SANDRA KAWAMOTO

PROPOSTA DE UM MODELO DE REPRESENTAÇÃO DE *DESIGN RATIONALE* PARA PROJETOS DE SISTEMAS ERP

São Paulo
2007

SANDRA KAWAMOTO

PROPOSTA DE UM MODELO DE REPRESENTAÇÃO DE *DESIGN RATIONALE* PARA PROJETOS DE SISTEMAS ERP

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de concentração: Sistemas Digitais

Orientador: Prof. Dr. Jorge Rady de Almeida Júnior

São Paulo

2007

Dedico este trabalho ao meu orientador

Prof. Jorge Rady

AGRADECIMENTOS

Em especial ao Prof. Jorge Rady pela oportunidade e pela orientação. Obrigada pelo estímulo e por ter acreditado neste trabalho. Sua demonstração de humildade, generosidade e compreensão foi uma lição de vida. Sinto-me honrada por ter sido orientada por uma pessoa tão maravilhosa.

Ao Enrico, por ter me incentivado a ingressar neste mestrado e principalmente por ter me apresentado ao Prof. Rady.

Ao Fabiano, pela sugestão do tema principal deste trabalho. Obrigada por esta contribuição fundamental.

Às professoras Lúcia, Selma, Rosângela, Débora e Renata pelas sugestões e críticas que tanto contribuíram para o desenvolvimento deste trabalho.

Aos colegas e amigos que prontamente atenderam ao meu pedido e me auxiliaram nos questionários durante este trabalho, em especial, Aline, Wally, Pilar, Kotaki, Mauro e Vergari.

A todos da empresa Microsiga (hoje Totvs) pelo aprendizado profissional e pessoal, em especial, Weber, Wilson e Henrique. Obrigada pela compreensão para que este trabalho fosse finalizado. Julio, você é uma pessoa muito especial. Obrigada pela paciência e convívio. Agradeço também às demais pessoas com quem trabalhei diretamente neste período: Enrico, Guilherme, Pérsio, Thays, Fábio, Adriana, Marcus, Eduardo, Juan, Manoel, Carol, entre outros.

A todos os colegas e professores do ICMC-USP, que me mostraram os primeiros passos no mundo da informática.

Aos colegas e professores dos departamentos de Engenharia Elétrica e Psicologia da USP, pela oportunidade de participar de aulas tão enriquecedoras durante o mestrado.

Hilton, agradeço pelo incentivo, pelas palavras e pelos elogios.

Rose, obrigada pelo incentivo, pela atenção, pelas chineladas (risos) e pelos ensinamentos. Admiro sua força e determinação.

Léo, obrigada pela amizade e companhia. Como uma só pessoa pode me fazer lembrar de tantas palavras nobres? Sensibilidade, serenidade, justiça, doação e luz. Sentirei saudades!

À Aline, pela disponibilidade, amizade e carinho. Obrigada pela companhia nas aulas de street, step, taekwondo, natação, hidro, piano, tricô, maquiagem, etc.

Thays, obrigada pela confiança, ajuda, incentivo, amizade, carinho e sensibilidade. Desculpe-me pelas vezes que não pude estar 100% ao seu lado.

Ao Alex, pelas conversas, pelo apoio, por ter contribuído para o meu crescimento como pessoa e por ter me ajudado a conhecer tantas coisas novas.

Graça, meu anjo da guarda, minha secretária e amiga. Obrigada por estar sempre pronta a me ajudar.

D. Therezinha, obrigada pela torcida, pelo cuidado, pelo apoio e atenção. Sem a sua colaboração, este trabalho não teria sido finalizado.

Sr. Loro, obrigada por suas lições de generosidade, paciência, amor e sabedoria.

Bó, obrigada pelo incentivo e pelos conselhos. Wal, obrigada por ser uma pessoa tão atenciosa, prestativa e bem-humorada. Ao casal, obrigada pelo carinho com o Matheus. Agradeço em especial pelos passeios que renderam horas preciosas de pesquisa.

Luciane, obrigada pelo carinho, por estar sempre pronta a me ouvir e a me ajudar. Reginaldo, obrigada pela amizade e convívio. Tenho vocês como referência de sensatez, compreensão, cumplicidade, maturidade e equilíbrio. Julia, minha afilhada querida, sua ternura me encanta! Obrigada por nossos longos papos.

Aos meus pais, que não mediram esforços para que eu pudesse me dedicar aos estudos. Obrigada por hoje eu ter condições de passar por mais esta etapa do meu aperfeiçoamento. Amo vocês!

Ao Matheus, por ter me mostrado o que realmente vale a pena na vida. Por me contagiar com tanta energia, curiosidade, ternura e carinho. Por exalar emoção. Desculpe-me pelas vezes que estive ausente para trabalhar nesta pesquisa.

E, finalmente, agradeço a todos que colaboraram direta ou indiretamente para a realização deste trabalho.

“Não sabendo que era impossível,

ele foi lá e fez"

(Jean Cocteau, dramaturgo francês)

RESUMO

Atenção especial tem sido dada às atividades de documentação e suporte a projetos em Engenharia de Software, principalmente quando estão relacionadas a sistemas complexos e equipes distribuídas. Geralmente apenas se registram as informações relativas às decisões finais de determinada fase. Não são documentadas as razões de cada decisão e nem as alternativas que foram consideradas e descartadas. A captura e recuperação, de maneira eficiente, deste tipo de informação é a finalidade do estudo do *Design Rationale*. O registro destas informações pode facilitar a manutenção, o reuso e também a própria fase de elaboração do projeto, na medida em que possibilita um melhor entendimento do sistema, por meio da disseminação de conhecimento, comunicação e integração entre a equipe de projeto. A questão é quando e como capturar estas informações sem muita interferência junto às atividades usuais dos projetistas, a fim de que os benefícios possam superar os custos envolvidos nesta tarefa. O presente trabalho tem como objetivo mostrar com razões plausíveis a grande utilidade da aplicação da técnica de *Design Rationale* em projetos de sistemas ERP, propondo um novo modelo de representação das decisões de projeto para estes sistemas.

Palavras-chave: *Design Rationale.* Modelos de representação. *Enterprise Resource Planning.* ERP.

ABSTRACT

Special attention has been given to documentation and support activities in Software Engineering design, mainly when they are related to complex systems and distributed teams. Usually, information related to the final decisions of each phase is registered. However, the reasons of each decision and the alternatives that were considered and discarded are not documented. Capture and recovery of this type of information, in an efficient way, are the purposes of the Design Rationale study. Recording this information can facilitates maintenance, reuse and even the design phase, providing a better understanding of the system, by knowledge spread, communication and integration among the project team. The main concern is when and how to capture this information with low interference in designers' usual activities, so that benefits can overcome costs involved in this task. The purpose of the present work is to show with plausible reasons the great usefulness of the application of Design Rationale technique in ERP systems design, proposing a new representation model for recording design decisions in these systems.

Keywords: Design Rationale. Representation Models. Enterprise Resource Planning. ERP.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

APS	*Advanced Planning System*
BPM	*Business Process Management*
CRM	*Customer Relationship Management*
DME	*Device Modeling Environment*
DR	*Design Rationale*
DRL	*Decision Representation Language*
DW	*Data Warehouse*
EDN	*Electronic Design Notebook*
ERP	*Enterprise Resource Planning*
ES	Engenharia de Software
gIBIS	*Graphical Issue Based Information System*
IBIS	*Issue Based Information System*
KBDS	*Knowledge-Based Design System*
MRP	*Material Requirement Planning*
MRP-II	*Manufacturing Resource Planning*
OMS	*Order Management System*
OP	Ordens de serviços
PHI	*Procedural Hierarchy of Issues* - modelo extensão do IBIS
PPAP	Processo de Aprovação de Peça de Produção
QOC	*Questions, Options and Criteria*
REMAP	*Representation and Maintenance of Process Knowledge* – modelo extensão do IBIS
SAC	Serviço de Atendimento ao Consumidor
SFA	*Sales Force Automation* - Automação da Força de Vendas
TFA	*Technical Force Automation* - Automação da Força Técnica
TI	Tecnologia da Informação
TWR	*Tower* - Torre de Controle
WMS	*Warehouse Management System* - Gerenciamento de Armazéns

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação tem como tema central a proposta de um novo modelo de representação do *Design Rationale para* projetos de sistemas computacionais. A proposta apresentada é aplicada a um projeto de desenvolvimento de um software para ERP (*Enterprise Resource Planning*).

Neste capítulo, apresenta-se o objetivo do trabalho, a justificativa da escolha do tema, trabalhos correlatos e finalmente a organização da dissertação.

1.1 Contextualização

A área de informática continua evoluindo em ritmo acelerado. Dutoit et al. (2006) afirma que as novas tecnologias (orientadas a *web*, componentes, padrões de aplicação, etc.) e os modelos de processo (ágil, orientados a risco, orientados a modelo) refletem os desafios da engenharia de software nos dias atuais: construção de software mais complexo em equipes distribuídas, mais rápidas e a custos menores.

Entretanto, complementa Dutoit et al. (2006), a ênfase em tecnologias e em modelos de processo encobre o fato de que engenharia de software é, principalmente, uma atividade baseada em pessoas e que o sucesso de um projeto ou produto depende das decisões tomadas durante todo o seu projeto.

Normalmente, a documentação padrão de projetos contém a descrição do projeto final, isto é, as decisões finais que foram tomadas. O *Design Rationale* engloba não apenas as decisões, mas também as razões que deram suporte a cada decisão, incluindo suas justificativas e alternativas

consideradas (LEE, 1996). O *Design Rationale* fornece um auxílio para revisão, manutenção, documentação, avaliação e aprendizado do projeto.

O *Design Rationale* é especialmente importante para projetos de software. Em geral, o software sofre várias alterações durante o seu ciclo de desenvolvimento, não só para realizar correções como também para alterar ou incorporar novos requisitos. A fase de manutenção de software é muito custosa e torna-se mais complexa se não for feita pela equipe de arquitetos originais. Esta equipe pode até nem estar disponível nesta fase. O software geralmente é manipulado por equipes e pessoas distintas e apenas parte delas participa efetivamente de todo processo de análise de projeto. Outra particularidade de um projeto de software é a existência de várias soluções para um mesmo problema. Por todas estas características, o *Design Rationale* tem grande potencial para ser uma tecnologia que agregue valor ao processo de desenvolvimento de software.

1.2 Objetivo

O primeiro objetivo desta dissertação é o de mostrar a grande utilidade que a aplicação da técnica de *Design Rationale* (DR) representa em projetos de sistemas ERP, que se constituem em um tipo de sistema especialmente sensível a tirar proveito de decisões tomadas em sistemas já desenvolvidos.

Outro grande objetivo deste trabalho é propor um novo modelo de representação do *Design Rationale*, a ser aplicado em projetos de sistemas ERP. Esse novo modelo visa cobrir alguns aspectos deficientes de três modelos estudados do *Design Rationale*, fornecendo uma ferramenta de documentação e análise mais poderosa.

A proposta apresentada tem como base a descrição e comparação de três modelos de *Design Rationale*: QOC (*Questions, Options and Criteria*), IBIS (*Issue Based Information System*) e DRL (*Decision Representation Language*). Estes são os principais modelos considerados nas pesquisas sobre o assunto (DUTOIT et al., 2006; ROSSI, 2000; SHUM, 1991a; STUMPF, 1997). Além da revisão bibliográfica, a proposta foi construída tendo como base a análise de casos reais de decisões de projeto de sistemas ERP.

1.3 Justificativa

A opção pela escolha do *Design Rationale* como conceito básico deste trabalho, justifica-se pelo fato de que a sua utilização pode proporcionar grandes benefícios ao processo de desenvolvimento de software (SHUM; HAMMOND, 1994; FISCHER et al., 1995; CONKLIN; BURGESS-YAKEMOVIC, 1995). Entretanto, na prática ainda não houve adesão significativa à sua utilização (BOSCH, 2004). No início das pesquisas, muitos esperavam que sua aplicação prática pudesse, rapidamente, ser difundida. Não houve, no entanto, a previsão de quão difícil seria definir abordagens e sistemas que fossem utilizados com sucesso em projetos reais (DUTOIT et al., 2006). O *Design Rationale* ainda não é utilizado adequadamente em casos reais e praticamente não há casos de captura de informações, fornecendo pouca oportunidade para investigar o problema da captura do *Design Rationale* na prática.

Foram escolhidos sistemas ERP porque praticamente todas as características específicas deste tipo de aplicação são favoráveis à utilização do *Design Rationale*.

Segundo Dutoit et al. (2006), após mais de 35 anos de pesquisa em *Design Rationale*, algumas questões básicas continuam sem solução, incluindo:

1. Como capturar as justificativas de decisões de projeto (*rationales)*, isto é, como extrair e armazenar as informações;
2. Qual a melhor forma de representar as justificativas das decisões de projeto;
3. Como formalizar estas informações, isto é, como transformar a informação na forma de representação desejada;
4. Como as informações gravadas podem ser utilizadas;
5. Quais são as potenciais barreiras à captura, representação, formalização e uso.

Este trabalho contribui principalmente para a pesquisa da questão 2, pois analisa os modelos de representação do *Design Rationale* propondo um modelo de representação mais adequado a sistemas ERP.

Atualmente, ainda não existe um padrão de representação do *Design Rationale* e muitas pesquisas estão sendo realizadas neste sentido.

1.4 Trabalhos Correlatos

A pesquisa bibliográfica identificou diversos trabalhos sobre sistemas ERP, a maioria investigando as dificuldades de implantação do software (GODLA, 1999; BINGI; SHARMA; MARKUS, 2000; BROWN, 2004; BREHM; HEINZL; MARKUS, 2004). O trabalho de mestrado “Sistemas Integrados de Gestão Empresarial: Estudos de Casos de Implementação de Sistemas ERP” (SOUZA; ZWICKER, 2000) descreve e analisa como ocorrem os processos de decisão, seleção, implementação e utilização de sistemas ERP e realiza

um estudo empírico de natureza qualitativa a fim de analisar benefícios e dificuldades da utilização destes sistemas.

Similarmente, vários trabalhos foram realizados sobre os modelos de representação do *Design Rationale* e ferramentas de suporte, como, por exemplo, Shum (1991b), Garcia, Howard e Stefik (1993), Francisco (2004), Burge (2005) e Lara (2005).

Os trabalhos mais próximos encontrados na literatura apresentam a aplicação de um modelo de *Design Rationale* a um tipo específico de sistema. O trabalho de doutorado de Paiva descrito em um artigo (PAIVA; FORTES, 2005) pesquisa o uso de *Design Rationale* em projetos de softwares acadêmicos, com o intuito de garantir a evolução destes projetos.

O trabalho proposto nesta dissertação difere dos estudados, tendo em vista não ter sido identificado em nenhum deles, a utilização de *Design Rationale* em sistemas ERP.

1.5 Organização do trabalho

No capítulo 2 apresenta-se um estudo sobre projetos de software, com ênfase em sistemas *Enterprise Resource Planning* (ERP).

No capítulo 3 descreve-se a teoria básica que envolve o conceito de *Design Rationale*, contemplando sua definição e formas de obtenção, captura e uso das informações importantes. Ao final, apresentam-se três modelos de representação do *Design Rationale*, realizando-se uma análise comparativa entre eles.

No capítulo 4 apresenta-se a proposta desta dissertação, ou seja, analisa-se a utilização do *Design Rationale* em ambientes de desenvolvimento de sistemas *Enterprise Resource Planning* (ERP), apresentando uma proposta de modelo de representação mais adequado a este tipo de sistema.

Finalmente, no capítulo 5 apresentam-se as conclusões finais desta dissertação, bem como possíveis trabalhos a serem desenvolvidos no futuro, como conseqüência dos estudos aqui realizados.

2 ERP – ENTERPRISE RESOURCE PLANNING

Enterprise Resource Planning (ERP) é o processo pelo qual uma empresa gerencia e integra as partes importantes de seu negócio (DICTIONARY.COM). Um sistema de informação de gerenciamento de ERP integra áreas como planejamento, compras, estoque, vendas, financeiro e recursos humanos. Tipicamente, cada departamento tem seu próprio sistema otimizado para as tarefas daquela área em particular. Com o ERP, cada departamento continua tendo seu próprio sistema, mas eles podem se comunicar e trocar informações mais facilmente com o restante da empresa.

O termo ERP refere-se ao gerenciamento de informação que tem a finalidade de melhorar os serviços ao cliente, aumentar a produtividade e gerenciar custos e estoques de maneira efetiva. Um sistema integrado de informação que trata as diferentes áreas de negócio como uma unidade coesa, pode auxiliar a organização a melhorar a produtividade e qualidade, tornando-a competitiva (MADU; KUEI, 2004).

O termo ERP é geralmente utilizado no contexto de software. À medida que esta idéia de gerenciamento e integração das diversas áreas da empresa tornou-se mais popular, várias aplicações de software foram desenvolvidas para auxiliar as empresas a implementar ERP em sua organização.

Este capítulo tem como objetivo apresentar uma descrição de ERP no contexto de software, contexto este em que recebem a denominação de “sistemas ERP”. Inicialmente é apresentada a definição da expressão, seguida de um histórico de sua evolução, descrição dos processos e funcionalidades que compõem o sistema, ciclo de vida e finalmente benefícios e dificuldades da utilização de um ERP.

2.1 Sistemas ERP

Um sistema ERP é um sistema de informação integrado, normalmente adquirido na forma de pacotes comerciais, com o objetivo de dar suporte à maioria das operações de uma empresa (SOUZA; ZWICKER, 2000a). Procura atender a requisitos genéricos do maior número possível de empresas, incorporando modelos de processos de negócio obtidos pela experiência acumulada de fornecedores, consultorias e pesquisa em processos de *benchmarking*. A integração é viabilizada pelo compartilhamento de informações entre os diversos módulos, armazenadas em um banco de dados centralizado.

No GLOSSÁRIO DE TERMOS DA QUALIDADE (2005), sistema ERP é definido como um sistema de gestão baseado na análise do contexto organizacional, visando facilitar o fluxo de informações entre todos os departamentos da empresa e suas atividades, tais como fabricação, logística, finanças e recursos humanos. Substitui o software departamental por um programa unificado que proporciona uma visão única da empresa, em tempo real.

Os sistemas ERP têm como característica o fato de serem, na verdade, pacotes, ou seja, programas desenvolvidos por empresas de software independentes, vendidos a empresas que as utilizam. Estes pacotes são projetados para suprir necessidades gerais de um ramo de negócio e não necessidades específicas de uma empresa em particular (BREHM; HEINZL; MARKUS, 2001). Adotando pacotes padronizados, as organizações podem reduzir, consideravelmente, os custos, riscos e demora do desenvolvimento de um software sob medida, além de se beneficiar dos serviços de suporte fornecido pelos vendedores e consultores do sistema.

Para Skok e Legge (2001), um sistema ERP constitui-se em mais do que o simples uso de um software pré-desenvolvido. Representa uma iniciativa de

gerenciamento de mudança que engloba a revisão de processos por toda a organização, demandando um gerenciamento minucioso.

Embora os sistemas ERP tenham sofrido grandes mudanças e se popularizado na última década, relativamente pouco se tem escrito sobre o assunto. Os trabalhos existentes têm seu foco, em sua maioria, no gerenciamento e implantação de projetos (BROWN; VESSEY, 1999; BASU; LEDERER, 2004; BINGI; SHARMA; GODLA, 1999; KIRCHMER, 1998; MARKUS; TANIS, 2000).

2.2 Histórico

Na década de 60, já existiam os sistemas de produção, cujo foco era somente o controle de estoque e manutenção automatizada de depósitos. O propósito dos sistemas de controle de estoques era, simplesmente, refletir a quantidade de produto disponível.

Na década de 70, apareceram os sistemas de planejamento de requisição de material (*Material Requirement Planning* - MRP). A partir do cronograma de finalização de um conjunto de produtos principais, o sistema de planejamento de requisição de material (MRP) utilizava dados do estoque atual para calcular a requisição de material necessário à construção destes produtos, sugerindo um cronograma de reabastecimento do estoque. O sistema assegurava a disponibilidade de estoque suficiente.

Os sistemas de planejamento de recursos de produção (*Manufacturing Resource Planning* - MRP-II) substituíram os sistemas MRP na década de 80. Foram projetados para gerenciar as ordens de produção, o planejamento da produção e o estoque (MARKUS; TANIS, 2000). Seu objetivo era fornecer técnicas adicionais para planejar cada um dos aspectos de uma

empresa de produção, desde o planejamento operacional mais básico até o planejamento financeiro e a recolocação de funcionários. Além disso, o sistema tinha a capacidade de criar relatórios. A Figura 1 mostra a representação de MRP e MRP-II feita por Scott (1994).

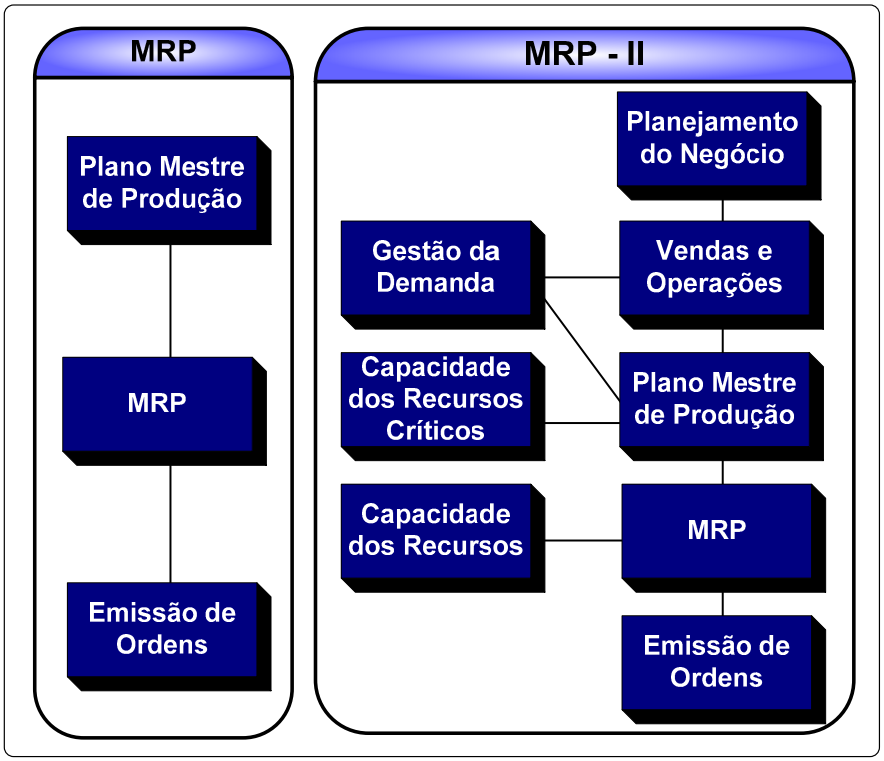

Figura 1 - MRP e MRP-II – Adaptado de Scott (1994)

A maioria dos autores considera os sistemas ERP como uma extensão ou evolução dos sistemas MRP-II (CORRÊA; GIANESI; CAON, 1997; CUNHA, 1998; HEHN, 1999). Eles englobam outros negócios além da produção como, por exemplo, distribuição e serviços. Os sistemas ERP integrados tornaram-se conhecidos no início da década de 90 (VOGT, 2002). A Figura 2 mostra a representação do foco de um ERP, adaptado de Scott (1994). Através de uma base de dados única e centralizada, diretores, acionistas, funcionários, clientes e fornecedores têm acesso aos principais processos e informações de uma empresa: vendas e distribuição, apoio a serviços, gerenciamento de recursos humanos, gerenciamento de materiais, manufatura, finanças e controladoria.

