. *Análise de Pontos de Função Medição, Estimativas e Gerenciamento de Projetos de Software.* 9ª Edição Revisada. São Paulo: Editora Érica Ltda, 2010.

VIEIRA, E. L. *Uso do Conceito de Passos Obrigatórios para Aprimorar o Processo de Contagem do Método "Pontos de Caso de Uso"*. Dissertação (Dissertação de Mestrado) – USFC – Universidade Federal de Santa Catarina, Florianópolis, SC, 2007

YU. E. S. *Social* Modeling *and i**. Faculdade de Informação. Universidade de Toronto, Canadá, 1995.