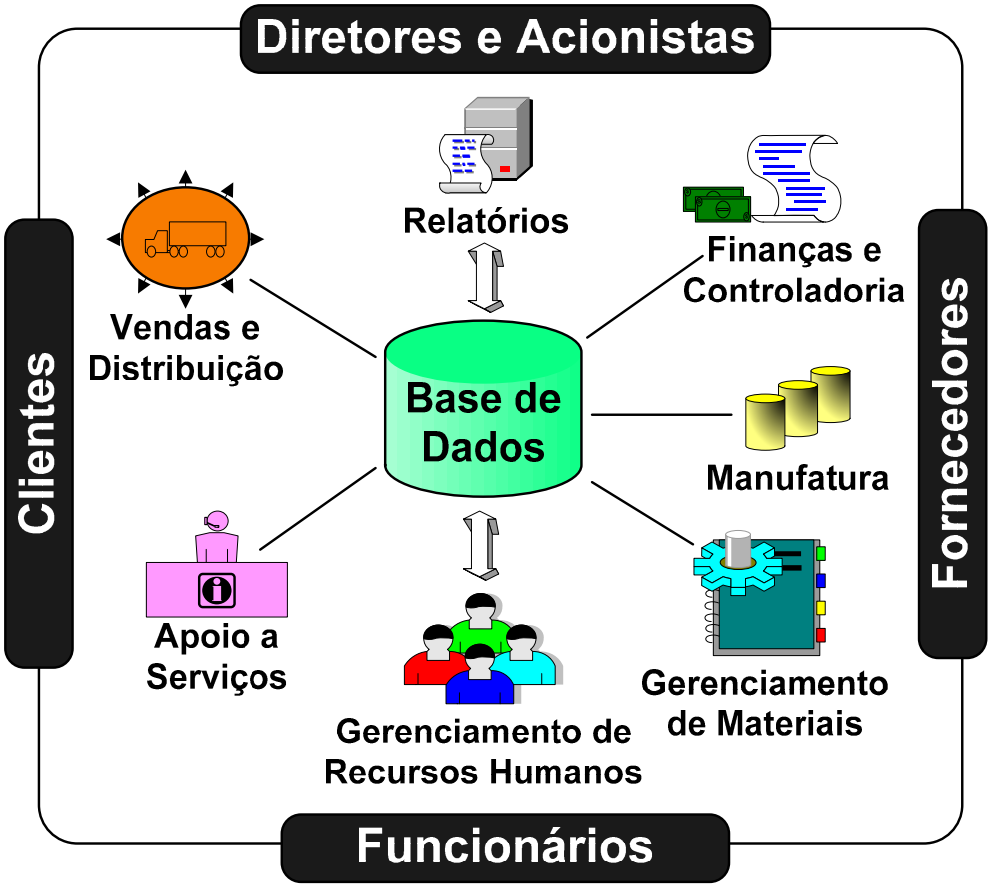

Figura 2 - ERP – Adaptado de Scott (1994)

A idéia de se ter um sistema complexo que integrasse diversas áreas de uma empresa foi sendo estudada nas décadas seguintes. O Quadro 1 traz um resumo desta evolução.

Os primeiros sistemas ERP rodavam em mainframes. O sistema SAP R/2 é um exemplo. Todas as aplicações e o banco de dados rodavam em um computador único e centralizado, de alto desempenho. Aplicações clientes remotas tinham que utilizar uma conexão de resposta automática (*dial-in connection*) temporária ou uma linha permanente alugada para acessar o sistema. As dificuldades de configuração e altos custos de manutenção fizeram com que muitas empresas tivessem resistência em utilizar a tecnologia de mainframe. Além disso, empresas com grande número de escritórios espalhados fisicamente apresentaram desvantagens, já que um

sistema mainframe centralizado, com número elevado de conexões remotas, logo se tornou um gargalo no desempenho.

Década	Nome	Termo em inglês	Principais objetivos
60	Sistemas de produção	*Manufacturing*	➢ Controle de estoque ➢ Manutenção automatizada do depósito
70	Sistemas de planejamento de requisição de material	*Material Requirement Planning* (MRP)	➢ Cálculo automático da requisição de material necessário à construção de produtos
80	Sistemas de planejamento de recursos de produção	*Manufacturing Resource Planning* (MRP II)	➢ Planejamento da empresa desde os aspectos operacionais mais básicos até o planejamento financeiro e a recolocação de funcionários ➢ Criação de relatórios
90	Sistemas integrados de gestão	*Enterprise Resource Planning* (ERP)	➢ Planejamento de outras áreas da empresa como, por exemplo, distribuição e serviços

Quadro 1 - Evolução dos sistemas de produção aos sistemas integrados de gestão

A chegada dos sistemas cliente/servidor prometia melhorar esta situação. Seu sistema distribuído era particularmente apropriado para acesso remoto. Adicionalmente, estes sistemas tinham, potencialmente, maior escalabilidade que os sistemas em mainframe, visto que o número de servidores pode ser alterado, praticamente, sem limite. No caso de um mainframe, não há um meio de aumentar a capacidade computacional se o servidor não estiver mais suportando a carga de trabalho. Para Vogt (2002), foi a tecnologia cliente/servidor que impulsionou os sistemas ERP na última década. O sistema R/3 da SAP, Corpore da RM e Protheus da Microsiga são exemplos de sistemas ERP baseados na tecnologia cliente/servidor.

2.3 Processos e Funcionalidades

Conforme apresentado no item anterior, os primeiros sistemas de gestão tinham o foco no controle de estoque e requisição de material. Com o passar do tempo, outras áreas das empresas foram também automatizadas: financeira, recursos humanos, documentação, contabilidade, planejamento e controle orçamentário, *call center*, CRM (*Customer Relationship Management*), entre outros.

Logo se verificou a necessidade da criação de funcionalidades específicas a determinados ramos de negócio, que apresentavam características bem específicas, diferentes do ERP tradicional. A quantidade de personalização necessária para atender a estes segmentos era significativa. Assim, os sistemas ERP passaram a contemplar módulos ou processos específicos para a gestão educacional, hospitalar, concessionárias automotivas, distribuição e logísticas, etc.

A Figura 3 mostra os os processos e funcionalidades da versão atual do sistema ERP desenvolvido em uma empresa nacional. Estas informações foram coletadas na página da Internet da empresa (MICROSIGA SOFTWARE S/A).

A seguir, é apresentada uma breve descrição de cada funcionalidade.

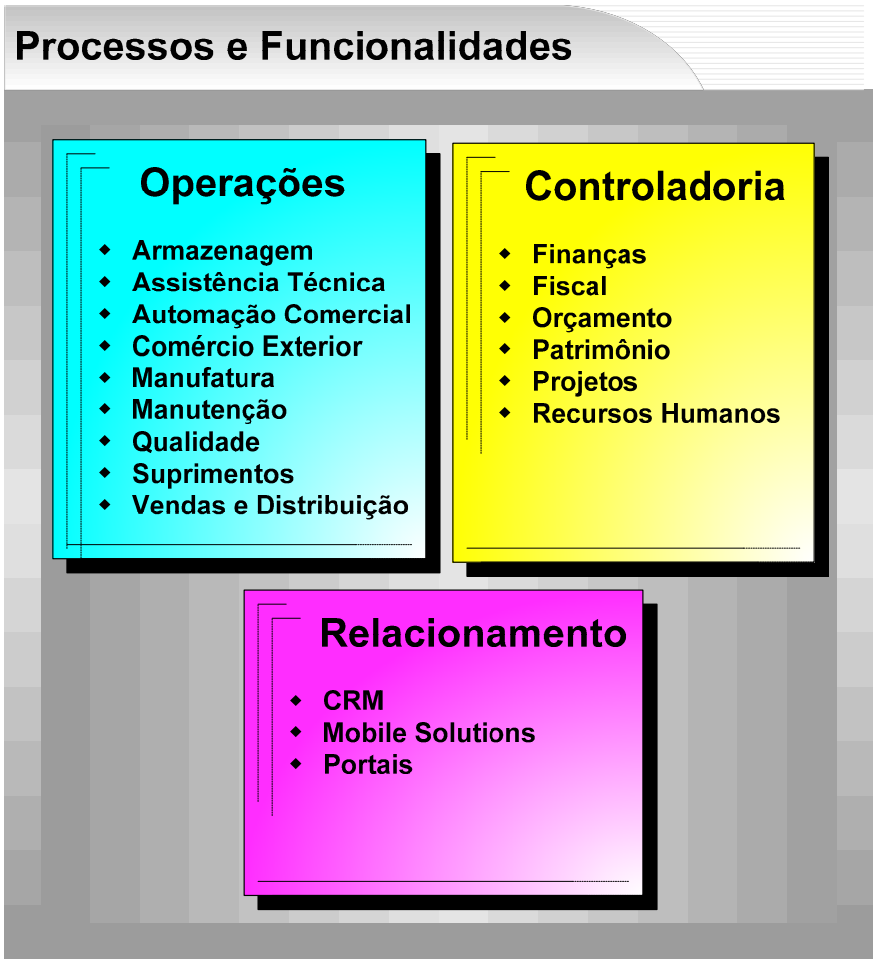

Figura 3 - Processos e Funcionalidades de um Sistema ERP

2.3.1 Operações

Os quesitos abrangidos por esta área de operações compreendem: gestão dos processos de venda, estoque, manutenção, vendas, entre outros, a seguir descritos:

- **Armazenagem**: gestão de armazéns e controle do fluxo e da movimentação de materiais, apresentando informações detalhadas de saldos em estoque e oferecendo suporte para o processo de formação de preços e análise gerencial de custos;
- **Assistência Técnica**: controle aos processos de atendimento e prestação de serviços;

- **Automação Comercial**: gestão comercial que engloba desde controles operacionais relacionados à frente da loja – caixas e atendimentos – até a gestão financeira, de estoques e compras;
- **Comércio Exterior**: gestão dos processos de importação e exportação, incluindo o controle e registro de laudos técnicos, a emissão dos documentos pertinentes ao processo, o acompanhamento das etapas, etc.
- **Manufatura**: gestão de estoques e custos, através do controle do fluxo e da movimentação de materiais, apresentando informações detalhadas sobre saldos em estoque e oferecendo suporte para o processo de formação de preços, prazos de entrega e análise gerencial de custos;
- **Manutenção**: gestão da manutenção de bens, recursos produtivos e ativos da corporação, sejam equipamentos, ferramentas, prédios, veículos, etc., prevendo o gerenciamento de disponibilidade de itens de estoque, ferramentas, mão-de-obra, especialidades e terceiros;
- **Qualidade**: identificação e controle de não-conformidades, bem como as ações de melhoria; identificação de possíveis melhorias nos processos, produtos e projetos; controle dos clientes internos, externos e de fornecedores; manutenção de históricos para melhoria contínua e revisão do fluxo do processo;
- **Suprimentos**: gestão, por meio dos processos de logística, desde a entrada de pedidos de clientes até a entrega do produto no seu destino final, envolvendo aí o relacionamento entre documentos, matérias-primas, equipamentos, informações, insumos, pessoas, meios de transporte, organizações, tempo, etc.
- **Vendas e Distribuição**: processos operacionais e de controle que permitem monitorar a venda, desde a fase de prospecção até a entrega do produto.

2.3.2 Controladoria

Os quesitos abrangidos por esta área de controladoria compreendem: planejamento e controle orçamentário, controle do patrimônio, recursos humanos e gestão financeira e fiscal, a seguir descritos:

- **Finanças**: gestão administrativa que possibilita o acompanhamento de todos os eventos e recursos da empresa, desde o processo de pagamentos e recebimentos até o monitoramento do fluxo de caixa;
- **Fiscal**: gestão de carga tributária e das obrigações fiscais das corporações, incluindo o cálculo de tributos e a geração de relatórios legais;
- **Orçamento**: planejamento, simulação e controle de todo ciclo orçamentário, desde a sua criação até o confronto com os valores realizados, objetivando um nível de desempenho nos negócios;
- **Patrimônio**: controle do ativo permanente da empresa, ou seja, aplicações de recursos feitas em caráter permanente, ora representando bens adquiridos para uso da empresa como veículos, móveis e utensílios, ora representando aplicações de recursos na compra de ações ou quotas de outras empresas de caráter permanente, ou ainda, representando aplicações de recursos em despesas que devem onerar os resultados de vários exercícios. Inclui também o controle de apólices de seguros dos bens, o cálculo de depreciação e correção e inventário;
- **Projetos**: controle integrado de custos, prazos e do trabalho efetivamente realizado no decorrer de um determinado projeto, servindo de suporte aos usuários de diversos níveis da empresa;
- **Recursos Humanos**: controle de avaliações de desempenho, cargos e salários, folha de pagamento, medicina e segurança do trabalho, ponto eletrônico, recrutamento e seleção de pessoal e treinamentos.

2.3.3 Relacionamento

Os quesitos abrangidos por esta área de relacionamento compreendem o gerenciamento das trocas de informações com os clientes. Engloba qualquer modo de obtenção de informações sobre os produtos e serviços prestados, como, por exemplo, Serviço de Atendimento ao Consumidor (SAC) e ouvidoria. Inclui também as soluções móveis e os portais, conforme descrito a seguir:

- **CRM** (*Customer Relationship Management*): gestão de pesquisas, telecobrança, telemarketing, SAC, ouvidoria, *ombudsman* e "lealdade" dos clientes;
- **Mobile Solutions**: solução para ser utilizada por dispositivos móveis; útil tanto para o profissional que está em campo como para o gerente que fica no escritório, permitindo que cada um deles analise os dados, atualize as oportunidades e adote as melhores práticas;
- **Portais**: disponibilização de informações e serviços na WEB, como, por exemplo, informações sobre clientes, fornecedores, funcionários, vendedores, projetos, etc.

O sistema pode ser utilizado em conjunto com ferramentas complementares para setores que requerem processos específicos.

2.4 Ciclo de Vida

O ciclo de vida representa as diversas etapas pelas quais um projeto de desenvolvimento e utilização de sistemas de informação passa. Em sistemas tradicionais, o ciclo de vida inclui as etapas de levantamento de requisitos do

sistema, definição de escopo do projeto, análise de alternativas, projeto do sistema, codificação, testes e manutenção.

O ciclo de vida de pacotes comerciais deve ser considerado de maneira diferente dos modelos de ciclo de vida tradicionais, pois não se trata, efetivamente, de um desenvolvimento interno de sistemas proprietários, e sim, da aquisição e adaptação de um sistema desenvolvido por outra empresa com o objetivo de atender diversas empresas. Por exemplo, a fase de levantamento de requisitos difere em muito da mesma fase no ciclo tradicional.

Por sua vez, os sistemas ERP apresentam diferenças em seu ciclo de vida em relação aos outros pacotes comerciais, principalmente no que se refere à sua abrangência funcional e à integração entre seus diversos módulos. O escopo dos pacotes ERP é muito abrangente. São pacotes completos que requerem mais conhecimento, esforço e habilidade para adaptá-los às características de uma organização em particular.

Souza e Zwicker (2000b) propõem um modelo de ciclo de vida de sistemas ERP, construído com base nas características do ciclo de vida de pacotes comerciais, nas características específicas dos sistemas ERP e em uma revisão da literatura a respeito da seleção, implementação e utilização de sistemas ERP. O modelo, ilustrado na Figura 4, considera as seguintes etapas: decisão e seleção, implementação e utilização. Na fase de decisão e seleção, levam-se em consideração as funcionalidades e arquitetura técnica do produto, custos, serviço e suporte pós-venda, situação financeira, visão e tecnologia do futuro. A fase de implantação corresponde ao processo pelo qual os módulos do sistema são colocados em funcionamento na empresa. Compreende a parametrização dos módulos, migração de dados e treinamento dos usuários. Fatores importantes na implantação são: experiência dos usuários com sistemas e conhecimento prévio sobre as

discrepâncias entre o sistema e a empresa; comprometimento da alta direção; envolvimento das áreas usuárias e da tecnologia.

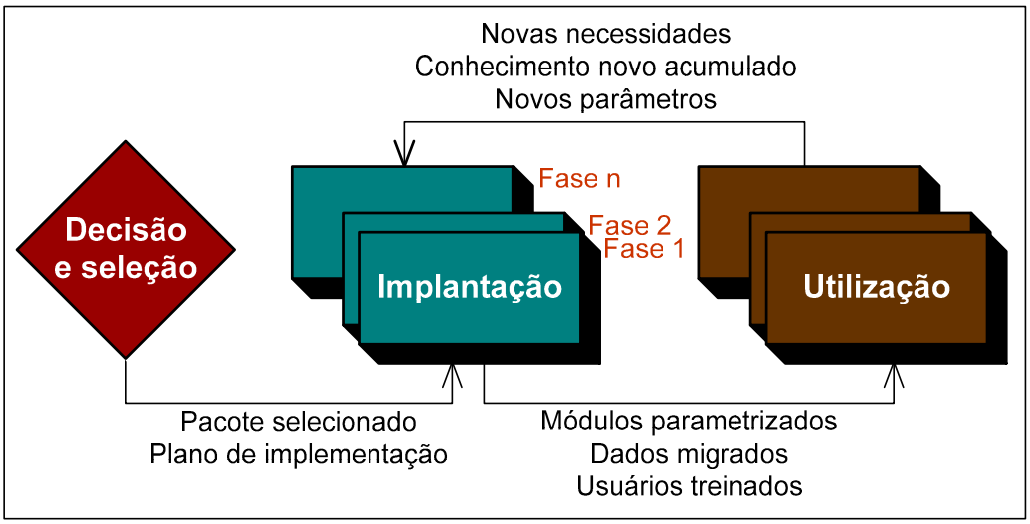

Figura 4 - Ciclo de vida de sistemas ERP- Adaptado de Souza e Zwicker (2000b)

Na fase de utilização, o sistema entra em operação na empresa. Podem surgir novas necessidades ou acerto do que foi feito, voltando-se à fase de implementação até que o sistema esteja de acordo com projeto definido no contrato.

2.5 Benefícios e Dificuldades

A implantação de um sistema ERP é complexa tanto do ponto de vista técnico como de mudança organizacional. Ela deve ser muito bem planejada e gerenciada a fim de minimizar as dificuldades enfrentadas, principalmente no início das atividades. Há um alto custo inicial a ser investido e o período de transição, geralmente, demanda um esforço extra da equipe de tecnologia da informação da empresa. Apesar disso, as vantagens que a utilização de um sistema ERP proporciona parecem compensar estas dificuldades iniciais.

2.5.1 Benefícios

Os sistemas ERP são ferramentas poderosas para a integração de dados e informações em uma empresa, proporcionando vantagens competitivas às organizações.

Em termos gerais, os benefícios da utilização de um sistema integrado para gestão de uma empresa são:

- Redução de tempo: redução do tempo necessário para a realização de processos chave do negócio como, por exemplo, entrega de produtos, fechamento financeiro, entrada de informação de preço, verificação de crédito e recuperação de informações para tomada de decisões (DAVENPORT, 2000).
- Melhor gerenciamento financeiro: melhor gerenciamento da empresa como um todo, principalmente da área financeira, obtendo melhor controle e visão da empresa.
- Novas formas de interação com clientes e fornecedores: com um sistema integrado, há maior facilidade para o oferecimento de outras opções de contato com clientes e fornecedores: comércio eletrônico, vendas através de dispositivos móveis, operações via Internet, etc.
- Aumento da competitividade da empresa (HOLLAND; LIGHT; KAWALEK, 1999): oferecendo uma variedade maior de serviços, com tempo reduzido e possuindo um gerenciamento eficaz da empresa, aumenta-se o poder competitivo da mesma.
- Redução do risco de desenvolvimento e implantação (KELLY; HOLLAND; LIGHT, 1999): a tentativa de construir um sistema próprio pode ser muito arriscada, por se tratar de um sistema muito complexo e grande. Além disso, comprando-se um sistema conhecido no

mercado é possível usufruir toda uma infra-estrutura de *help-desk*, documentação, atualizações, etc.

- Maior conhecimento de processos implícitos: realizando os processos através de um sistema integrado, proporciona um melhor entendimento dos processos chave e de regras de decisão. Estas informações ficam documentadas no sistema, facilitando o entendimento do processo de trabalho por funcionários novos (DAVENPORT, 2000).

A adoção de um sistema integrado por si só não garante todos estes benefícios. O sucesso vai depender de vários fatores, principalmente qualidade do software, equipe de implantação, colaboração dos patrocinadores e da equipe de informática da empresa.

2.5.2 Dificuldades

O principal problema dos sistemas ERP para a maioria dos autores é a grande dificuldade de implantação, que pode levar até três anos para ser finalizada (SOUZA; ZWICKER, 2000b). Esta dificuldade é conseqüência da necessidade de introdução de mudanças organizacionais profundas. As empresas, geralmente orientadas a uma visão hierárquica e departamental, são obrigadas a adaptarem-se a uma visão orientada a processos, com atividades que integram os departamentos. Algumas vezes, a mudança é até mais radical. As empresas são obrigadas a mudar seus procedimentos para adaptar-se às funcionalidades dos pacotes.

As respostas observadas em processos de mudanças variam de uma empresa para outra, pois elas estão intimamente ligadas à cultura organizacional de cada empresa. Por isso, há a necessidade de conhecer os aspectos da organização. As decisões estratégicas tomadas são frutos dos

pressupostos e valores da empresa, partes integrantes da sua cultura organizacional. Segundo Seldin, Ferruccio e Caulliraux (2003), a pouca importância dada a este aspecto pela literatura acadêmica, pelos consultores e pelos responsáveis pela implantação de sistemas integrados acaba por se tornar um dos principais fatores de fracasso das implantações. Um caso de estudo descrito pelos mesmos autores (SELDIN; FERRUCCIO; CAULLIRAUX, 2003) apresenta o conflito de poder causado pela implantação de um sistema integrado de gestão. A disponibilidade em larga escala dos dados de uma organização pode causar um sentimento de perda do que os autores chamaram de “posse da informação” pelos funcionários envolvidos nas tarefas afetadas pela introdução da nova tecnologia.

Complexidade é outra característica da implantação de um software integrado de gestão. Por isso, são fatores críticos para a implementação de sistemas ERP o comprometimento da alta direção em encarar o gerenciamento do projeto como algo crítico, o comprometimento dos gerentes envolvidos pelos resultados, a passagem de responsabilidades sobre o sucesso do projeto para as áreas usuárias, o treinamento e a comunicação. Por estarem ligados não só ao aspecto tecnológico, mas principalmente ao aspecto humano, estes fatores são dificilmente satisfeitos em sua totalidade.

Holland, Light e Gibson (1999) afirmam que o problema da implantação de um sistema ERP inclui dimensões estratégicas, organizacionais e técnicas.

Zhang et al. chamam atenção para a importância do envolvimento do usuário. A implantação de um sistema integrado representa uma ameaça na visão da maioria dos usuários, causada pelo controle feito sobre o seu trabalho. O envolvimento dos usuários restaura ou melhora a percepção de controle através da participação em todo plano de projeto, desde o estágio de definição das necessidades sistêmicas da empresa até o processo de implantação do software. Envolvendo os usuários no estágio de definição

das necessidades pode-se diminuir sua resistência ao sistema ERP, pois as pessoas ficam com a sensação que elas que escolheram o produto e que tomaram a decisão.

A habilidade de gerenciamento de projeto, de gerenciamento de mudanças e o conhecimento a respeito do pacote, em geral, não estão disponíveis nas empresas (LOZINSKY, 1996). Conseqüentemente, há a necessidade de utilização de uma consultoria externa para a implantação. Uma das dificuldades das empresas de consultoria e dos vendedores de sistemas ERP é o baixo número de analistas competentes durante a fase de implantação. Isto provavelmente foi causado pela rápida expansão do mercado ERP. Scheer e Habermann (2000) acreditam que para dar suporte adequado à implantação de um sistema ERP os analistas precisam estar capacitados em pelo menos três dimensões. Primeiramente, eles devem ter conhecimento sobre sistemas integrados. Em segundo lugar, eles devem estar familiarizados com a tecnologia e saber como ela pode ajudar o negócio. Finalmente, os analistas precisam ter um entendimento das práticas do negócio. Acredita-se que seja muito difícil e dispendioso contratar pessoas com capacidades nestas três dimensões. O sucesso ou fracasso de um projeto depende em grande parte de como as empresas lidam com este desafio.

Bowersox et al. atribuem as falhas dos sistemas ERP ao gerenciamento e controle de projeto não satisfatórios, especificação de objetivos incompleta, falta de comunicação e complexidade do projeto subestimada. Estes problemas afetam qualquer implantação de software, mas eles se tornam mais graves quanto maior e mais integrada é a aplicação como um todo (HOLLAND; LIGHT; GIBSON, 1999).

Scheer e Habermann (2000) comentam que a necessidade de espaço de armazenamento de dados muito grande, requisitos de rede e despesas extras com treinamento são freqüentemente mencionados como problemas

dos sistemas ERP. Entretanto, a quantidade de reengenharia do processo de negócio e as tarefas envolvidas no processo de implantação do sistema são as maiores razões de descontentamento com o sistema ERP.

Como qualquer mudança, esta exige dos envolvidos comprometimento, flexibilidade e adaptação. Miller (2001) afirma que pessoas são sempre elementos chave em qualquer processo de melhoria. Portanto, métodos para auxiliar a equipe no aprendizado de uma tecnologia ou processo são extremamente importantes.

Por ter como característica grande generalidade e enorme complexidade, os sistemas ERP tornam-se propensos a apresentar erros e baixo desempenho, com alto custo de manutenção e muitas dificuldades na implantação (VOGT, 2002).

Há vários estudos sobre falhas na implantação de sistemas ERP. *Design Rationale* pode ser muito útil para minimizar os problemas desta fase (VOGT, 2002).

3 DESIGN RATIONALE

Os projetos de software orientados a artefato ainda são encontrados na prática. Neste paradigma, a ênfase está na geração e rastreamento dos artefatos de projeto intermediários (requisitos, especificações, protótipos e documentação) que conduzirão ao sistema final propriamente dito. Mas, o processo de desenvolvimento destes artefatos fica implícito, escondido nas reuniões, cadernos de anotações, e-mails ou na memória dos programadores (SHUM, 1994). Conseqüentemente, estas informações não ficam disponíveis e o que é pior, sua recuperação pode nem ser possível no momento que for necessário.

Nas novas tecnologias (orientação a *web*, componentes e modelos de aplicação) e nos modelos de processos mais novos (ágil, orientado a risco, orientado a modelo), verifica-se problema semelhante. Estas novas tecnologias e modelos de processos refletem o desafio à engenharia de software, atualmente: construir softwares mais complexos em equipes distribuídas, de maneira mais rápida e a custos inferiores. Entretanto, a ênfase em tecnologia e modelos de processo muitas vezes deixa em segundo plano o fato de que a engenharia de software é, essencialmente, uma atividade baseada em pessoas e que o sucesso de um projeto ou produto depende das decisões tomadas durante a engenharia (DUTOIT et al., 2006).

Geralmente, registram-se apenas as informações relativas às decisões finais de determinada fase, pois os analistas consideram onerosa (em termos de tempo) a tarefa de documentar cada uma das alternativas investigadas. Entretanto, registrando-se e recuperando-se estas informações de maneira adequada, podem-se obter várias vantagens ao longo do projeto: melhor conhecimento do projeto, maior facilidade de manutenção e comunicação da equipe de projeto, melhor possibilidade de reuso, maior facilidade de

integração de um novo membro à equipe e também diminuiria a possibilidade de se implementar uma tentativa de opção já descartada no passado. Estes itens são de muito interesse das empresas de informática, visto que uma das principais preocupações é a disseminação de conhecimento, comunicação e integração entre as pessoas.

À primeira vista, estas vantagens não deixam dúvidas de que vale a pena incorporar a captura destas informações ao processo de desenvolvimento de software. A grande questão é como capturar e recuperar estas informações de maneira eficiente e com o mínimo de interferência às atividades do processo usual. Neste ponto é que entra o estudo do *Design Rationale*, cuja idéia básica está na linha da captura e recuperação deste tipo de informação.

O Design rationale tem sido explorado desde 1970 em vários domínios de aplicações. Na década de 80, engenheiros de software começaram a adaptar as primeiras abordagens às suas necessidades. Atualmente, estas abordagens não chegam a oferecer suporte nem à metade da totalidade das atividades do projeto (DUTOIT et al., 2006).

Neste capítulo, são descritos os principais conceitos relacionados a *Design Rationale*. Inicia-se com a apresentação da própria definição do termo, seguida da descrição das formas de obtenção do *Design Rationale*, sua utilidade e formas de representação, captura e uso das informações. Ao final do capítulo, são apresentados três modelos de representação do *Design Rationale*.

3.1 Definição

Há vários trabalhos relacionados à análise de projeto e gerenciamento de decisão que citam o termo *Design Rationale* para descrever o registro de

informações que capturam a razão dos projetistas terem tomado decisões específicas durante a elaboração do projeto. Entretanto, esta atividade não é limitada à fase de elaboração do projeto, mas sim aplicada a todas as fases do desenvolvimento de software. Como muitas pesquisas sobre *rationale* tiveram, no passado, seu foco em atividades de projeto, o termo *Design Rationale* é o que predominou e também é o que se utiliza mais freqüentemente (SAUER, 2003).

Design Rationale refere-se à documentação das alternativas, escolhas e decisões realizadas durante o processo de desenvolvimento de um projeto, bem como das razões de se ter tomado determinado rumo. Vários pesquisadores já apresentaram suas definições. A seguir, são apresentadas algumas delas.

De acordo com Gruber e Russel (1991), *Design Rationales* consistem em explicações dos relacionamentos entre estrutura, comportamento e funcionalidade de artefatos, podendo-se citar como uma determinada estrutura implementa uma funcionalidade, ou como um certo comportamento se forma através de uma estrutura. Eles também explicam o processo de tomada de decisão.

Para Souza et al. (1998), *Design Rationale* é uma representação da documentação que contém as razões e argumentos referentes a um artefato específico. Inclui tanto as razões de uma decisão de projeto como a justificativa para elas, as alternativas consideradas e que foram descartadas, alterações avaliadas e os argumentos que levaram à decisão.

Design Rationale é a informação que explica porque um artefato é estruturado da maneira que é e tem o comportamento que tem (CONKLIN; BURGESS-YAKEMOVIC, 1995).

Segundo Hu et al. (2000), *Design Rationale* é a explicação da razão de um artefato ou alguma parte dele ter sido projetado de determinada maneira. Inclui as considerações, raciocínios, alterações e tomadas de decisão do projeto de um artefato. Estas informações podem ser valiosas e até críticas a várias pessoas que lidam com o artefato.

Em (BURGE; BROWN, 2000), *Design Rationale* é definido como as decisões tomadas durante a fase de análise de projeto e as razões que levaram a tais decisões.

Cada definição fornece um enfoque próprio ao assunto, mas em termos gerais, refere-se a uma definição semelhante. *Design Rationale* está relacionado às informações sobre as razões, considerações e as justificativas de uma tomada de decisão, bem como sobre as alternativas consideradas e que foram, eventualmente, descartadas. Este trabalho utiliza o termo *Design Rationale* com este significado.

3.2 Principais Abordagens

A obtenção de informações, visando à técnica do *Design Rationale*, ou seja, a captura e recuperação das informações necessárias podem ser realizadas de várias maneiras. A utilização de uma determinada abordagem não exclui a existência de outras, ou seja, é possível utilizar, em um mesmo sistema, alternativas distintas de abordagens ao *Design Rationale*. As principais abordagens (BURGE; BROWN, 1998; GARCIA; HOWARD; STEFIK, 1994) são descritas a seguir.

3.2.1 Baseada em Argumentação

Consiste na gravação dos argumentos e questões que tenham surgido durante um projeto (CONKLIN; BEGEMAN, 1988). Engloba questões levantadas, respostas alternativas a estas questões e argumentos contra e a favor a cada alternativa (BURGE; BROWN, 1998).

Design Rationale baseado em argumentação pode auxiliar na resolução de problemas como, por exemplo:

- Esclarecer requisitos mal definidos e rastrear sua evolução (POTTS; TAKAHASHI; ANTON, 1994);
- Representar o ponto de vista de vários *stakeholders*, incluindo usuários finais (SJÖBERG; TIMPKA, 1995);
- Manter consistência nas tomadas de decisão, através da propagação das alterações pela rede (LEE, 1990b);
- Comunicar informações sobre as decisões do projeto às demais pessoas envolvidas (MCKERLIE; MACLEAN, 1994);
- Criar uma base de conhecimento cumulativa, através do reuso sistemático do *rationale* (CARROLL; ROSSON, 1991).

Design rationale baseado em argumentação é uma abordagem essencialmente representacional e que usa um formato gráfico semiformal para organizar a estrutura de argumentos. É formado por nós e relacionamentos. Oferece uma maneira de manter consistência nas tomadas de decisão, acompanhar decisões e disseminar o conhecimento do raciocínio do projeto com outras pessoas (HU et al., 2000).

Resumidamente, o *Design Rationale* baseado em argumentação busca realizar duas tarefas ao mesmo tempo: oferecer **suporte imediato** ao processo de desenvolvimento, através da análise reflexiva de problemas complexos, o que envolve a obtenção e o armazenamento de *rationale* por um **longo tempo** a fim de ser usado como recurso reutilizável (SHUM;

MACLEAN; BELLOTTI, 1997). Este é o tipo de abordagem utilizado pelos principais modelos de representação do *Design Rationale*.

3.2.2 Baseada em Histórico

Trata-se do histórico do projeto, ou seja, da seqüência de eventos que ocorreram durante o projeto. Esta informação pode ser guardada como entradas do documento de projeto ou como um arquivo de mensagens de e-mails, entre outros (BURGE; BROWN, 1998).

As informações armazenadas são suficientemente ricas para reconstruir o projeto e conseqüentemente fornecer explicações sobre ele. Neste modelo, a documentação deve conter todas as informações e ações que aconteceram durante o projeto. De acordo com Lakin (1989), *rationale* explícito e modelos de produto sobrecarregam as atividades do projeto. O projetista acaba se distraindo com grande quantidade de detalhes, que poderiam ser melhor tratados em um estágio mais avançado.

O *Electronic Design Notebook* (EDN) é uma versão eletrônica do tradicional caderno de papel (HIROSE; CANNON; LEIFER, 1994; LAKIN, 1989; LEIFER, 1991). Ele possibilita a construção de esboços de modelos conceituais de uma maneira rápida e fácil e com uma vantagem: os esboços, cálculos, modelos e comentários podem ser processados. Eles podem ser pesquisados e recuperados, sistematicamente, e compartilhados com outros projetistas. Entretanto, o EDN não determina nenhuma estrutura particular ou prática de trabalho. O próprio usuário deve ordenar os documentos gerados e conectar partes destes documentos através de conexões de hipertexto.

Uma das suposições desta abordagem é que a obtenção e o armazenamento das ações explicam o projeto. Entretanto, muitas decisões e critérios de projeto não são registrados. As ações do projetista podem refletir parte do seu processo de desenvolvimento, mas não há como garantir que as ações possam reproduzir o processo de raciocínio do projetista. O histórico auxilia no entendimento do projeto, mas não é suficiente para explicar por que uma alternativa específica foi selecionada. A gravação da documentação é relativamente simples, em termos operacionais, mas não há como assegurar que o que foi gravado será útil ou mesmo se terá significado. Isto reduz a motivação para se usar este tipo de abordagem. Além disso, este tipo de rationale não é prático para dar suporte às primeiras etapas do projeto, pois, nesta fase, os integrantes da equipe necessitam de acesso rápido a informações que revelam e explicam as hipóteses consideradas no projeto (GARCIA; HOWARD; STEFIK, 1994).

3.2.3 Baseada em Modelo de Dispositivo

Contempla a obtenção e o armazenamento do modelo de comportamento do dispositivo; um modelo do próprio dispositivo é utilizado para gerar explicações textuais e gráficas sobre como e porque um dispositivo funciona de determinada maneira (GRUBER, 1990). Utiliza um modelo para explicar o projeto, e assume que estas explicações sobre os dispositivos são suficientes para todos os esclarecimentos necessários.

É mais poderoso e específico ao domínio que o baseado em argumentação e histórico. É baseado em muitas das técnicas e hipóteses dos sistemas especialistas baseados em modelo de dispositivo, principalmente os de diagnóstico (GARCIA; HOWARD; STEFIK, 1993).

Esta abordagem foi mais bem sucedida para diagnóstico do que em projetos propriamente ditos. Na atividade de diagnóstico, o modelo para o sistema composto de dispositivos é considerado como completo. Por outro lado, na maioria dos projetos, algumas partes do sistema não são especificadas (GARCIA; HOWARD; STEFIK, 1993).

3.2.4 Baseada em Documentos Ativos

O *Design Rationale* é gerado, previamente, com base no conhecimento guardado não apenas sobre o projeto em questão, mas sobre outros projetos. Para cada decisão que é tomada, o sistema compara a decisão do usuário com a decisão que ele teria tomado baseado no conhecimento existente. Se as ações do usuário forem conflitantes com as recomendações do sistema, ele é avisado e poderá alterar sua decisão ao modificar algum critério da base de conhecimento (BURGE; BROWN, 1998).

A maior parte das pesquisas em *Design Rationale* tem como foco a sua representação, captura e uso. Este é o ponto chave para determinar a sua utilização nas empresas, visto que estas características estão diretamente relacionadas à determinação do custo-benefício em se utilizar esta tecnologia. Estes assuntos são discutidos nos itens 3.3 a 3.5.

3.3 Representação das Informações

A representação do *Design Rationale* varia de formal a informal, conforme ilustrado na Figura 5.

Uma abordagem formal possibilita a utilização dos dados, automaticamente, pelo computador, mas nem sempre apresenta as informações em uma forma que as pessoas possam entender. Além disso, requer que os dados inseridos no sistema tenham um formato mais rigoroso. Uma abordagem informal utiliza dados em formatos que são facilmente gerados e entendidos pelas pessoas, mas não podem ser facilmente utilizados pelo computador (BURGE, 2005).

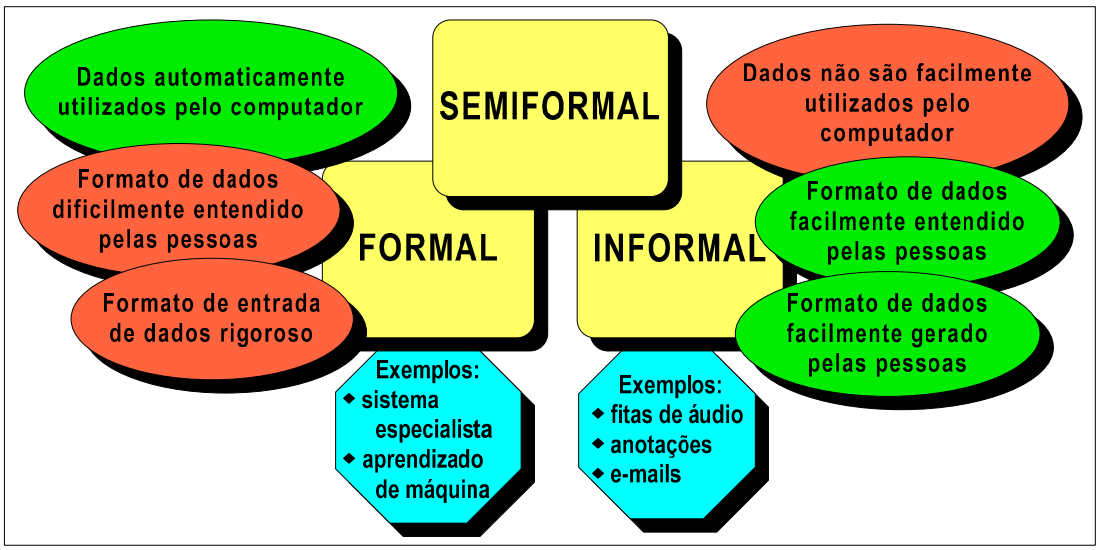

Figura 5 - Representação do *Design Rationale* – de formal a informal

Nos itens a seguir, estas formas de representação do *Design Rationale* são descritas de maneira mais detalhada.

3.3.1 Representação Informal

Representações são classificadas como informais quando elas capturam informações na forma gerada pelo projetista durante o projeto em vez de exigir que uma nova estrutura seja utilizada (BURGE, 2005). As justificativas de projeto (*rationales)* são capturadas em uma forma não estruturada.

Existem vários exemplos de representações informais, entre eles, fitas de áudio ou de vídeo, anotações, relatórios, e-mails, descrição de experimentos, etc.

3.3.2 Representação Formal

Representações formais contêm informação em formato entendido pela máquina, tornando mais fácil o processamento e a interpretação da informação, através da utilização de um computador. O tipo de representação formal dependerá das operações a serem efetuadas.

Entre os exemplos de representações formais estão as regras embutidas em um sistema especialista (CONKLIN; BURGESS-YAKEMOVIC, 1995) e um sistema de aprendizado de máquina que utiliza rastreamentos de solução de problemas passados para solucionar problemas futuros.

3.3.3 Representação Semiformal

Representações semiformais são desenvolvidas, tipicamente, na forma de argumentação.

Abordagens semiformais tentam se beneficiar das vantagens das outras duas abordagens: oferecer certo poder computacional, mas em um formato que é entendido pelas pessoas.

Quanto mais formal for a representação das justificativas de projeto, mais serviços o sistema poderá oferecer. Entretanto, formalizar o conhecimento não é uma tarefa fácil. Lee (1987) sugere a formalização incremental: a idéia principal é transformar uma representação semiformal em uma

representação formal. Conseqüentemente, as justificativas de projeto podem ser capturadas sem muita sobrecarga ao sistema usual (pois elas são capturadas de maneira semiformal), mas assim que são transformados em formais, podem ser utilizados para dar suporte a mais serviços computacionais.

3.4 Captura das Informações

Uma das dificuldades em utilizar a tecnologia de *Design Rationale* é a captura das informações. O armazenamento de todas as decisões tomadas, bem como as rejeitadas, pode consumir muito tempo e ter um alto custo. No geral, quanto mais o processo de captura for intrusivo, maior a resistência encontrada. Além do tempo extra que será necessário a esta atividade, ela pode desviar o foco do projetista de sua tarefa usual. Por outro lado, não se pode esperar que o projetista saiba expressar estas informações sem o auxílio de uma ferramenta que o guie, pois é uma tarefa cognitiva de difícil execução (GRUBER; RUSSEL, 1991).

A captura refere-se à obtenção e armazenamento do *Design Rationale*, tanto durante como após a fase de análise de projeto. A quantidade de dados que é capturada varia muito. Alguns sistemas optam por gravar o máximo possível de informações, enquanto outros utilizam uma abordagem mais focada. O desafio é proporcionar um modelo que realmente capture informações que auxiliem o usuário, sem *overhead* ou restrições excessivas.

Há vários métodos propostos, incluindo a gravação de reuniões e discussões do projeto em fitas de vídeo, indo até métodos em que os projetistas devem documentar manualmente cada decisão tomada. Lee (1997) define as seguintes formas de captura: reconstrução, subproduto metodológico, aprendiz e geração automática. Chen (1990) inclui também o historiador. Os métodos de captura não são mutuamente exclusivos e uma

ferramenta ou abordagem pode se encaixar em várias categorias. É possível também que uma ferramenta seja classificada em categorias diferentes dependendo de como ela é utilizada. A seguir, são apresentados estes métodos de captura do *Design Rationale*.

3.4.1 Reconstrução

As informações são capturadas fora da fase de análise de projeto do software (geralmente depois). A captura pode ser feita através da análise de seu próprio conhecimento, através de entrevistas com as pessoas envolvidas no projeto ou capturando o raciocínio em formato não-processado, como vídeo e então transformando estes dados em um formato mais estruturado. Outra maneira é realizar a engenharia reversa com o intuito de inferir um plano sobre o projeto do artefato a um conhecimento geral de funções e comportamentos do dispositivo (LEE, 1996). A vantagem é ser uma abordagem não-intrusiva e a desvantagem é que a captura das justificativas de projeto pode não ser precisa ou completa.

3.4.2 Subproduto Metodológico (*methodological byproduct*)

Segundo Lee (1997), no método de captura chamado subproduto metodológico, a informação aparece naturalmente durante a fase de análise do projeto do software, através da utilização de uma metodologia adequada. A metodologia utilizada, além de auxiliar no desempenho das tarefas do projeto, captura também as justificativas do projeto. Esta abordagem é atrativa porque o próprio usuário que realiza a captura das informações se beneficia do método e as justificativas do projeto são obtidas a um custo relativamente baixo. O desafio desta abordagem é fornecer um método que

realmente auxilie o usuário sem a imposição de exigências e restrições excessivas (LEE, 1996).

Para Burge (2005), a definição de Lee está sujeita a mais de uma interpretação. Certamente as atividades de projeto de software baseadas em ferramentas que realizam, automaticamente, a captura das justificativas de projeto podem ser considerados como produtores da informação, constituindo-se em um subproduto metodológico. Porém, em outros casos, não é claro se o modo de captura das justificativas do projeto pode ser considerado como subproduto metodológico. Por exemplo, no método descrito por Ganheshan et al. (1994), a descrição do projeto é modificada somente se forem feitas modificações ou refinamentos dos objetivos do projeto, ou seja, as atividades do projeto forçam a captura das suas justificativas. Uma segunda interpretação poderia considerar que esta captura é realizada como parte da fase de análise de projeto e, nesta visão, seria classificada como subproduto metodológico.

Este método de captura das justificativas do projeto é caracterizado por dois objetivos e sete princípios, listados a seguir (SCHNEIDER, 2006).

Objetivos:

1. Capturar as justificativas das decisões durante tarefas específicas contidas nos projetos de software.
2. Ser o menos intrusivo possível ao processo já existente.

Princípios:

1. Ter foco na tarefa do projeto que realiza a captura das justificativas das decisões.
2. Capturar as justificativas durante esta tarefa (e não como uma atividade separada).

3. Deixar o mínimo possível de obrigações extras à pessoa responsável pela captura das informações.
4. Ter foco na gravação durante a atividade original.
5. Utilizar um computador para gravar e capturar informações específicas extras.
6. Analisar gravações, buscando identificar padrões.
7. Encorajar, mas não insistir na inclusão de novas atividades de gerenciamento do *Design Rationale*.

3.4.3 Aprendiz

Neste tipo de abordagem, o sistema acompanha as ações tomadas pelo projetista e faz perguntas quando não entende ou não concorda com uma ação. As justificativas de projeto são, de certa maneira, pré-geradas. Se as ações do projetista estiverem de acordo com o que o sistema esperava, o raciocínio gerado pelo sistema é salvo. Segundo Lee (1997), a vantagem desta abordagem é que esta interação traz benefícios tanto para o usuário que está inserindo a informação, como para o sistema. Além disso, as informações são capturadas no momento que estão sendo utilizadas, diminuindo a probabilidade de obter informações imprecisas ou incompletas.

3.4.4 Geração Automática

Em sua forma mais simples, as informações são geradas a partir de um histórico de execuções (LEE, 1996). Esta abordagem tem a vantagem de obter as informações a um baixo custo e de ter a possibilidade de manter uma base de conhecimento consistente e atualizada. Entretanto, há um custo inicial alto para analisar as informações relevantes e construir as justificativas de projeto. Além disso, analisando o método mais

profundamente, percebe-se que muitas questões ligadas à pesquisa sobre aprendizado de máquina devem ser resolvidas: que partes do rastreamento da solução do problema devem ser guardadas, como salvar, como inferir justificativas de projeto a partir destas informações, como determinar sua importância e como adaptar estas informações à situação atual.

3.4.5 Historiador

Uma pessoa ou um programa realiza um rastreamento de todas as ações feitas durante a fase de elaboração do projeto do software. Este método é semelhante ao aprendiz, com exceção de que o sistema não faz sugestões. É semelhante à geração automática, exceto que o raciocínio é especificamente gravado durante a fase de análise do projeto, e não gerado posteriormente (CHEN, 1990).

O Quadro 2 mostra um resumo destas técnicas de captura levando-se em conta o momento em que a informação é capturada e o nível de interação com o usuário. Métodos que interagem muito com o usuário tendem a ser mais intrusivos do que os que não interagem. É claro que isso não necessariamente significa uma desvantagem da técnica caso esta interação seja vista como útil ou se é consistente com o processo normal da fase de análise de projeto do software.

A captura do *Design Rationale* não tem utilidade se não houver um modo eficiente de acessá-lo. Este é o assunto do próximo item.

	Momento da Captura		Interação com usuário	
	Durante o projeto	Depois do projeto	Baixa	Alta
Reconstrução		✓	✓	
Subproduto Metodológico	✓		✓	
Aprendiz	✓			✓
Geração Automática		✓	✓	
Historiador	✓		✓	

Quadro 2 - Quadro comparativo dos métodos de captura do *Design Rationale*

3.5 Formas de Utilização

O Design Rationale consiste em informações variadas sobre o desenvolvimento de um determinado sistema. Estas informações podem auxiliar diversas atividades relacionadas direta ou indiretamente ao projeto:

- **Verificação**: o *Design Rationale* pode ser utilizado para verificar se o projeto satisfaz os requisitos do sistema e as intenções do projetista.
- **Avaliação**: o *Design Rationale* pode ser útil para avaliar projetos (e partes de projetos) e escolhas realizadas, a fim de detectar inconsistências que podem afetar a qualidade do projeto. Pode fornecer auxílio durante toda a fase de análise de projeto do software. É possível verificar e avaliar decisões de projeto no momento em que são tomadas, evitando-se a violação de uma restrição ou efeitos colaterais não permitidos.
- **Manutenção**: o *Design Rationale* pode ser utilizado para determinar quais escolhas foram feitas durante as atividades do projeto com a finalidade de localizar fontes de problemas ou indicar que mudanças

devem ser feitas para modificar o projeto; além disso, rastreando-se as alternativas que já foram rejeitadas, o projetista pode evitar que se adote uma solução já rejeitada anteriormente.

- **Reuso**: o *Design Rationale* pode ser útil na definição das partes do projeto que podem ser reutilizadas e, em alguns casos, sugerir onde e como deve ser modificado para se adequar ao novo conjunto de requisitos. É especialmente importante que o projetista saiba por que as decisões foram tomadas.
- **Ensino**: o *Design Rationale* pode auxiliar no ensino de um projeto a novos integrantes, mostrando a razão de cada decisão tomada. Alguns sistemas de *Design Rationale* possibilitam ao usuário fazer perguntas sobre o projeto, o que é geralmente mais rápido e mais fácil do que analisar grande quantidade de documentação de um projeto.
- **Comunicação**: o *Design Rationale* pode melhorar a disseminação de conhecimento sobre um projeto, tanto durante, como depois do projeto inicial. Pode também detectar conflitos em ambientes com vários projetistas e diminuir o ciclo de revisão.
- **Documentação**: o *Design Rationale* pode auxiliar na documentação do projeto, complementando com informações importantes não captadas na documentação "convencional". Se estas informações estiverem guardadas em formato que o computador entende, podem ser utilizadas como parte de um sistema de geração de documentos personalizados, através da escolha de diferentes perspectivas. Por exemplo, o cliente deve precisar de um nível de detalhamento diferente de um projetista. Alguns sistemas vão além, permitindo que o usuário faça perguntas sobre o projeto.

Observa-se que o *Design Rationale* pode ser utilizado com inúmeros propósitos. A seguir, são descritos alguns sistemas que utilizam o *Design Rationale*, ilustrando esta variedade de opções.

Alguns sistemas somente suportam a recuperação das informações. O modo como é utilizado após esta recuperação é de responsabilidade do projetista. Um exemplo deste sistema é o JANUS, um ambiente para projeto de cozinhas (FISHER et al., 1995). Outros sistemas suportam a recuperação da informação e também oferecem a possibilidade de verificar a consistência e a completude no *Design Rationale* ou no projeto. Um exemplo é o SIBYL (LEE, 1990b). Ele oferece serviços que realizam a verificação de consistência da base de conhecimento e também serviços que avaliam as alternativas baseados em alegações a favor e contra as decisões tomadas.

No *Knowledge-Based Design System* (KBDS), palavras-chave são utilizadas para verificar a consistência de redes IBIS (*Issue Based Information System*) que contêm as justificativas de projeto (KING; BAÑARES, 1997). No C-Re-CS (KLEIN, 1997), é realizada uma verificação de consistência dos requisitos e uma estratégia de resolução de exceções detectadas é sugerida pelo sistema. O sistema *Accord* (ULLMAN, 2004) utiliza uma tabela verdade para fazer uma associação com as justificativas de projeto, tendo como objetivo realizar uma análise de riscos de conceitos ou do sistema que está sendo desenvolvido. Esta avaliação mostra se as melhores decisões possíveis foram tomadas.

Outra característica útil oferecida por alguns sistemas é a habilidade de fazer perguntas sobre o projeto ou sobre suas justificativas. O *Device Modeling Environment* (DME) suporta um conjunto de perguntas (pré-enumeradas) que podem ser utilizadas para obter explicações sobre as decisões do projeto (GARCIA; HOWARD; STEFIK, 1993). É possível, por exemplo, pedir explicações sobre a razão do valor de um parâmetro ter sido alterado.

Outro uso do *Design Rationale* é o suporte à colaboração, usando o *rationale* como um meio de comunicação entre diferentes pessoas ou equipes de um projeto. SHARED-DRIM é um sistema baseado no modelo *Design Recommendation and Intent Model* (DRIM), cujo principal objetivo é

capturar o *Design Rationale* para ser utilizado na mitigação de conflitos (PEÑAMORA et al., 1995). As informações sobre decisões aceitas ou rejeitadas são compartilhadas entre os projetistas participantes, diminuindo o ciclo de aprovação e modificação do projeto.

Os sistemas podem ser classificados como “iniciativa do usuário” ou “iniciativa do sistema”, dependendo de quem inicia o processo. Em um sistema de “iniciativa do usuário”, este decide quais as partes do *Design Rationale* deseja examinar e quando e quem irá acessar. Em um sistema de “iniciativa do sistema”, por outro lado, é o sistema que decide quando e como apresentar determinadas partes do *Design Rationale*. Estes sistemas precisam ter conhecimento suficiente para realizar decisões inteligentes, apresentando as informações de maneira não intrusiva ao usuário (LEE, 1996).

Uma questão crítica associada ao acesso das informações é o tempo para recuperação dos dados. Os projetistas se sentirão interrompidos se a resposta do sistema for lenta conforme suas expectativas, não importando o quanto este tempo é apropriado ao tamanho e complexidade da base de conhecimento (LEE, 1997). Por isso, há a necessidade de planejar e definir quais informações serão armazenadas, de que maneira e através de que ferramentas.

Apesar do grande número de vantagens que a utilização do *Design Rationale* pode oferecer, existem poucos exemplos de seu uso nas empresas.

No próximo item são apresentadas características importantes de ferramentas utilizadas para captura e recuperação de *Design Rationale*.

3.6 Ferramentas de Design Rationale

Há uma tendência à integração das ferramentas de *Design Rationale* a outras ferramentas de projeto, considerando o sistema de *Design Rationale* como sendo uma extensão do sistema de projeto de software (HU et al., 2000).

Segundo Burge (2005), uma ferramenta desenvolvida para dar suporte ao uso do *Design Rationale* deve ter as seguintes características:

1. Integração com o ambiente de desenvolvimento. Os projetistas não devem ser obrigados a abrir uma ferramenta computacional adicional somente para criar e acessar as justificativas de projeto. As informações precisam estar disponíveis nas ferramentas já em uso.
2. Associação do *Design Rationale* a artefatos de desenvolvimento. Por exemplo, deve haver uma maneira de associar, explicitamente, as justificativas de projeto, particularmente as alternativas, com o código que os implementa.
3. Apresentação automática do *rationale*. O usuário deve receber uma mensagem avisando que existem justificativas de projeto disponíveis quando estiver trabalhando com um código que possui esta informação. Não se deve haver a necessidade de, deliberadamente, verificar a existência ou não de justificativas de projeto associadas com o código que eles estão trabalhando.
4. Visualização dos argumentos do *Design Rationale*. As justificativas de projeto devem ser exibidas em um formato hierárquico, tirando proveito da estrutura de argumentação.
5. Verificação automática do *Design Rationale*. O status das justificativas de projeto (erros e avisos gerados durante inferências) deve ser atualizado, automaticamente, quando o *rationale* é adicionado, removido ou modificado.

6. Interface de consulta do *Design Rationale*. Deve haver uma maneira de procurar facilmente por uma informação específica.

A escolha de uma ferramenta adequada ao projeto é um dos pontos importantes para o sucesso de sua utilização.

A seguir, são apresentados os principais modelos de *Design Rationale.*

3.7 Modelos de Representação do Design Rationale

Uma grande variedade de abordagens a *Design Rationale* tem sido proposta. A maioria difere apenas na nomenclatura dos nós e de seus relacionamentos. Segundo grande parte dos pesquisadores, incluindo Stumpf (1997) e Shum (1991a), o foco está em três modelos: *Issue Based Information System* (IBIS), *Questions, Options and Criteria* (QOC) e *Decision Representation Language* (DRL). Estes modelos são descritos a seguir.

3.7.1 *Issue Based Information System* (IBIS)

Historicamente, o movimento de *Design Rationale* iniciou-se com o *Issue Based Information System* (IBIS), descrito por Kunz e Rittel. Tinha o intuito de fornecer suporte a grupos que eram confrontados com problemas complexos (KUNZ; RITTEL, 1970). O sistema guiava a identificação, estruturação e decisão de questões levantadas em grupos de resolução de problemas e fornecia informação apropriada à discussão, criando um plano de decisão. Não estava relacionado a software, mas sim a um modo de modelar a argumentação em geral.

Poucos anos depois, Rittel já estava convencido de que problemas de projeto eram fundamentalmente diferentes dos problemas bem-definidos da ciência e os chamou de “problemas difíceis” (*wicked problems*). Estes problemas não poderiam ser resolvidos através das abordagens analíticas tradicionais (RITTEL; WEBBER, 1973). Rittel (1972) propôs então uma abordagem argumentativa a estes tipos de problemas e utilizou o modelo IBIS para implementar esta abordagem.

O IBIS é utilizado para registrar idéias e relacionamentos durante discussões de projeto enquanto elas estão ocorrendo. Apresenta uma estrutura sobre como as questões são discutidas. O foco não está em como o problema é resolvido, como as alternativas são extraídas e avaliadas ou como se chega a um determinado consenso. O IBIS coloca mais ênfase na representação do processo pelo qual as decisões são tomadas. Portanto, o modelo IBIS tem o intuito de fornecer suporte à estruturação da discussão de forma que as informações capturadas e estruturadas possam auxiliar os desenvolvedores a resolver suas questões.

A estrutura do método IBIS consiste de três tipos de nós:

1. QUESTÕES: problemas em discussão;
2. POSIÇÕES: possíveis soluções para os problemas;
3. ARGUMENTOS: opiniões favoráveis ou não às soluções levantadas.

Uma QUESTÃO pode questionar, generalizar ou especializar outra QUESTÃO, resultando em SUBQUESTÕES, cada uma com POSIÇÕES e ARGUMENTOS (NGUYEN; SWATMAN; SHANKS, 1998). QUESTÕES são criadas e discutidas pelo fato de que diferentes opiniões são assumidas. Os ARGUMENTOS são construídos em defesa ou contra as diferentes POSIÇÕES até que a QUESTÃO seja resolvida. Isto ocorre quando os oponentes são convencidos ou quando é possível aplicar um procedimento

de decisão formal (KUNZ; RITTEL, 1970). Estes relacionamentos são ilustrados na Figura 6.

As QUESTÕES são os elementos organizacionais dos sistemas. Entre suas propriedades, pode-se citar (KUNZ; RITTEL, 1970):

- QUESTÕES têm a forma de perguntas;
- A origem de QUESTÕES são afirmações contraditórias;
- QUESTÕES são específicas a situações particulares. POSIÇÕES são desenvolvidas utilizando informações particulares do ambiente do problema;
- QUESTÕES são sugeridas, questionadas, especializadas, generalizadas ou substituídas.

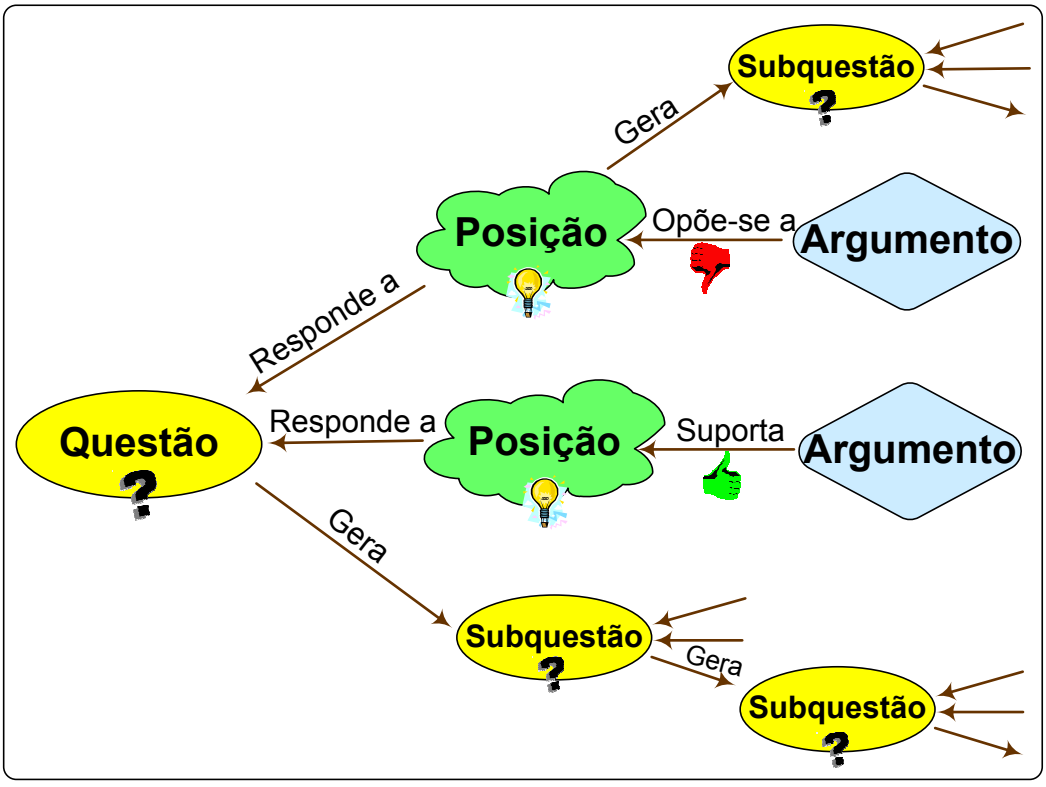

Figura 6 - IBIS: Representação dos nós, baseada em Kunz e Rittel (1970)

Conklin e Begeman (1988) adaptaram o IBIS para uso em engenharia de software, criando um sistema hipertexto chamado *graphical IBIS* (gIBIS). As principais alterações e extensões realizadas foram:

- Criação de um tipo adicional de nó e relacionamento chamado OUTRO, utilizado como um mecanismo de escape para casos em que não tenha sido possível encontrar uma maneira de expressar um pensamento dentro do modelo IBIS;
- Criação de um tipo adicional de nó chamado EXTERNO, que contém material não-IBIS, como documentos de requisitos, esboços de projetos ou códigos;
- Inclusão dos relacionamentos ESPECIALIZA e GENERALIZA entre dois nós POSIÇÃO e entre dois nós do tipo ARGUMENTO.

A Figura 7 mostra os tipos de nós e relacionamentos do modelo IBIS estendido utilizado na construção do gIBIS. As conexões possíveis são:

- RESPONDE A: uma POSIÇÃO pode responder a uma QUESTÃO; esta é a única possibilidade de ligação desta conexão;
- SUPORTA, OPÕE-SE A: os ARGUMENTOS podem justificar ou se oporem a uma POSIÇÃO;
- GENERALIZA, ESPECIALIZA: QUESTÕES podem generalizar ou especializar outras QUESTÕES;
- QUESTIONA, É SUGERIDA POR: QUESTÕES podem questionar ou serem sugeridas por outras QUESTÕES, POSIÇÕES e ARGUMENTOS;
- SUBSTITUI: uma QUESTÃO pode ser substituída por outra QUESTÃO;
- OUTROS: como um mecanismo de escape, o nó OUTROS pode se conectar com qualquer outro tipo de nó através do conector OUTROS.

Conklin e Begeman (1988) definem gIBIS como um sistema hipertexto que utiliza cores e um servidor de banco de dados relacional de alto desempenho para facilitar a construção e a pesquisa de redes do tipo IBIS.

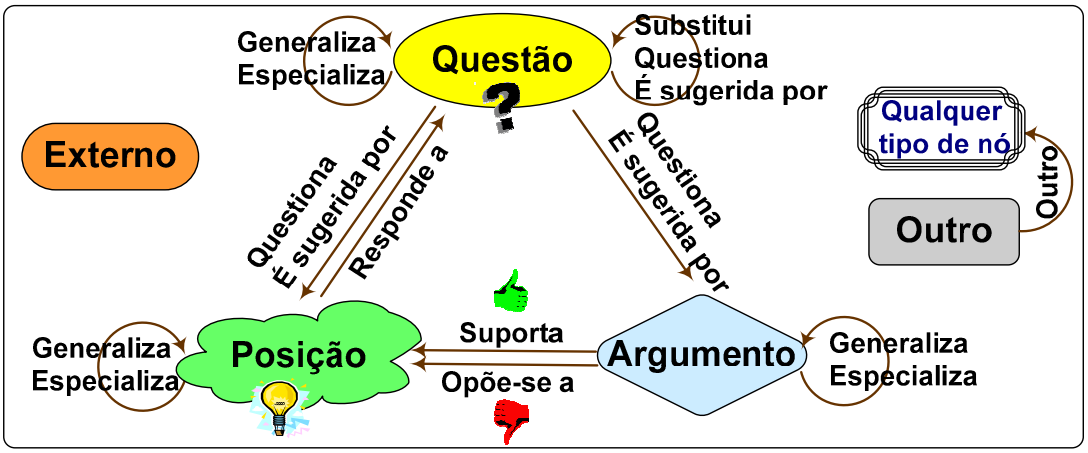

Figura 7 - gIBIS: Extensão do IBIS

Além do gIBIS, outros modelos foram criados como uma extensão ao IBIS (LOURIDAS; LOUCOPOULOS, 2000): *Representation and Maintenance of Process Knowledge* (REMAP), *Potts & Bruns*, *Potts*, *Procedural Hierarchy of Issues* (PHI), entre outros. Basicamente, são similares ao IBIS, mas possuem outras entidades no modelo.

A grande vantagem do modelo IBIS é que o *Design Rationale* pode ser capturado de maneira informal e em um estágio inicial do processo, assegurando que as questões do projeto estão entendidas. É considerado um modelo de discussão muito poderoso, podendo também ser utilizado por usuários que trabalham sozinhos. Eles afirmam que a organização de idéias os ajuda a prestar mais atenção às partes mais complexas e críticas do problema, além de ajudar a identificar falta de consistência dos pensamentos mais rapidamente.

Para os usuários que trabalham em equipe, a estrutura utilizada na discussão é muito útil e serve para mostrar opiniões pessoais, sinalizações de concordância e tornam as suposições e definições mais claras (CONKLIN; BEGEMAN, 1988).

Entretanto, por ser um modelo simples, com apenas três tipos de nós, muitas informações acabam sendo representadas em um único nó (MONK et al., 1995). O nó ARGUMENTO, por exemplo, armazena também requisitos, restrições e objetivos.

Além disso, por não ter uma representação hierárquica, gera uma rede complexa quando armazena muitos nós (FRANCISCO, 2004). Conseqüentemente, o desempenho da busca de informações fica comprometido. O modelo IBIS não pode fazer uso de uma exploração estruturada dos assuntos. O controle fica muito mais nas mãos do projetista e demais pessoas envolvidas que devem investigar maneiras factíveis de resolver cada problema.

Por um lado, IBIS é considerado intuitivo. Por outro lado, é restrito pelo contexto local e pela falta de objetivos explícitos e de resultados da argumentação (NGUYEN; SWATMAN; SHANKS, 1998).

3.7.2 *Questions, Options and Criteria* (QOC)

Desenvolvido por MacLean et al. (1991), este modelo destaca a exploração de um espaço de alternativas do projeto, o espaço do projeto e as escolhas realizadas. É uma abordagem que utiliza notação semiformal baseada em argumentação para representar, sistematicamente, visões do espaço de projeto. O espaço de projeto é um conjunto de relacionamentos ou dimensões conceituais usado para comparar projetos e soluções alternativas de instanciar estes conceitos (SHUM, 1991a). Segundo Shum et al. (1996), o espaço de projeto nunca poderá ser representado em sua totalidade, pois podem surgir questões até um nível infinito de detalhamento e de numerosas perspectivas.

O modelo apresenta uma estrutura baseada no processo como as alternativas são geradas e avaliadas. Esta organização possibilita a representação explícita de um espaço estruturado de alternativas de projeto e as considerações que levaram à sua escolha. Dentro das opções de espaço de projeto há possíveis respostas às perguntas (JARCZYK; LOFFLER; SHIPMAN, 1992).

O modelo desenvolvido inclui três tipos de nós: QUESTÕES, OPÇÕES e CRITÉRIOS. As QUESTÕES são os problemas chave a resolver, as OPÇÕES são as alternativas levantadas para resolver os problemas identificados e os CRITÉRIOS justificam as opções existentes. Além destes elementos existem também os julgamentos (*assessments*), que são os relacionamentos entre OPÇÕES e CRITÉRIOS. Em sua forma mais simples, os julgamentos podem possuir dois tipos de relacionamentos: SUPORTA ou OPÕE-SE A. Os ARGUMENTOS são usados para conduzir a discussão sobre a situação destas entidades e seus relacionamentos. A Figura 8 mostra a estrutura geral do modelo QOC.

Em resumo, o modelo visa identificar problemas chave (QUESTÕES) e levantar e justificar (via CRITÉRIOS) alternativas de projeto (OPÇÕES).

Uma diferença do QOC em relação ao IBIS é o armazenamento dos critérios (LU et al, 1999), possibilitando a captura explícita das restrições do projeto e facilitando a resolução do problema.

Segundo Hu et al. (2000), QOC é simples de aprender e utilizar e há um número crescente de projetos de pesquisa no assunto. Outra vantagem destacada pelos autores é que é relativamente fácil criar um QOC para a engenharia reversa de parte do sistema e preservá-la para uso futuro.

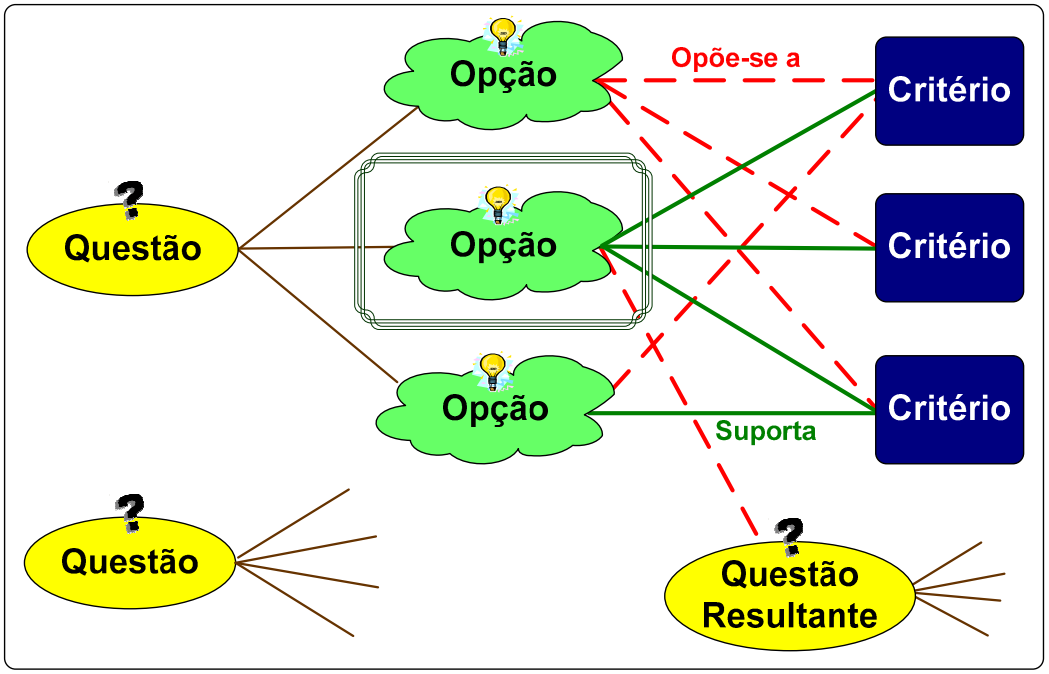

Figura 8 - Modelo QOC, baseada na descrição de MacLean et al. (1991)

3.7.3 *Decision Representation Language* (DRL)

Lee e Lai (1991) propuseram uma notação que era uma extensão dos modelos QOC e gIBIS. Como o próprio nome diz, *Decision Representation Language* (DRL) foi desenvolvido como uma linguagem para representar tomadas de decisão. Uma das principais preocupações foi o aumento da expressividade e funcionalidade. Para contemplar estas características, houve um aumento da complexidade (LOURIDAS; LOUCOPOULOS, 2000).

É um modelo de argumentação bem mais completo, envolvendo um número de nós e relacionamentos bem maior que os outros dois modelos. Jarczyk, Loffler e Shipman (1992) relacionam sete tipos de nós principais (ALTERNATIVA, OBJETIVO, ALEGAÇÃO, QUESTÃO, PROCEDIMENTO, GRUPO E PONTO DE VISTA) de uma hierarquia com 26 tipos de nós e mais de 20 relacionamentos.

Para Louridas e Loucopoulos (2000), existem 5 entidades principais: PROBLEMA DE DECISÃO, OBJETIVO, ALTERNATIVA, ALEGAÇÃO e QUESTÃO. Lee (1990a) inclui os nós PROCEDIMENTO e GRUPO. A seguir descrevem-se essas sete entidades:

- PROBLEMAS DE DECISÃO e OBJETIVOS; um PROBLEMA DE DECISÃO é um assunto de controvérsia em projeto e um OBJETIVO é um conjunto de requisitos (critérios) para solucioná-lo.
- ALTERNATIVAS representam possíveis soluções aos PROBLEMAS DE DECISÃO.
- ALEGAÇÕES são utilizadas para argumentação. Como todos os relacionamentos são subclasses do objeto ALEGAÇÃO, uma argumentação pode ser construída em cima de outra argumentação.
- QUESTÕES são utilizadas para conduzir as discussões durante o projeto.
- PROCEDIMENTOS são passos que devem ser realizados para se obter resposta a uma questão. Representam aspectos auxiliares no processo de tomada de decisão.
- GRUPOS reúnem vários objetos; possuem atributos que indicam qual o relacionamento entre estes objetos.

Os principais relacionamentos entre os nós do modelo DRL são (LEE, 1990a):

- Um PROBLEMA DE DECISÃO “é uma subdecisão de” um outro PROBLEMA DE DECISÃO;
- Um OBJETIVO “é um subobjetivo de” um PROBLEMA DE DECISÃO;
- Uma ALTERNATIVA “é uma alternativa para” um PROBLEMA DE DECISÃO;
- Uma ALEGAÇÃO “suporta” ou “é contra” uma ALTERNATIVA ou uma outra ALEGAÇÃO;.
- Uma ALEGAÇÃO “pressupõe” outra ALEGAÇÃO;

- Uma ALEGAÇÃO “responde” uma QUESTÃO;
- Uma ALEGAÇÃO “é um resultado de” um PROCEDIMENTO;
- Uma ALEGAÇÃO “é um membro de” um GRUPO;
- Uma QUESTÃO “questiona” ou “influencia” uma ALEGAÇÃO;
- Um PROCEDIMENTO “é uma resposta para” uma QUESTÃO;
- Um PROCEDIMENTO “é um subprocedimento de” outro PROCEDIMENTO;
- Um GRUPO “é um conjunto de possíveis respostas para” uma QUESTÃO.

A Figura 9 ilustra o modelo DRL, mostrando os possíveis relacionamentos entre os nós.

Apesar de ter também uma notação semiformal, ela suporta mais informação formal na descrição da informação se comparada às outras notações (JARCZYK; LOFFLER; SHIPMAN, 1992). Esta representação explícita do conhecimento é muito apropriada para possibilitar que o sistema forneça mais serviços ao projetista.

Esta abordagem foi utilizada no desenvolvimento de uma ferramenta baseada em conhecimento chamada SIBYL para representar justificativas de decisões de projeto (LEE, 1990b). Primeiramente, criam-se instâncias de um PROBLEMA DE DECISÃO. Em seguida, adicionam-se os OBJETIVOS e as ALTERNATIVAS (MONK et al., 1995). Através de módulos apropriados, é realizado o processo de avaliação e argumentação destas informações.

A ferramenta fornece vários serviços computacionais incluindo o gerenciamento de dependência (monitoramento de decisões que dependem uma das outras), gerenciamento de precedência (outras decisões compartilham os mesmos objetivos), gerenciamento de ponto de vista (argumentos compartilhando as mesmas suposições) e gerenciamento de plausibilidade (o poder de dar suporte à argumentação de uma alternativa).

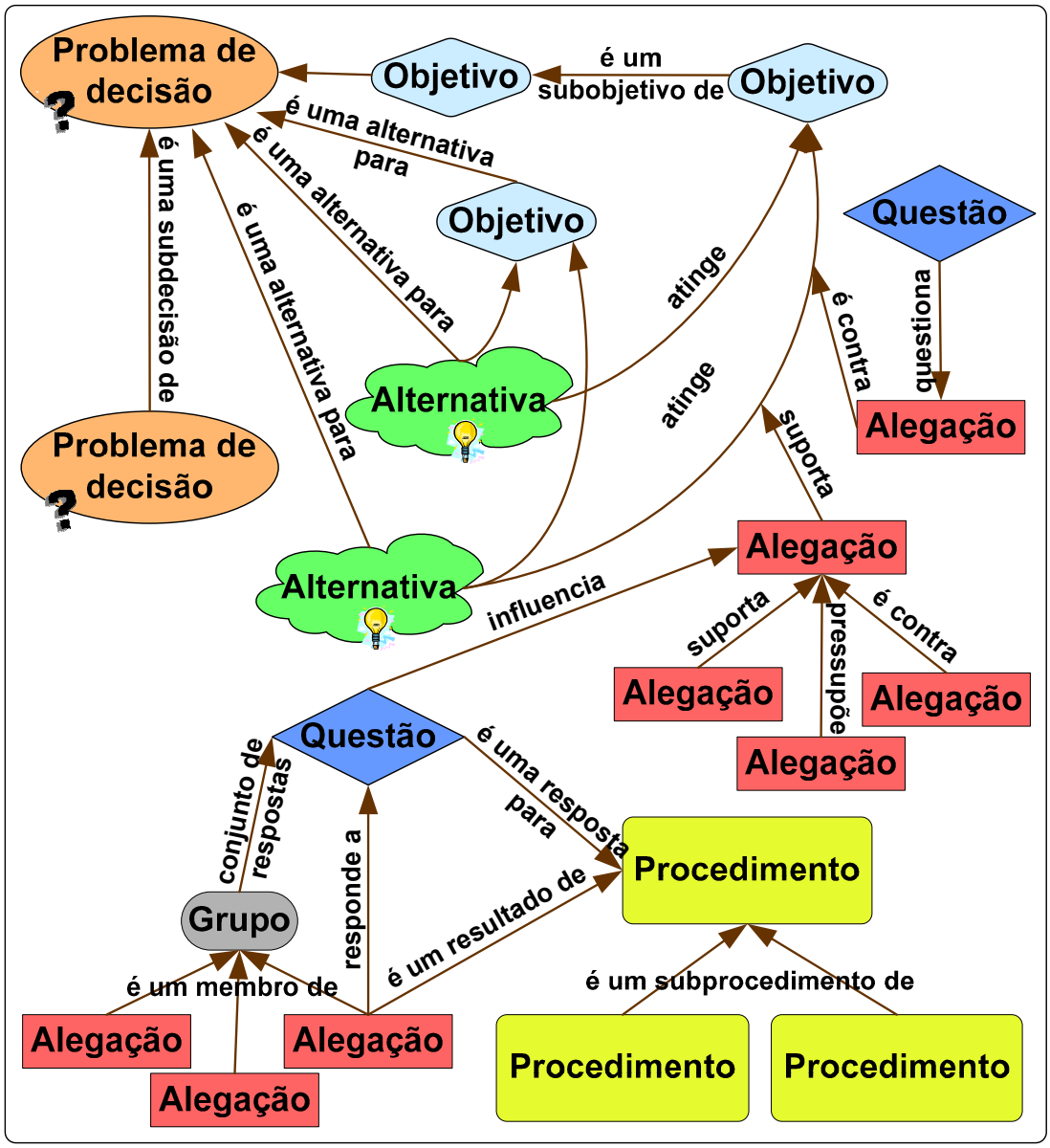

Figura 9 - Modelo DRL, baseado em Lee (1990a)

Lee e Lai (1991) defendem que uma representação deve suportar uma variedade de tarefas de projeto, tais como responder questões sobre o progresso do projeto, as alternativas geradas, as avaliações que levaram à escolha de determinadas alternativas e a possível transferência de conhecimento a projetos futuros ou outras pessoas. DRL foi desenvolvido para suportar todas estas questões. Sua ênfase é gerenciar os elementos qualitativos das tomadas de decisão e do gerenciamento de suas

dependências. Portanto, é mais um sistema de gerenciamento de *Design Rationale*.

No próximo item, é realizada uma análise comparativa entre estes três modelos.

3.7.4 Comparação entre os modelos

Os três modelos apresentados são baseados em argumentação e possuem notação semiformal. Eles diferem na extensão de características que almejam capturar e nas funções que suportam pelo modo como estes modelos são utilizados em uma representação.

Em termos gerais, os três modelos possuem os dois nós que se referem aos problemas e às alternativas de solução. Cada modelo utiliza uma nomenclatura própria para se referir a estas entidades. IBIS e DRL possuem um nó que define os argumentos para a escolha ou rejeição de uma alternativa. QOC e DRL possuem um nó que indica os critérios necessários à solução de um problema. Esta característica é importante para uma melhor e mais precisa tomada de decisão (FRANCISCO, 2004), pois estes critérios indicam propriedades desejadas ou requisitos que devem ser satisfeitos. Por outro lado, QOC não apresenta explicitamente as informações referentes aos argumentos, que acabam sendo armazenadas implicitamente nos nós OPÇÃO. DRL possui um número maior de tipos de nós e de relacionamentos, fornecendo uma maior flexibilidade, mas tornando seu entendimento mais complexo. O Quadro 3 mostra os termos utilizados por cada modelo.

	IBIS	QOC	DRL
Problema	Questão	Questão	Problema de decisão
Alternativa	Posição	Opção	Alternativa
Argumento	Argumento	---	Alegação
Critério	---	Critério	Objetivo

Quadro 3 - IBIS, QOC e DRL: nomenclatura utilizada na representação gráfica (elaborado pela autora)

O Quadro 4 apresenta um comparativo das características dos três modelos. Algumas delas já foram mencionadas no item 3.7. Outras são explicadas a seguir.

	IBIS	QOC	DRL
Ano de criação	1970	1991	1991
Criador	Kunz e Rittel	McLean	Lee e Lai
Tipo	Argumentativo	Argumentativo	Argumentativo
Representação	Semiformal	Semiformal	Semiformal
Objetivo	Prescritivo	Descritivo	Descritivo
Representação de critérios	Não	Sim	Sim
Objeto de representação	Qualquer questão de projeto	Somente propriedades do artefato	Qualquer questão de projeto
Simples ou complexo	Simples	Simples	Complexo

Quadro 4 - IBIS, QOC e DRL: características principais (elaborado pela autora)

Dependendo do objetivo do modelo, pode-se classificá-lo como descritivo ou prescritivo. No descritivo, o objetivo é somente descrever os processos de raciocínio dos projetistas. Não é feita nenhuma tentativa de alterar seu modo de pensar ou mesmo a decisão tomada. Entretanto, as informações guardadas podem melhorar processos de outras fases do ciclo de desenvolvimento de software, como, por exemplo, a implantação, manutenção ou reuso de artefatos de projeto. O *Design Rationale* pode também ser utilizado para transferência de conhecimento a novos integrantes da equipe.

A abordagem prescritiva, por outro lado, tem o objetivo de melhorar os processos durante a fase de elaboração do projeto através do aperfeiçoamento do modo de pensar dos envolvidos. Procura-se corrigir deficiências percebidas no raciocínio de questões do projeto tornando-o mais correto, mais consistente e mais completo. Nesta abordagem também se pode criar registros de *Design Rationale* para serem utilizados no auxílio a processos fora da fase de elaboração do projeto. Deve-se notar também que descritivo e prescritivo não são sempre mutuamente exclusivos. Por exemplo, algumas abordagens têm a intenção principal de serem descritivas, mas também têm alguns objetivos prescritivos (DUTOIT et al., 2006).

Considerando-se os objetivos básicos dos modelos, QOC e DRL são classificados como descritivos e IBIS é classificado como prescritivo. QOC é fundamentalmente descritivo visto que sua principal finalidade no sistema é criar uma representação das decisões dos projetistas que seja suficientemente detalhada para fornecer informações a outras fases do ciclo de desenvolvimento do artefato.

Para Dutoit et al. (2006), duas características, em resumo, distinguem QOC de IBIS. A primeira refere-se à abrangência das questões. Enquanto IBIS pode referir-se a qualquer tópico de projeto, as questões do modelo QOC tratam exclusivamente de características do artefato que está sendo projetado. Por isso, as questões do modelo QOC sempre têm possíveis respostas (OPÇÕES), que descrevem as propriedades do artefato projetado. O segundo fator que distingue os dois modelos é que o QOC utiliza julgamentos para indicar se as alternativas cumprem cada critério definido. Apesar de ser possível representar estes julgamentos no modelo IBIS, através dos argumentos, IBIS não possui uma representação explícita dos critérios como elementos do modelo. Nguyen, Swatman e Shanks (2000) complementam ressaltando que cada alternativa é avaliada isoladamente

por seus próprios argumentos. Não há um conjunto comum de argumentos a todas as alternativas.

Outra limitação do modelo IBIS é a geração de uma rede complexa quando armazena muitos nós, dificultando a busca de dados (FRANCISCO, 2004).

O modelo DRL é bem mais completo, envolvendo um número de nós e relacionamentos bem maior que os outros dois modelos. Um dos objetivos foi o aumento da expressividade e funcionalidade. Lee e Lai (1996) argumentam que o DRL é mais expressivo que os outros modelos de argumentação porque atende a uma faixa mais ampla de questões que podem surgir em variadas fases do ciclo de vida de um artefato. Por outro lado, para contemplar estas características, houve um aumento da complexidade (LOURIDAS; LOUCOPOULOS, 2000).

O Quadro 5 mostra um resumo das vantagens e limitações de cada modelo. A melhor escolha dependerá da aplicação e objetivos específicos do ambiente analisado.

	IBIS	QOC	DRL
Pontos fortes	• Modelo Simples • Auxílio às primeiras tarefas do processo • Trata questões sobre qualquer tópico do projeto	• Modelo Simples • Mostra explicitamente os critérios	• Representação expressiva; captura de relações complexas • Mais completo
Limitações	• Não possui nó representando os critérios • Rede complexa quando armazena muitos nós	• Trata apenas questões sobre o artefato • Auxílio apenas a outras fases do projeto	• Maior complexidade para entendimento

Quadro 5 - QOC, IBIS e DRL: análise (elaborado pela autora)

Esta análise comparativa entre os modelos forneceu a base para a proposta que é apresentada no próximo capítulo. Pela análise teórica, DRL parece ser muito complexo para ser utilizado em sistemas ERP. A melhor solução provavelmente será baseada no modelo QOC ou IBIS.

4 PROPOSTA DE UM MODELO DE REPRESENTAÇÃO DO *DESIGN RATIONALE* PARA PROJETOS DE SISTEMAS ERP

Neste capítulo é apresentada a proposta deste trabalho. Inicialmente, é realizada uma análise de adequação da utilização de *Design Rationale* em sistemas ERP. Em seguida, é descrita a pesquisa em campo, apresentando uma representação gráfica do *Design Rationale* para cada caso. Finalmente, é apresentada uma proposta de modelo de *Design Rationale* para projetos de sistema ERP.

4.1 Design Rationale e ERP

Neste item é realizada uma análise de aplicabilidade do *Design Rationale* a projetos de sistemas ERP. O intuito é confirmar que o *Design Rationale* é útil para estes tipos de sistemas de software.

Se as informações sobre as justificativas das decisões do projeto estiverem disponíveis, é possível utilizá-las de diversas maneiras com o objetivo de fornecer suporte às atividades de projeto. Gruber e Russel (1991) apresentam algumas destas atividades:

- Verificação do trabalho: como as informações capturadas explicam determinadas partes do projeto, é possível realizar verificações com o intuito de descobrir inconsistências;
- Esclarecimentos sobre o projeto: *Design Rationale* fornece informações importantes sobre o projeto. É muito útil principalmente em projetos complexos;

- Compartilhamento de informações: um histórico do projeto pode evitar a tentativa de utilizar alternativas já exploradas. Isso é especialmente importante em projetos em que pessoas que não participaram do projeto inicial devem definir correções ou adicionar novas funcionalidades ao sistema;
- Memória institucional: em grandes organizações, muitos projetistas trabalham, simultaneamente, em vários projetos, ao longo de grandes períodos de tempo. Eles precisam compartilhar modelos de artefatos comuns, comunicarem-se uns com os outros e também com outros departamentos. Capturar o conhecimento através *do Design Rationale*, em um formato explícito, é importante para acumular e compartilhar conhecimento dentro da organização.

Analisando-se um projeto de sistema ERP, observa-se que ele possui diversas características de sistemas com maiores chances de se beneficiar da utilização da tecnologia de *Design Rationale*.

Os sistemas ERP são desenvolvidos ao longo de um extenso período e, geralmente, por grandes grupos de pessoas. O usual é que o software sofra evoluções sistêmicas e tecnológicas sempre a partir do produto atual, ou seja, novas versões do sistema são construídas tendo como base os projetos anteriores. Por ter um legado considerável (muitas linhas de código escritas), dificilmente decide-se por utilizar uma nova tecnologia que não leve em conta o sistema existente. Conseqüentemente, apesar de uma versão ser considerada um projeto, na realidade, ela está ligada e é dependente de todas as versões anteriores.

Portanto, o projeto de uma versão, em uma visão mais global, abrange o projeto de todas as versões anteriores. Uma nova característica ou a alteração de algo já existente pode necessitar de informações do projeto inicial do sistema, realizado há vários anos. Jarczyk, Loffler e Shipman (1992) ressaltam que um exemplo de tarefa que se beneficiaria muito em

guardar informações sobre decisões de projeto são os projetos de versões de sistemas de software.

O ciclo de vida deste tipo de software difere dos demais pelo fato de que são realizadas alterações muito constantes no sistema (para acertos, mudanças legais, novas funcionalidades, etc.). Geralmente ocorre a liberação de diversas atualizações entre uma versão e outra. Além disso, uma característica deste tipo de sistema é a existência de personalização para atender às necessidades específicas de cada cliente. Esta atividade é considerada um projeto à parte, realizada por uma outra equipe da empresa fornecedora do software ou mesmo por uma equipe de outra empresa.

É muito comum a existência de empresas de consultoria que realizam a implantação e personalização do sistema. Informações sobre as decisões dos projetos do sistema ERP podem ser valiosas para esta fase, evitando inconsistências, trabalhos desnecessários e tentativa de alternativas já descartadas. As pesquisas mais recentes sugerem que a falta de conhecimento do sistema ERP tem sido um assunto de grande importância para muitas organizações (DAVENPORT, 2000; MARKUS et al., 2000).

Não é raro o projetista de uma determinada rotina não estar presente quando houver a necessidade de se fazer alguma alteração na sua funcionalidade. Isso se deve a dois fatores principais: longo período do ciclo de vida do software e a alta rotatividade de pessoas nesta área.

Por todas estas características, o projeto de um sistema ERP constitui um caso muito propício à utilização da técnica de *Design Rationale.* É um projeto complexo, que sofre grande quantidade de alterações, possui um ciclo de vida longo e envolve um grande número de pessoas. Conseqüentemente, será muito útil se for possível armazenar informações com o objetivo de realizar verificações do projeto ou fornecer esclarecimentos aos envolvidos. A Figura 10 mostra um resumo das

características de um sistema ERP e também alguns benefícios da utilização da técnica de *Design Rationale*.

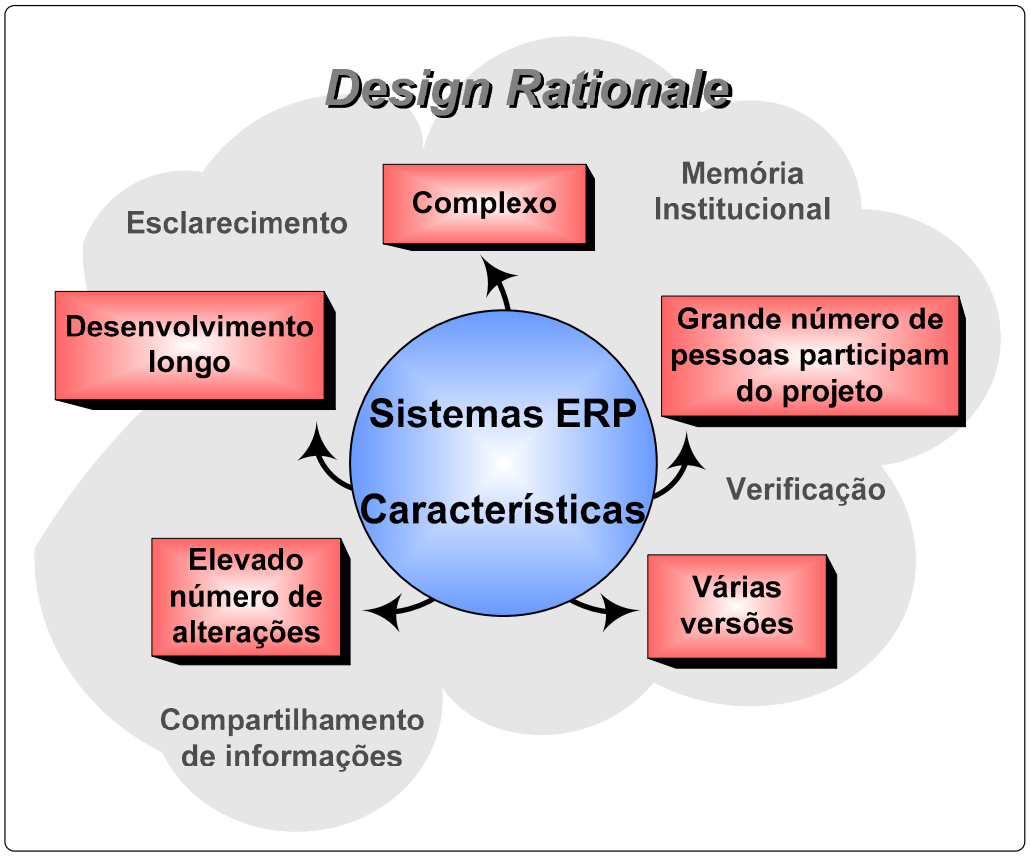

Figura 10 - Características de Sistemas ERP e benefícios do *Design Rationale* (elaborado pela autora)

A documentação das decisões do projeto gera um histórico, facilitando o compartilhamento e o acúmulo de informações. Pode, por exemplo, evitar a tentativa de uma alternativa já descartada no passado. Estas informações serão importantes para outras fases do ciclo de vida do ERP e também para a própria fase de elaboração do projeto, pois é grande a probabilidade de que seja realizada por pessoas que não participaram da definição inicial do sistema.

Adicionalmente, o sistema ERP tem uma fase de implantação do sistema, na qual são feitas as personalizações para cada empresa. Esta é a fase mais crítica, que muitas vezes gera descontentamento tanto da equipe que está

implantando, como da equipe formada por funcionários da empresa cliente. Pesquisas recentes têm revelado uma diminuição considerável do nível de satisfação das implantações de sistemas ERP no período de 1998 a 2000 (SKOK; LEGGE, 2001). Acredita-se que grande parte do problema seja causada por falta de informação e comunicação adequada. O *Design Rationale* contribuirá especialmente em um dos maiores problemas de todo o ciclo de desenvolvimento do sistema ERP.

É muito difícil, provavelmente até impossível, prever todas as decisões de projeto que serão questionadas (GRUBER; RUSSEL, 1991). Por isso, um sistema de *Design Rationale* deve ter a habilidade de responder a uma grande quantidade de questões sobre o projeto. Uma outra tática é concentrar-se nas informações de uma parte específica do projeto. Esta estratégia pode ser mais eficaz nos casos em que há um histórico de projetos anteriores e é possível prever quais são as partes do projeto com maior probabilidade de terem suas informações questionadas.

Esta última opção parece ser muito adequada a projetos de sistemas ERP. Por histórico, geralmente é possível determinar os módulos que mais sofrem atualizações, quais funções apresentam mais problemas, quais processos são críticos, etc. Então, pode-se concentrar os esforços em partes específicas do projeto.

4.2 Análise de casos reais

Neste item, são apresentados casos reais de decisões de projeto de sistema ERP. Primeiramente, tentou-se coletar os dados de maneira escrita, através do envio de e-mails a coordenadores de desenvolvimento de sistema ERP de três grandes empresas. O texto enviado encontra-se no APÊNDICE A. Do total de trinta e três pessoas, apenas uma pessoa respondeu à

solicitação. Percebendo que o problema era o entendimento do que estava sendo solicitado, a coleta dos dados passou a ser feita através de entrevistas informais.

De uma maneira geral, os entrevistados mostraram certa dificuldade em lembrar exemplos. Mesmo quando um caso era lembrado, nem todas as alternativas ou razões da decisão eram lembradas. Quase metade dos entrevistados solicitou ajuda de outra pessoa da equipe para lembrar de algum detalhe. No total, treze pessoas foram entrevistadas. Em uma pesquisa realizada por Tang et al. (2005), 74% dos entrevistados afirmaram que esqueciam as razões relacionadas às decisões de projeto.

Após as entrevistas, vinte e dois casos foram coletados. Quatro casos foram selecionados para detalhamento neste trabalho. Os demais têm uma representação bastante semelhante e, praticamente, não acrescentam nenhuma informação extra para a proposta do modelo.

A seguir, estes quatro casos de decisão de projeto de sistema ERP são apresentados.

4.2.1 Caso 1: Linguagem de programação

Muitas rotinas são escritas especificamente para um determinado cliente e acabam ficando sem padronização. Além disso, não há a preocupação de se escrever um código que possa ser utilizado no sistema padrão, que seja independente do banco de dados, do idioma, do número de filiais, etc. Algumas vezes, decide-se utilizar estas rotinas no módulo padrão e então, é necessário fazer a compatibilização para funcionar no sistema base.

Para facilitar, decidiu-se desenvolver um programa que realizasse as principais correções nos programas fontes. Este programa poderia ser escrito em uma das seguintes linguagens:

1. Linguagem C: Desenvolvimento em um tempo maior, pois a equipe não tinha experiência com a linguagem. Estimou-se dois meses. Linguagem apresenta, em geral, melhor desempenho que as demais.

2. Linguagem Pascal (Delphi): Desenvolvimento em menor tempo, pois além de ter recursos com conhecimento da linguagem, o ambiente de desenvolvimento é bem amigável. Estimou-se o prazo em uma semana.

3. Linguagem Advpl: É a linguagem proprietária da empresa, utilizada tanto na implementação dos módulos básicos como nas personalizações de clientes. O tempo de desenvolvimento foi estimado em três semanas. É maior que o tempo necessário para se desenvolver em Delphi, pois apesar da equipe ter domínio da linguagem, o ambiente de desenvolvimento não é tão amigável, principalmente na parte de projeto de interface. A grande vantagem é a utilização de uma linguagem proprietária. Utilizando tecnologia própria, provavelmente, facilitará sua incorporação aos demais aplicativos da empresa.

A representação do *Design Rationale* para este caso é ilustrada na Figura 11. Ela foi construída utilizando-se o modelo IBIS.

Considerando-se que cada linha contínua verde ligada à alternativa indica um argumento que dá suporte a esta alternativa e que uma linha tracejada vermelha indica que esta alternativa possui um argumento que se opõe a ela, pode-se verificar porque a linguagem Advpl foi a escolhida.

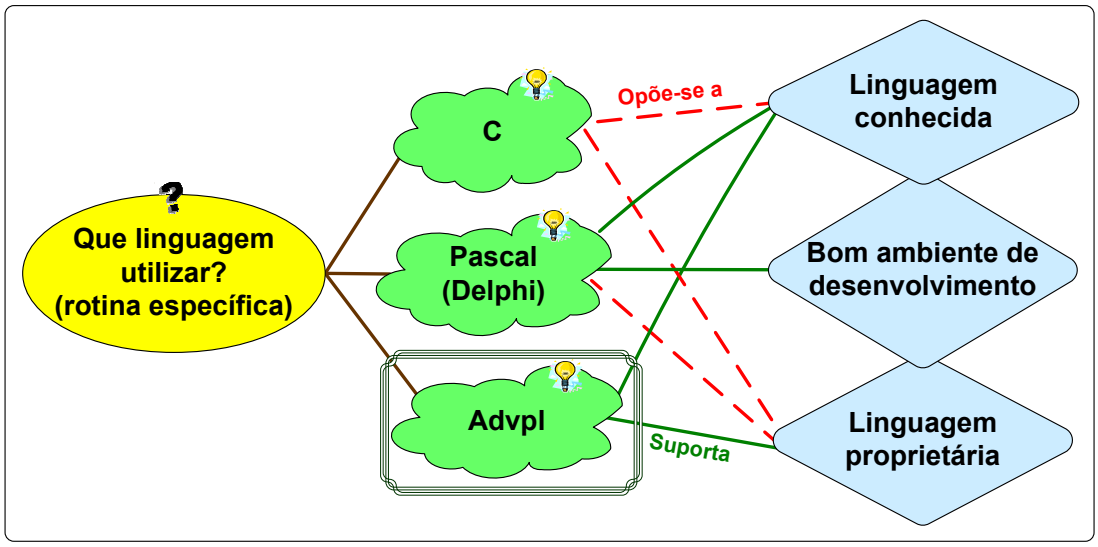

Figura 11 - Representação do *Design Rationale* para o Caso 1

4.2.2 Caso 2: Análise de Crédito

O módulo de Automação Comercial, conforme descrito no item 2.3, gerencia desde o controle operacional relacionado à frente da loja (caixas e atendimentos) até a gestão financeira, de estoques e compras.

Neste módulo, existe um processo de análise de crédito que é realizado no momento do pagamento da mercadoria comprada se a operação for realizada com cheque. Consiste em verificar a situação de crédito do cliente no mercado.

Durante a fase de análise do projeto, duas alternativas foram levantadas para a realização deste processo:

1. Consulta via *Web service*: cada terminal que necessita da informação envia um processo solicitando a consulta. Este processo pode demorar certo tempo. Por isso, é comum haver mais de uma requisição sendo aguardada ao mesmo tempo, pois normalmente uma nova requisição de consulta só pode ser enviada novamente

quando a primeira retornar. Esta solução é de implementação mais simples e fornece segurança, evitando que uma compra seja efetivada sem que haja a verificação do crédito. Por outro lado, pode acontecer de um cliente mudar para um modo de pagamento que não necessite da verificação (dinheiro, por exemplo). Nesta solução, depois que o pedido de análise de crédito foi solicitado ao sistema, é necessário aguardar o retorno do processo para então finalizar a compra. Além disso, outras pessoas podem estar aguardando para efetuar a análise de crédito e terão que aguardar a realização de uma consulta que nem será utilizada.

2. Semáforo local: em vez de simplesmente enviar as requisições, o sistema faz uma fila local de consultas. Caso uma requisição de análise de crédito seja cancelada, o sistema verifica se a requisição está na fila. Se não estiver, nada é feito. A consulta já está sendo realizada. Se estiver na fila, retira a requisição e continua a operação de compra. Neste caso, é necessário implementar esta lógica da fila, mas oferece uma solução mais flexível e com desempenho bem melhor na prática.

A Figura 12 mostra a representação gráfica do problema utilizando o modelo IBIS. Embora seja possível representar todas as informações na figura, não fica claro qual a alternativa escolhida e por qual razão, já que as duas alternativas apresentam o mesmo número de argumentos que dão suporte e também o mesmo número de argumentos que se opõe à alternativa (dois).

Normalmente, um argumento é mais importante que outro. Em vista disso, uma alternativa pode suportar até menos argumentos que outra alternativa, mas mesmo assim, ela pode ser a escolhida. Isso pode acontecer porque os argumentos que ela satisfaz são mais importantes que os demais.

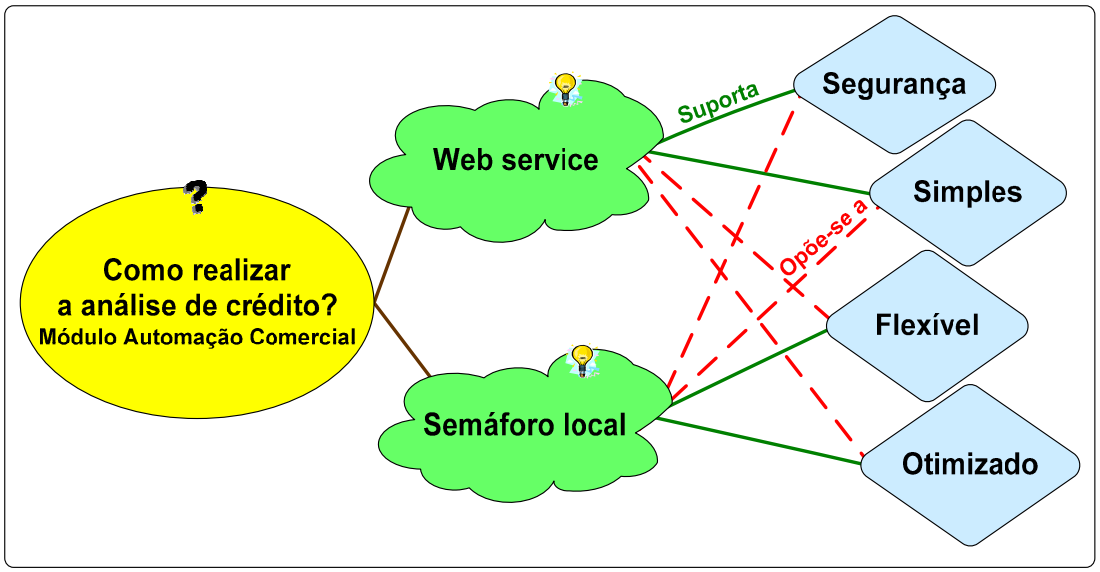

Figura 12 - Representação do *Design Rationale* para o Caso 2

Neste caso, a opção escolhida foi a utilização de um semáforo local, otimizando o processo.

4.2.3 Caso 3: Recursos Humanos – Tabela de Apontamentos

No módulo de Recursos Humanos, existe uma rotina que faz o controle dos apontamentos de entrada e saída de cada funcionário. Basicamente, é feita uma comparação entre uma tabela com os horários padrão de trabalho do funcionário e a tabela de apontamentos mensal de entradas e saídas do funcionário. Com base nesta análise, é possível calcular descontos e horas extras no final do mês.

Pelo grande volume de dados, esta rotina precisa ser realizada com desempenho muito bom. O ponto principal está na definição do modo de manipulação destas tabelas:

1. Tabela Física: dados ficam em uma tabela do banco de dados. Apesar do processamento direto ser mais lento do que ter os dados

em memória, é possível usufruir diversas funcionalidades do banco de dados (integridade, gatilhos, índices, etc.).

2. Dados em memória: dados são lidos do banco de dados e ficam em memória durante a análise. Esta opção requer implementações extras, como, por exemplo, carga dos dados, validações e ordenação dos dados. A vantagem é a rapidez no acesso aos dados depois que estão prontos para serem manipulados.

A Figura 13 mostra a representação do *Design Rationale* para este caso. Observa-se que a pergunta principal gera uma outra questão, que é como obter o melhor desempenho.

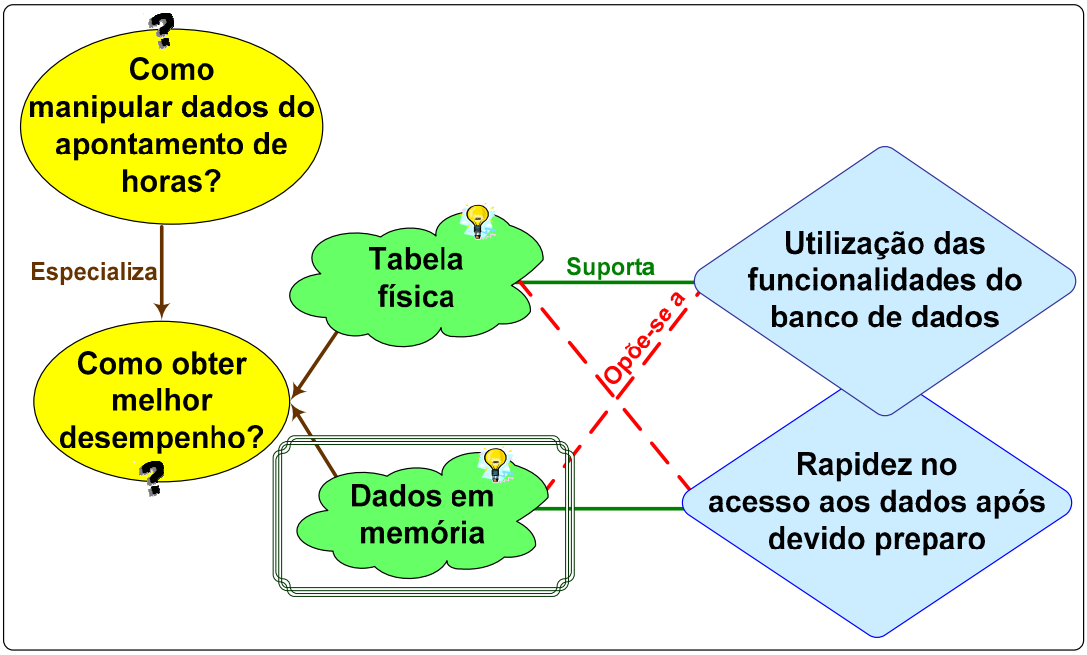

Figura 13 - Representação do *Design Rationale* para o Caso 3

Inicialmente, a alternativa “Dados em memória” foi a escolhida.

Este caso não acrescenta nenhum elemento novo fora a especialização da questão em outra questão - “Como obter melhor desempenho?“. Esta

característica já estava presente no gIBIS, uma extensão do modelo IBIS, mostrada na Figura 7.

O interessante deste caso é que o entrevistado relatou que hoje a opção escolhida não é mais a melhor. O aumento de funções para tratar exceções acabou tornando a opção dos dados em memória mais lenta. Deste modo, a alternativa “Tabela física” passou a ser a melhor solução.

4.2.4 Caso 4: *Business Process Management* (BPM)

O objetivo do *Business Process Management* (BPM) é fornecer maior visibilidade e transparência dos processos e conseqüente possibilidade de monitoramento e auditoria. Uma das características desta ferramenta é apresentar flexibilidade para alterações e otimizações de processos conforme a demanda.

O *Business Process Management* (BPM) permite a modelagem de processos, a parametrização de regras de negócios associadas ao fluxo, definição e restrição de acesso de usuários quanto às habilidades e atividades a serem desenvolvidas e o controle sobre características de execução do processo (tempo e custo de execução). Além disso, permite o desenvolvimento de aplicações informatizadas, chamada quarta camada de software, integradas ou não a sistemas legados, tais como: sistemas proprietários, ERP, CRM, Telefonia, entre outras tecnologias.

Por todas estas características, é que o BPM está cada vez mais se popularizando. No caso de sua utilização com ERP, pode-se ter a funcionalidade integrada ao sistema ou então utilizar uma ferramenta externa. Uma das empresas de ERP analisadas decidiu por sua integração ao ERP padrão.

Durante a fase de análise do projeto do módulo, uma das principais decisões foi o modo de entrada de dados no sistema. Basicamente havia duas opções:

1. Modelo pré-definido: o modelo contendo todas as características que serão informadas no sistema é definido via programação, não podendo ser alterado durante a sua execução. Por isso, é mais fácil de implementar e apresentar as telas, mas é menos flexível, exigindo intervenção da área de TI caso seja necessário realizar alguma alteração. Neste modelo, o sistema vai solicitando as informações ao usuário à medida que ele vai percorrendo as telas.

2. Modelo dinâmico: a definição do modelo de entrada de dados é feita pelo gerente da empresa durante a execução do sistema. Após a realização desta atividade, o ambiente está pronto para a entrada das informações. A idéia é que este gerente entenda do negócio da empresa e não necessariamente precise entender de informática. Conseqüentemente, o modelo é definido dinamicamente e ele pode ser modificado sem a intervenção da equipe de TI. A implementação desta solução é mais complexa e necessita de um validador do modelo definido pelo usuário. Entretanto, esta solução é a que possibilitará a criação de um sistema com características que seguem o conceito de *Business Process Management* (BPM).

A representação do *Design Rationale* do caso relatado no modelo IBIS ficaria conforme mostra a Figura 14. Da maneira como está, não é possível definir qual a melhor alternativa, já que cada uma suporta dois argumentos e opõe-se a outros dois.

Para a solução desta questão, optou-se pela implementação do modelo dinâmico.

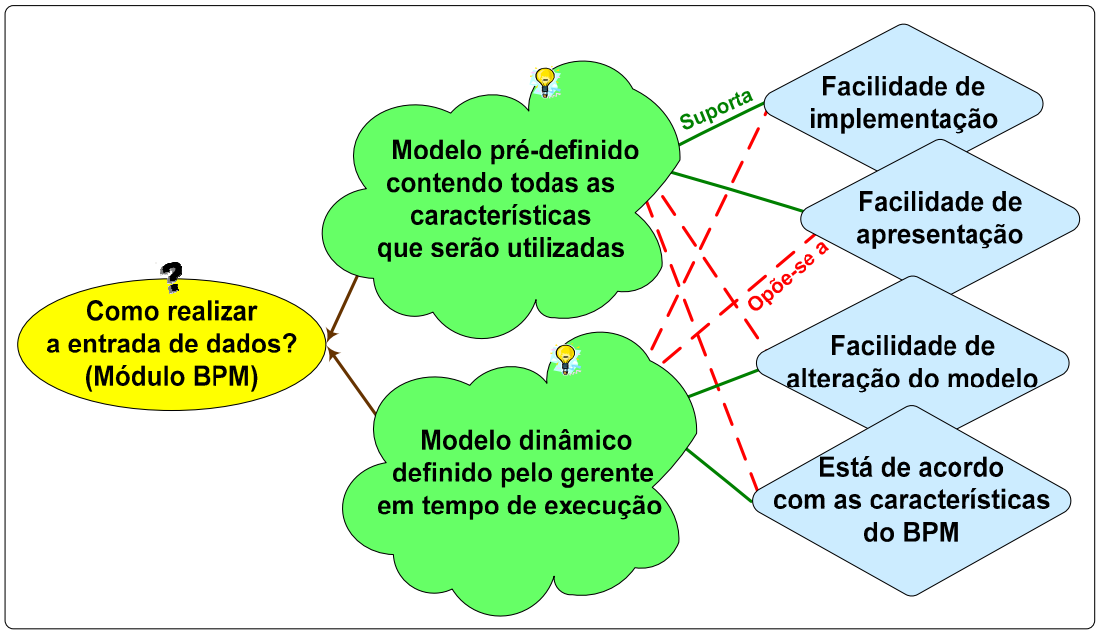

Figura 14 - Representação do *Design Rationale* para o Caso 4

4.3 Definição do modelo

Neste item é apresentada a proposta de um modelo para a representação do *Design Rationale*, considerando sistemas ERP. Hu et al. (2000) afirmam que um bom modelo de representação é essencial para uma recuperação efetiva.

O modelo proposto tem o objetivo de capturar as informações de *Design Rationale* na fase de análise de projeto do ciclo de desenvolvimento de um sistema ERP. A recuperação das informações pode ser feita durante todo o ciclo de vida do software, incluindo a própria fase de análise de projeto, a fase de implantação do sistema e o desenvolvimento de outras versões do produto.

Procurou-se a definição de um modelo que pudesse representar os principais elementos encontrados nas decisões de sistemas ERP e que fosse ao mesmo tempo simples e intuitivo. Além disso, o objetivo era propor um modelo que pudesse ser útil durante a fase de análise de projeto,

auxiliando as discussões e escolhas e que também contribuísse para as demais fases do ciclo de vida do sistema, fornecendo uma documentação mais completa. Para satisfazer este último critério, a representação gráfica deveria por si só explicar a escolha de uma determinada alternativa.

Inicialmente a idéia era realizar o trabalho em uma parte do sistema ERP. Entretanto, após a análise das características deste tipo de software e dos casos de decisões relatados no item 4.2, verificou-se que não havia a necessidade de realizar esta limitação. Praticamente não há diferença entre os módulos nas características relevantes a esta pesquisa.

No item 3.7 foram descritas e analisadas as principais características de três modelos de representação de *Design Rationale*: QOC (*Questions, Options and Criteria*), IBIS (*Issue Based Information System*) e DRL (*Decision Representation Language*). Além disso, foi feito um estudo comparando-se as características dos três modelos.

DRL foi descartado, pois sua ênfase é no gerenciamento das tomadas de decisão e de suas dependências. Sua complexidade dificulta sua aplicação em um projeto tão extenso e com grande número de pessoas como os sistemas ERP. A alta rotatividade é outro empecilho de se adotar este modelo. O tempo de treinamento necessário para se entender a nova tecnologia pode acabar inviabilizando a adoção deste modelo. Então, IBIS e QOC pareciam ser os modelos mais apropriados para este caso.

No Caso 1, visto no item 4.2.1, verificou-se que o modelo IBIS era o que melhor representaria as características do exemplo. A justificativa principal para escolha deste modelo como base é que as questões levantadas podem se referir a qualquer questão do projeto e não só ao artefato que está sendo projetado. A representação gráfica foi validada apresentando-a aos próprios entrevistados.

Para a grande maioria dos casos, verificou-se que somente o modelo IBIS não era suficiente para apresentar um modelo auto-explicativo, principalmente com relação à alternativa escolhida. Levando-se em conta apenas o número de argumentos que são favoráveis a uma alternativa, corre-se o risco de não escolher a melhor alternativa. A solução foi definir pesos para cada argumento. Estes pesos definem o grau de importância dos argumentos para o problema em questão. O intervalo válido de valores é de zero a dez e quanto maior o número, maior a sua importância.

Deste modo, o modelo pôde representar uma característica muito comum nos casos relatados: cada argumento possui uma importância diferente para a solução da questão. A Figura 15 mostra o modelo IBIS com os pesos nos argumentos.

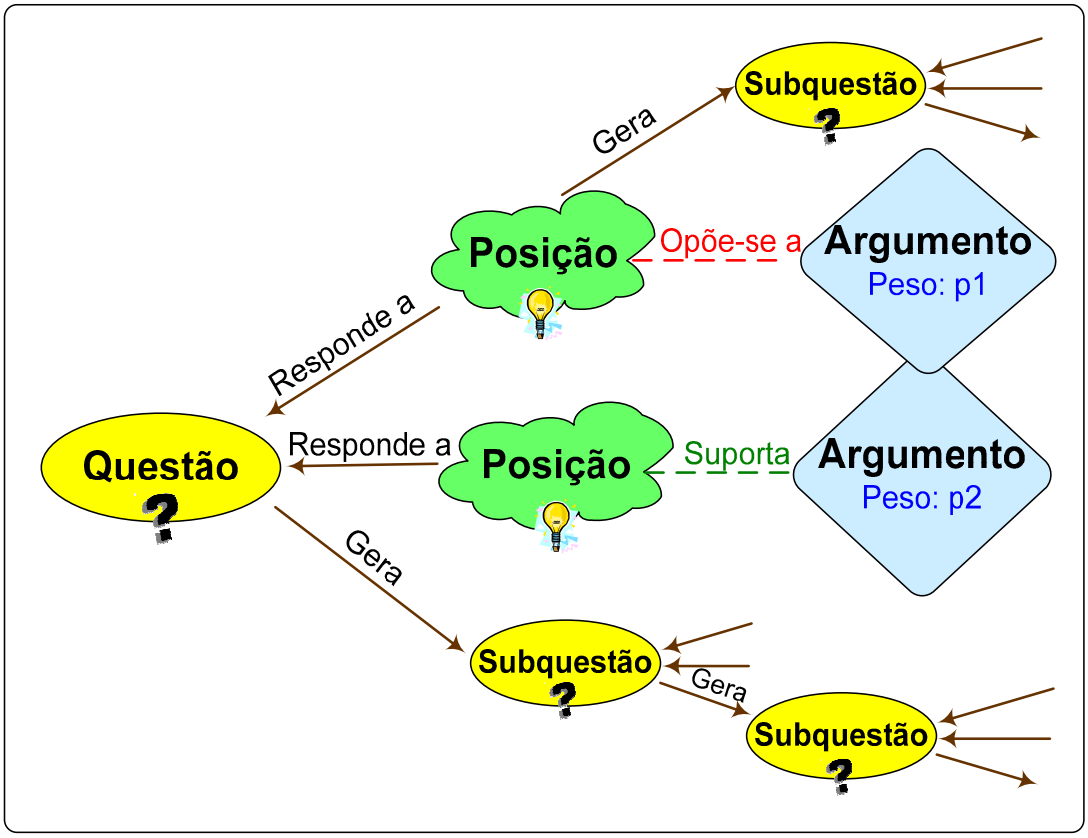

Figura 15 - Modelo de *Design Rationale* para sistemas ERP: inclusão do atributo “Peso”

Finalmente, após a análise de casos como o que foi apresentado no item 4.2.4 (BPM), confirmou-se uma das limitações do modelo IBIS apresentada na literatura: falta da representação explícita dos critérios como elementos do modelo (NGUYEN; SWATMAN; SHANKS, 2000). A inclusão de um elemento específico para representar um critério (requisito ou restrição) eliminou esta limitação. Esta informação, que está presente no modelo QOC, tornou o modelo mais completo. A Figura 16 ilustra a proposta final do modelo de *Design Rationale* para sistemas ERP. Este modelo foi definido através da associação de características dos modelos IBIS e QOC e da inclusão de um atributo de quantificação da importância dos argumentos.

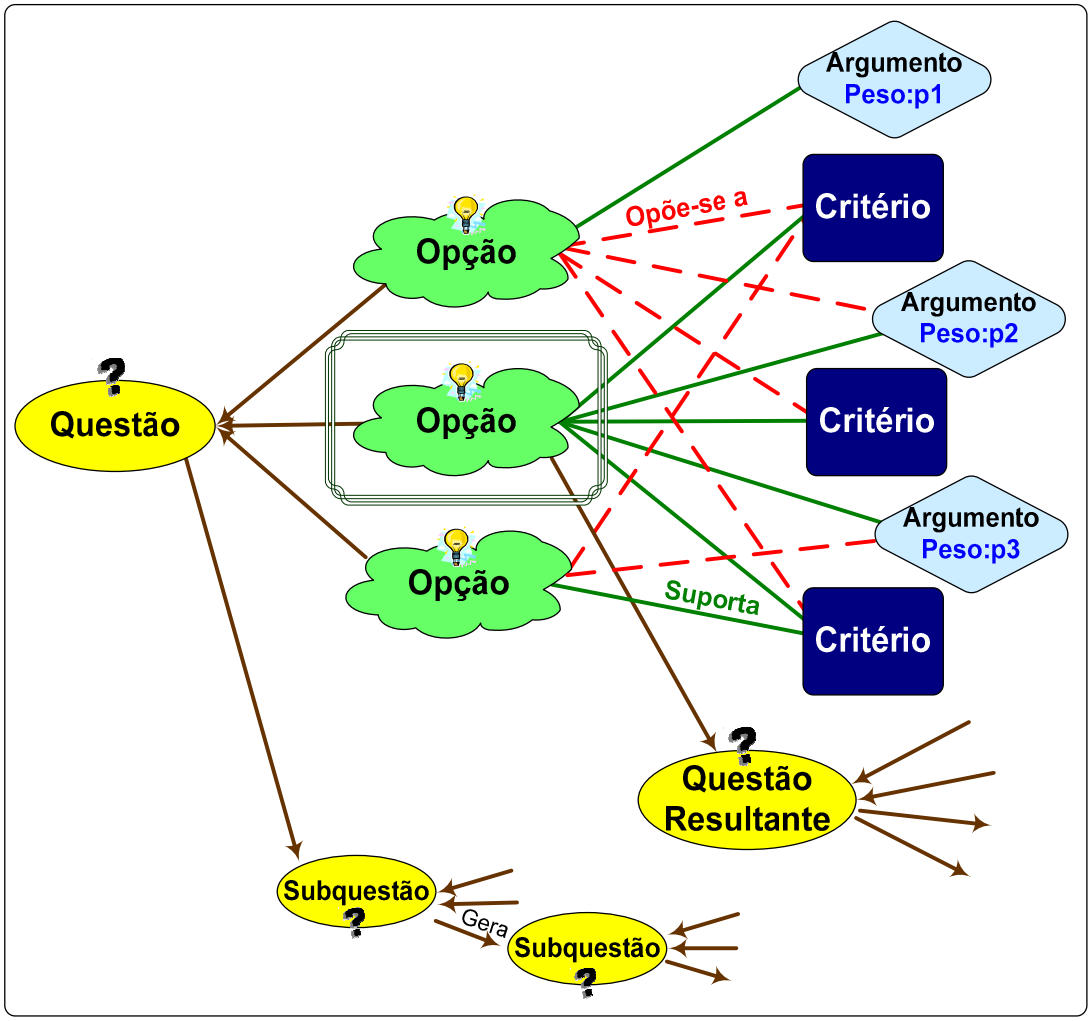

Figura 16 - Proposta de um modelo de *Design Rationale* para sistemas ERP

Foi possível representar adequadamente todos os casos de decisões de projeto de sistemas ERP levantados na pesquisa através deste modelo. A representação gráfica neste novo modelo dos quatro casos apresentados no item 4.2 será realizada a seguir.

4.4 Aplicação do Modelo Proposto aos Casos Reais

Neste item é feita a aplicação do modelo proposto no item anterior aos casos reais apresentados no item 4.2.

4.4.1 Caso 1: Linguagem de programação

Para o Caso 1 não há mudanças na representação do Design Rationale, tendo em vista o novo modelo proposto.

4.4.2 Caso 2: Análise de Crédito

A Figura 17 mostra a representação gráfica do problema utilizando o novo modelo.

Para representar a diferença de importância entre os argumentos, foram utilizados os pesos nos argumentos. Dessa forma, pode-se visualizar porque a alternativa de semáforo local foi a escolhida.

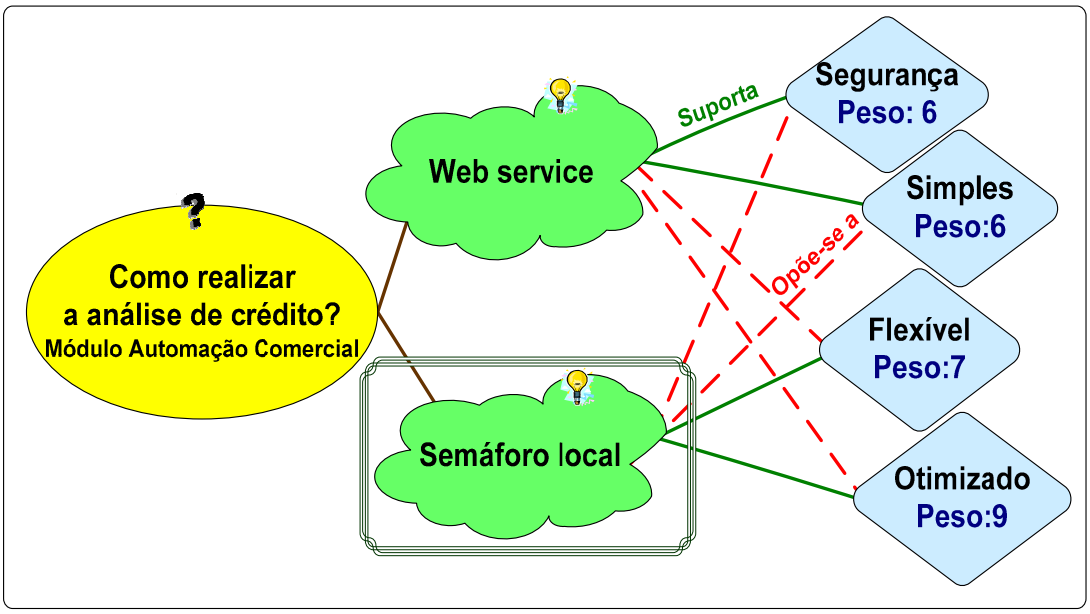

Figura 17 – Representação do *Design Rationale* para o Caso 2

4.4.3 Caso 3: Recursos Humanos – Tabela de Apontamento

A Figura 18 mostra a representação do *Design Rationale* para o caso da tabela de apontamentos. Como o segundo argumento é o que fornece melhor resposta à questão (Peso: 8), a alternativa “Dados em memória” foi a escolhida.

Conforme citado anteriormente, o aumento de funções para tratar exceções acabou tornando a opção dos dados em memória mais lenta. Para refletir esta mudança na representação gráfica, basta alterar o “Peso” do argumento correspondente, conforme ilustrado na Figura 19.

Deste modo, a alternativa “Tabela física” parece a ser a melhor solução.

Este caso reforça a utilidade de se ter a propriedade “Peso” nos argumentos indicando a sua importância na solução da questão levantada. Ela é essencial para o entendimento da mudança de alternativa escolhida.

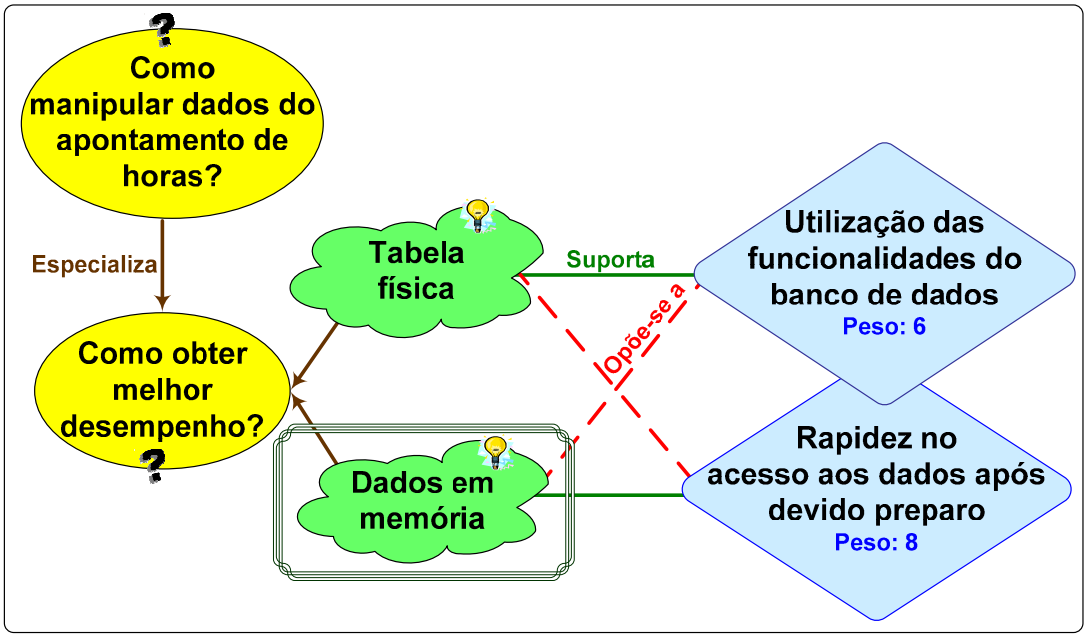

Figura 18 - Representação do *Design Rationale* para o Caso 3

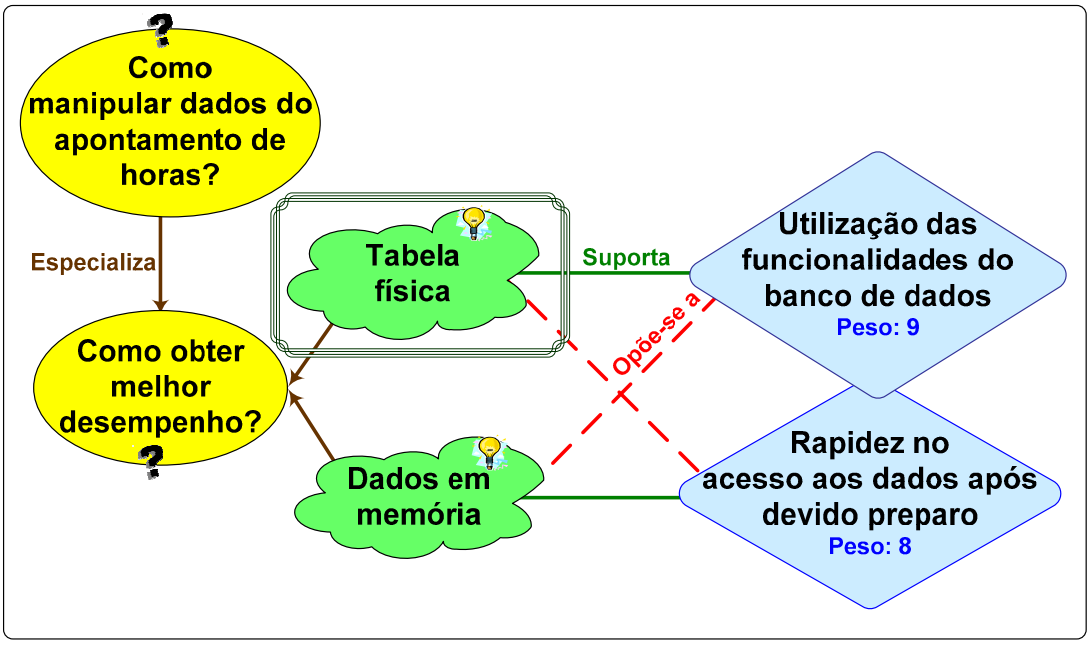

Figura 19 - Representação do *Design Rationale* para o Caso 3 – situação atual

4.4.4 Caso 4: *Business Process Management* (BPM)

A representação do *Design Rationale* do caso relatado no novo modelo ficaria conforme mostra a Figura 20.

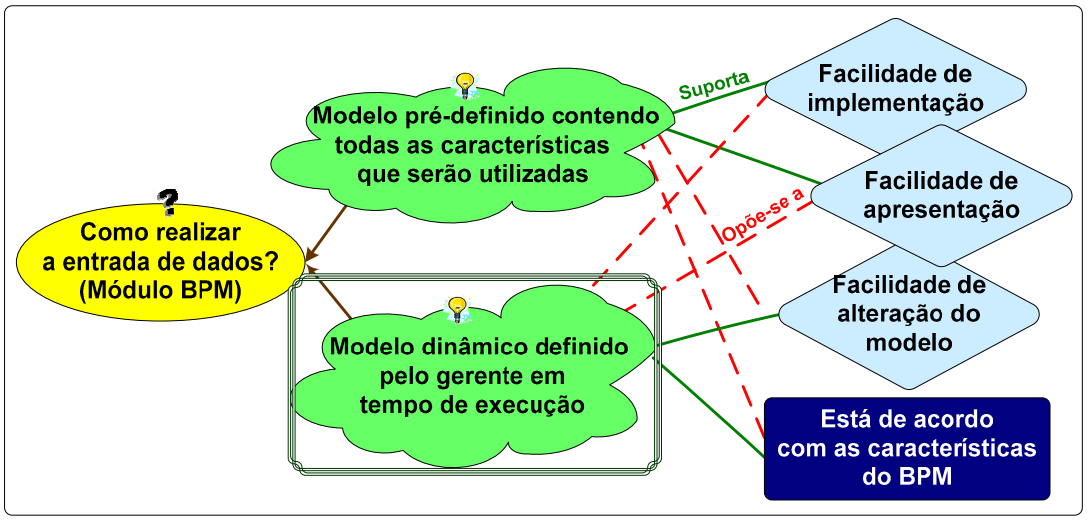

Figura 20 - Representação do *Design Rationale* para o Caso 4

A figura pode ser interpretada mais facilmente com a representação do nó CRITÉRIO. O nó “Está de acordo com as características do BPM” é um requisito do sistema e, portanto, a solução deve suportar este item. Conseqüentemente, o “Modelo dinâmico” é solução do problema.

5 CONCLUSÕES

Neste capítulo descrevem-se as principais considerações resultantes deste trabalho, bem como se apresentam alguns trabalhos futuros possíveis de serem realizados, como conseqüência da pesquisa aqui desenvolvida.

5.1 Considerações Finais

Para muitos pesquisadores de *Design Rationale* o valor óbvio de se capturar as informações sobre as decisões de projeto significava que esta técnica seria rapidamente difundida nas empresas. Não foi o que se constatou na prática. Praticamente não há relatos de sua utilização em projetos reais. Por outro lado, as pesquisas acadêmicas sobre o assunto continuaram crescendo.

Embora existam várias pesquisas sobre *Design Rationale* e *Enterprise Resource Planning* (ERP), separadamente, não foi encontrado nenhum estudo sobre os dois temas em conjunto. Desta forma, o estudo da aplicação da técnica do *Design Rationale* aos Sistemas ERP demonstrou o grande valor de tal forma de utilização. As justificativas de projeto podem ser documentadas de forma bastante completa e precisa, auxiliando sobremaneira os projetistas desse tipo de sistema.

Outro fator importante está na constatação de algumas deficiências encontradas em duas formas de modelagem do *Design Rationale*, o que possibilitou a proposição de uma adaptação à forma de modelagem, que se demonstrou bastante apropriada para registrar as razões de projeto de Sistemas ERP.

Após a elaboração da proposta do modelo de *Design Rationale*, foi possível constatar que realmente a maior dificuldade não está na definição de processos ou modelos. Os desafios técnicos são pequenos perto dos desafios humanos. O modelo proposto para sistemas ERP, apesar de simples, atende a todos os casos de uso coletados.

Deve-se ressaltar que o modelo proposto visou à captura de informações de *Design Rationale* na fase elaboração de projeto do ciclo de desenvolvimento de um Sistema ERP. A recuperação das informações pode ser feita durante todo o ciclo de vida do Sistema, incluindo a própria fase de elaboração do projeto, além da fase de implantação, bem como o desenvolvimento de outras versões do produto.

O modelo proposto é capaz de representar modelos que representem os principais elementos encontrados nas decisões de Sistemas ERP, auxiliando nas principais decisões de projeto, fornecendo ainda uma documentação mais completa.

5.2 Trabalhos Futuros

Há diversos trabalhos que podem ser desenvolvidos como seqüência desta dissertação. Dentre eles, pode-se citar a implementação de uma ferramenta baseada no modelo proposto e, integrando todas as características citadas. Tal ferramenta poderia, inicialmente, ser utilizada em uma aplicação experimental, para, posteriormente, ser utilizada em um ambiente real. Tang et al. (2005) afirma que as ferramentas de suporte para captura e recuperação de *Design Rationale* ainda são inadequadas para uso em ambientes reais.

Acredita-se que para o desenvolvimento de uma ferramenta que realmente seja utilizada na prática, há a necessidade de pesquisar os fatores humanos que geram a resistência à sua utilização. Conhecendo os motivos, provavelmente será possível definir modelos ou processos para minimizar estas resistências. Esta é uma outra área que ainda precisa ser explorada.

O foco deste trabalho concentrou-se na captura de informações durante a fase de análise de projeto. Uma outra fase que merece destaque e que pode ser o foco de futuras pesquisas concentra-se na implantação de sistemas ERP, por ser a fase de projeto que mais apresenta problemas. Provavelmente a captura de decisões desta fase será também de muita utilidade.

Finalmente, pode ser pesquisado e ser feito na prática o emprego da técnica de *Design Rationale* em outras aplicações da Engenharia de Software como, por exemplo, o desenvolvimento de sistemas de Inteligência Empresarial, de Relacionamento com o Cliente, ou mesmo de sistemas críticos, nos quais é vital a correção do software.

REFERÊNCIAS

BASU, V.; LEDERER, A. L. **An agency theory model of ERP implementation**, Proceedings of the 2004 SIGMIS conference on Computer personnel research: Careers, culture, and ethics in a networked environment, Apr. 22-24, 2004, Tucson, AZ, USA.

BINGI, P.; SHARMA, M. K.; GODLA, J. K. **Critical Issues Affecting an ERP Implementation**, Information Systems Management, vol. 16, no. 3, 1999, pp. 7-14.

BOSCH, J. **Software Architecture: The Next Step**, Software Architecture: First European Workshop, EWSA 2004, St Andrews, UK., pp 194-199, 2004.

BOWERSOX, J., DAUGHERTY, P., DRÖGE, C., GERMAIN, R., ROGERS, D. **Logistical Excellence: it's not business as usual**. Burlington, Digital Equipment Corporation, 1992.

BREHM, L.; HEINZL, A.; MARKUS, M. L. **Tailoring ERP systems: a spectrum of choices and their implications**. System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference, 9 pp.-, 3-6 Jan. 2001.

BROWN, W. **Enterprise resource planning (ERP) implementation planning and structure: a recipe for ERP success**. Proceedings of the 32nd annual ACM SIGUCCS conference on User services, p.82-86, October 10-13, 2004, Baltimore, MD, USA.

BROWN, C.; VESSEY, I. **ERP Implementation Approaches: Toward a Contingency Framework**, Proceedings of Twentieth International Conference on Information Systems, Charlotte, North Carolina, 1999, pp. 411-416.

BURGE, J. E. **Software Engineering Using design RATionale**, PhD Dissertation, CS Dept., WPI, May 2005. Disponível em: http://www.wpi.edu/Pubs/ETD/Available/etd-050205-085625/. Acesso em: 10/12/2005.

BURGE, J. E.; BROWN, D. C. **Design Rationale Types and Tools**. 1998. Disponível em: http://www.cs.wpi.edu/Research/aidg/DR-Rpt98.html. Acesso em: 20/01/2004.

BURGE, J. E.; BROWN, D. C. **Reasoning with Design Rationale.** In Proceedings of the Artificial Intelligence Design Conference, 2000.

CARROLL, J. M.; ROSSON, M. B. **Deliberated Evolution: Stalking the View Matcher in Design Space.** Human-Computer Interaction, 6, p. 281-318, 1991. Also in T.P. Moran and J.M. Carroll (Eds.) Design rationale: Concepts, techniques, and use, (pp. 107-145). Hillsdale, NJ: Lawrence Erlbaum Associates, 1996.

CHANG, S. **ERP life cycle implementation, management and support: implications for practice and research**, System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference, 10 pp.-, 5-8 Jan. 2004.

CONKLIN, J. **Design Rationale and Maintainability**. System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, vol.2, pp.533-539, Jan. 1989.

CONKLIN, J.; BEGEMAN, M. L. **gIBIS: A Hypertext Tool for Exploratory Policy Discussion**. Proceedings of the 1988 Conference on Computer Supported Cooperative Work (CSCW-88), ACM, Portland, Oregon, pp. 140-152, 1988.

CONKLIN, J.; BURGESS-YAKEMOVIC, K. **A Process-Oriented Approach to Design Rationale**, in Design Rationale Concepts, Techniques, and Use, T. Moran and J. Carroll, (editors), Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428, 1995.

CORRÊA, H. C.; GIANESI, I.; CAON, M. **Planejamento, programação e controle da produção: MRP II/ERP: conceitos, uso e implantação**. São Paulo: Gianesi Corrêa & Associados, Atlas, 1997.

CUNHA, M. A. L. **Gestão integrada de processos de negócio.** Simpósio de Administração da Produção, Logística e Operações Industriais - Anais. São Paulo: FGV, 1998. p. 184-194.

DAVENPORT, T. H. **Mission Critical** – Realizing the Promise of Enterprise Systems, Boston, MA: Harvard Business School Press; 2000.

DECK, M. **Managing Process Diversity While Improving Your Practices**. IEEE Software - May/June 2001.

DICTIONARY.COM - multi-source dictionary search service produced by Lexico Publishing Group, LLC, a leading provider of language reference products and services on the Internet. Disponível em http://dictionary.reference.com/. Acesso em 01/08/2006.

DUTOIT, A. et al. **Rationale Management in Software Engineering**, Editors, Springer, 2006.

FISCHER, G. et. al. **Making argumentation serve design**. In T. Moran and J. Carroll, editors, Design Rationale Concepts, Techniques, and Use, pages 267-294. Lawrence Erlbaum Associates, 1995.

FRANCISCO, S. D. **DocRationale: uma ferramenta para suporte a Design Rationale de artefatos de software**. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos - SP, 2004.

GARCIA, A.; HOWARD, H.; STEFIK, M. **Active Design Documents:** A New Approach for Supporting Documentation in Preliminary Routine Design, Tech. Report 82, Stanford Univ. Center for Integrated Facility Engineering, Stanford, Calif., 1993.

GARCIA, A.; HOWARD, H.; STEFIK, M. **Design Rationale for Collaboration:** The Active Document Approach, Submitted to Research in Engineering Design Journal, Mar. 1994.

GIBSON, N.; HOLLAND, C.; LIGHT, B. **Enterprise Resource Planning: A Business Approach to Systems Development**, Proceedings of 32nd Hawaii International Conference on System Sciences, Hawaii, 1999.

GLOSSÁRIO DE TERMOS DA QUALIDADE. Tecnologia da Informação, Qualidade e Produtividade no Setor de Software - 2005. Disponível em: http://www.mct.gov.br/Temas/info/Dsi/qualid99/99anexo2.htm. Acesso em 01/05/2006.

GRUBER, T. R. **Model-based Explanation of Design Rationale**, in Proceedings of the AAAI-90 Explanation Workshop, Boston, July 30, 1990.

GRUBER, T. R.; RUSSEL, D. M. **Design Knowledge and Design Rationale: A framework for representation, capture, and use**. Technical Report KSL 90-45, Knowledge Systems Laboratory, Stanford, California, 1991, 40p.

HEHN, H. F. **Peopleware: como trabalhar o fator humano na implementação de sistemas integrados de informação (ERP)**. São Paulo: Editora Gente, 1999.

HIROSE, A.; CANNON, D.; LEIFER, L. **Development of a Prototype Design Process Recorder Based on Hypergraphs**, 6th International Conference on Design Theory and Methodology, Minneapolis, MN, USA, 1994, vol. 68, pp. 259-271, publ. American Society of Mechanical Engineers, Design Engineering Division, ASME, New York, NY, USA, 1994.

HOLLAND, C.; LIGHT, B.; GIBSON, N. **A Critical Success Factors Model for Enterprise Resource Planning Implementation**. In Proceedings of The European Conference in Information Systems, Copenhagen, pp. 273- 287, June 1999.

HOLLAND, C.; LIGHT, B.; KAWALEK, P. **Beyond enterprise resource planning projects:** Innovative strategies for competitive advantage, Proceedings of 7th European conference on information systems - ECIS'99, Copenhagen, Denmark, pp. 288- 301, 1999.

HU, X. et al. **A Survey on Design Rationale**: Representation, Capture and Retrieval. Engineering with Computers: An Int'l Journal for Simulation-Based Engineering, v. 16, p. 209–235, 2000.

JARCZYK, A.; LOFFLER, P.; SHIPMAN, F. **Design Rationale for Software Engineering:** A Survey: In Proceedings of 25th Annual Hawaii International Conference on System Sciences, 8-10 Jan. 1992.

KANJANASANPETCH, P.; IGEL, B., **Managing knowledge in enterprise resource planning (ERP) implementation**, Engineering Management Conference, 2003. IEMC '03. Managing Technologically Driven Organizations: The Human Side of Innovation and Change, pp. 30- 35, 2-4 Nov. 2003.

KELLY, S.; HOLLAND, C.; LIGHT, B. **Enterprise resource planning: A business approach to systems development**, Proceedings of 5th Conference on information systems, AMCIS 1999, Milwaukee, Wisconsin, USA, pp. 785- 787.

KIRCHMER, M. **Business process oriented implementation of standard software:** how to achieve competitive advantage quickly and efficiently? Berlin: Springer, 1998.

KUNZ, W.; RITTEL, W. J. **Issues as Elements of Information Systems**, Working Paper No 131, University of California, Berkeley, 1970, reprinted in 1979. Disponível em: http://www-iurd.ced.berkeley.edu/pub/WP-131.pdf . Acesso em: 06/12/2005.

LAKIN, F. et al. **The electronic design notebook**: performing medium and processing medium, vol. 5, pp. 214-226, 1989.

LARA, S. M. A. **Um Suporte à captura informal de design rationale**. São Carlos-SP, Dezembro de 2005. 130f. Dissertação (Mestrado). Instituto de Ciências Matemáticas e de Computação de São Carlos, Universidade de São Paulo.

LEE, J. **SIBYL: A qualitative design management system**. In P.H. Winston and S. Shellard, eds., Artificial Intelligence at MIT: Expanding Frontiers, Cambridge MA: MIT Press, pp. 104-133, 1990a.

LEE, J. **SIBYL: A Tool for Managing Group Decision Rationale**. In Proceedings of the CSCW'90 Conference, ACM Press, New York, 79-92 , 1990b.

LEE, J. **Design rationale Systems: Understanding the Issues.** IEEE Expert, Vol. 12, no. 3, pp78-85. May/June 1997.

LEE, J.; LAI, K. **A Comparative Analysis of Design Rationale Representations**, Human-Computer Interaction Special Issue on Design Rationale, 1991, 6 (3-4) pp. 251-280.

LEE, J.; LAI, K. **What's in Design Rationale?** in Design Rationale - Concepts, Techniques, and Use, T. Moran and J. Carroll, Eds. New Jersey: Lawrence Erlbaum, 1996, pp. 21-51.

LEIFER, L. J. **Instrumenting the Design Process:** for Real-time Text-graphic Design Process Records. International Conference on Engineering Design, ICED'91, Zuerich, 1991, Ed. V. Hubka, vol. 1, pp. 314-321.

LOURIDAS, P.; LOUCOPOULOS, P. **A Generic Model for Reflective Design**, ACM Transactions on Software Engineering and Methodology (TOSEM), v.9 n.2, p. 199-237, Apr. 2000.

LOZINSKY, S. **Software: Tecnologia do negócio** - Em busca de benefícios de sucesso na implantação de pacotes de software integrados. Rio de Janeiro: Imago,1996. 242p.

MACAULAY, D. **The Way Things Work**. Houghton-Mifflin, New York, NY., 1989.

MACLEAN, A. et al. **Questions, options and criteria: Elements of design space analysis**. Human-Computer Interaction, 6(3&4), p. 201-250, 1991.

MADU, C. N.; KUEI, C. **ERP and Supply Chain Management** - Chi Publishers, 214 pp, ISBN 0-9676023-4-3, 2004.

MARKUS, M. L.; TANIS, C. **The Enterprise Systems Experience - From Adoption to Success**, in R.W. Zmud, ed., Framing the Domains of IT Research: Glimpsing the Future Through the Past, Cincinnati, OH: Pinnaflex Educational Resources, Inc., 2000, pp. 173-207.

MARKUS, M. L. et al. **Learning from adopters' experiences with ERP: problems encountered and success achieved**. Journal of Information Technology, 15(4), pp. 245-265, 2000.

MCKERLIE, D.; MACLEAN, A. **Reasoning with Design Rationale: Practical Experience with Design Space Analysis**. Design Studies, 15, p. 214-226, 1994.

MICROSIGA SOFTWARE S/A. Página da Internet da empresa Microsiga – empresa brasileira do grupo Totvs. Disponível em: www.microsiga.com.br. Acesso em 13/09/2006.

MILLER, A. **Organizational Change**. IEEE Software - May/June 2001.

MONK, S.; SOMMERVILLE, I.; PENDARIES, J. M.; DURIN, B. **Supporting Design Rationale for System Evaluation**. In: Proceedings of the Fifth European Software Engineering Conference, Barcelona, Espanha, 1995, p. 307–323.

NBR ISO/IEC 12207, **ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS** - Tecnologia de Informação - Processos de ciclo de vida de software. Rio de Janeiro: ABNT, 1998.

NEILL C. J.; LAPLANTE P. A. **Requirements Engineering: The State of the Practice**. IEEE Software. November/December 2003, 40-45.

NGUYEN, L.; SWATMAN, P. A.; SHANKS, G. **Supplementing Process-Oriented with Structure-Oriented Design Explanation within Formal Object-oriented Method**. Software Engineering Conference, 1998. Proceedings. 1998 Australian, no.pp.118-132, 9-13 Nov 1998.

PAIVA, D. M. B.; FORTES, R. P. M. **Model for Academic Software Development Including Design Rationale Elements**. In: I IFIP Academy on the State of Software Theory and Practice - PhD Colloquium, 2005, Porto Alegre, RS. Proceedings of I IFIP Academy on the State of Software Theory and Practice - PhD Colloquium, 2005. v. 1. Short Paper.

POTTS, C.; TAKAHASHI, K.; ANTON, A. **Inquiry-based requirements analysis**. IEEE Software, March 1994, p. 21-32.

RIBBERS, P. M. A; SCHOO, K.-C., **Designing complex software implementation programs**, System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference, no.pp. 3391- 3401, 7-10 Jan. 2002.

RITTEL, H. **On the Planning Crisis: Systems Analysis of the 'First and Second Generations'**, in Proceedings of the Systems Analysis Seminar, European Association of National Productivity Centers, 1971, 1972.

RITTEL, H.; WEBBER, M. **Dilemmas in a General Theory of Planning**, Policy Science, Vol. 4, pp. 155-169, 1973.

ROSSI, M. et al. **Method Rationale in Method Engineering**. Proceedings of the HICSS-33, Maui, HI, IEEE Computer Society, 2000.

SAUER, T. **Using Design Rationales for Agile Documentation**. Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2003 (WETICE'03).

SCHEER, A.; HABERMANN F. **Making ERP a success**. Communications of the ACM 2000, 43(4): 57-61.

SCHNEIDER, K. **Rationale as a By-Product**, Rationale Management in Software Engineering, Dutoit, A., McCall, R., Mistrik, I., and Paech, B., Editors., Springer, 2006.

SCOTT, B. **Manufacturing planning systems**. London. McGraw-Hill. 1994.

SELDIN, R.; FERRUCCIO, M. A. R.; CAULLIRAUX, H. M. **O Papel da Cultura Organizacional na Implantação de Sistemas Integrados de Gestão – uma Abordagem sobre Resistência à Mudanças**. VI SIMPOI - Simpósio de Administração da Produção, Logística e Operações Internacionais – 2003 - São Paulo.

SHIPMAN, F.; MCCALL, R. **Integrating different perspectives on Design Rationale:** Supporting the Emergence of Design Rationale from Design Communication. Artificial Intelligence in Engineering Design, Analysis, and Manufacturing, v. 11, n. 2, p. 141–154, 1997.

SHUM, S. B. **Cognitive Dimensions of Design Rationale**. In D. Diaper, & N. Hammond (Eds.), People and Computers VI, Proceedings of HCI'91, Cambridge University Press, 1991a.

SHUM, S. B. **A Cognitive Analysis of Design Rationale Representation**. Tese de Doutoramento, University of York, 1991b.

SHUM, S. B.; HAMMOND, N. **Argumentation-Based Design Rationale: What Use at What Cost?** International Journal of Human-Computer Studies, 40, 4, 1994, pp. 603-652.

SHUM, S. B.; MACLEAN, A.; BELLOTTI, V. M. E. **Graphical argumentation and design cognition**, Human-Computer Interaction, 1997.

SJÖBERG, C.; TIMPKA, T. **Inside multidisciplinary design in medical informatics: Experiences from the use of an argumentative design method**. Proceedings of the MEDINFO'95 Tri-annual World Conference in Medical Informatics, Vancouver, 1995.

SKOK, W.; LEGGE, M. **Evaluating enterprise resource planning (ERP) systems using an interpretive approach**, Proceedings of the 2001 ACM SIGCPR conference on Computer personnel research, p.189-197, Apr. 2001, San Diego, California, USA.

SOUZA, C. A.; ZWICKER, R. **Ciclo de vida de sistemas ERP**. Caderno de pesquisas em administração, São Paulo. v. 1, n. 11, 1º trim., 2000a.

SOUZA, C. A.; ZWICKER, R. **Sistemas Integrados de Gestão Empresarial: Estudos de Casos de Implementação de Sistemas ERP**. Dissertação de Mestrado – FEA/USP, 2000b, 253p. – São Paulo.

SOUZA, C. R. B. et al. **A Model Tool for Semi-automatic Recording of Design Rationale in Software Diagrams**. In Proceedings of the String Processing and Information Retrieval Symposium, 1998, pages 306–313.

STUMPF, S. C. **Argumentation-based Design Rationale -** the Sharpest Tools in the Box. Research Note RN/98/103, Computer Science Department, University College London, University of London, U.K., 1997. Disponível em: http://www.cs.ucl.ac.uk/staff/S.Stumpf/Reports/IN9801.html Acesso em: 10/02/2006.

TANG, M. et al. **A Survey of the Use and Documentation of Architecture Design Rationale**, 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), page 89-98.

VOGT, C. **Intractable ERP:** A Comprehensive Analysis of failed Enterprise-Resource-Planning Projects. Software Engineering Notes, Vol. 27, No. 2, Mar. 2002.

ZHANG, L. et al. **Critical success factors of enterprise resource planning systems implementation success in China**. System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference,10 pp.-, 6-9 Jan. 2003.

ZELKOWITZ, M. V.; WALLACE, D. R. **Experimental Models for Validating Technology.** Computer , vol.31, no.5 pp.23-31, May 1998.

REFERÊNCIAS CONSULTADAS

ALI BABAR, M.; GORTON, I.; BARBARA, K. **A Framework for Supporting Architecture Knowledge and Rationale Management**, Rationale Management in Software Engineering, Dutoit, A., McCall, R., Mistrik, I., and Paech, B., Editors., Springer, 2006.

BARD, J. F.; GRAHAM, A. **Developing a data-collection system for PCB assembly: a case study in software engineering**, Engineering Management, IEEE Transactions, vol.40, no.2pp.191-202, May 1993.

BASILI, V. R.; SHULL, F.; LANUBILE, F. **Building knowledge through families of experiments**, Software Engineering, IEEE Transactions, vol.25, no.4pp.456-473, July/Aug. 1999.

BECK, K. et al**. Industrial experience with design patterns**, Software Engineering, 1996., Proceedings of the 18th International Conference on, no.pp.103-114, 25-29 Mar. 1996.

BROADFOOT, G. H.; BROADFOOT, P.J. **Academia and industry meet:** some experiences of formal methods in practice, Software Engineering Conference, 2003. Tenth Asia-Pacific, no.pp. 49- 58, 10-12 Dec. 2003.

BUCHANAN, R. **Myth and maturity: Toward a new order in the decade of design**. In V. Margolin and R. Buchanan (Eds) The Idea of Design. Cambridge, MA: MIT Press, pp. 75-85.

BUDGEN, D.; BRERETON, P**. Encapsulating empirical practices within the software engineering curriculum**, Software Engineering Education & Training. Tenth Conference, no.pp.111-119, 1-16 Apr. 1997.

CARBONERAS, M. C.; INSA, C. M.; SALORT, E. V., **ERP implementation in the stone industry: special difficulties and solutions in the production area**, Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, vol.2, pp. 146-149, Sept. 2003.

CARD, D. N. **Statistical techniques for software engineering practice**, Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference, no.pp. 722- 723, 23-28 May 2004.

CATER-STEEL, A. P. **Low-rigour, rapid software process assessments for small software development firms**, Software Engineering Conference, 2004. Proceedings. 2004 Australian, no.pp. 368- 377, 2004.

CHEN, A.; MCGINNIS, B; ULLMAN, D. G. **Design History Knowledge Representation and its basic Computer Implementation**, Proc. 2nd ASME Int. Conf. on Design Theory and Methodology, J. Rinderle (Ed.), Chicago, IL, ASME, DE-Vol. 27, 1990, pp. 175-184.

CIOLKOWSKI, M. et al. **Software Reviews: The State of the Practice**. IEEE Software. Nov./Dec. 2003, 46-51.

CONKLIN, J. **The IBIS Manual: A Short Course in IBIS Methodology**. Disponível em: <http://www.touchstone.com/tr/wp/IBIS.html>. Acesso em: mar. 2006.

CONKLIN, J. **Hypertext: An introduction and survey**. IEEE Computer, 20, 9, 17-41.

CONKLIN, J. **Designing organizational memory:** Preserving intellectual assets in a knowledge economy. Corporate Memory Systems, Inc. [http://www.cmsi.com/info/pubs/desom/].

CONKLIN, J. **Design rationale and maintainability**. Proceedings 22nd Hawaii International Conference on System Science, 555-561. IEEE Computer Society Press.

CONKLIN, J.; BEGEMAN, M. L. **gIBIS: A hypertext tool for exploratory policy discussion**. ACM Transactions on Office Information Systems, 6, 303-331.

CONKLIN, J. **Interissue dependencies in gIBIS**. Technical Report STP-091-89, Microelectronics and Computer Technology Corporation.

CONKLIN, J.; BEGEMAN, M. L. **gIBIS: A tool for all reasons, Journal of the American Society for Information Science**, (May 1989), 200-213.

CONKLIN, J.; YAKEMOVIC, K. C. B. **A process-oriented approach to Design Rationale**. Human-Computer Interaction, 6 (3&4), 357-391, 1991.

CROSS, S. E. **A quality doctrine for software: do it right the first time**, Software Engineering Conference, 2002. Ninth Asia-Pacific, no.pp. 187- 194, 2002.

CURTIS, B. **Measurement and experimentation in software engineering**, Proceedings of the IEEE , vol.68, no.9pp. 1144- 1157, Sept. 1980.

DANEVA, M.; WIERINGA, R. **Requirements engineering for cross-organizational ERP implementation:** undocumented assumptions and potential mismatches, Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference, no.pp. 63- 72, 29 Aug .- 2 Sept. 2005.

DULEY, R.; MAJ, S. P.; VEAL, D. **Teamwork and Trust: Universities, Industry and the Professional Software Engineer**. Software Engineering Education and Training, 2001. Proceedings. 14th Conference, 2001 - Pages:153-161.

GALLAGHER, K.; MASON, R. M.; VANDENBOSCH, B. **Managing the tension in IS projects:** balancing alignment, engagement, perspective and imagination, System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference, no.pp. 10 pp.-, 5-8 Jan. 2004.

GRAAF, B. et al. **Embedded Software Engineering:** The State of the Practice. IEEE Software. Nov./Dec. 2003, 61-69.

HALLIKAINEN, P.; KIMPIMÄKI , H.; KIVIJÄRVI, H. **Supporting the Module Sequencing Decision in the ERP Implementation Process**, System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference, vol.8, no.pp. 181a- 181a, 04-07 Jan. 2006.

HEFNER, R.; TAUSER, J. **Things they never taught you in CMM school, Software Engineering Workshop**, 2001. Proceedings. 26th Annual NASA Goddard, no.pp.91-94, 2001.

HUA-YANG LIN et al. **An analysis of ERP systems based on n-tier architecture**, Information Technology Interfaces, 2004. 26th International Conference, no.pp. 181- 186 Vol.1, 7-10 June 2004.

JANSEN, S. et al. **Integrated development and maintenance of software products to support efficient updating of customer configurations:** a case study in mass market ERP software, Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference, no.pp. 253- 262, 26-29 Sept. 2005.

JEDLITSCHKA, A.; PFAHL, D. **Experience-based model-driven improvement management with combined data sources from industry and academia**, Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium, no.pp. 154- 161, 30 Sept.-1 Oct. 2003.

JURETA, I. J.; FAULKNER, S.; SCHOBBENS, P. **Justifying Goal Models**, Requirements Engineering Conference, 2006. RE 2006. 14th IEEE International, no.pp. 116- 125, 11-15 Sept. 2006.

KAWAMOTO, S. ; ALMEIDA JR, J. R. **A Influência da Interação Universidade-Empresa no Desenvolvimento de Processos de Engenharia de Software**. In: Congresso Nacional de Tecnologia da Informação e Comunicação - SUCESU2005, 2005, Belo Horizonte. Anais do Congresso Nacional de Tecnologia da Informação e Comunicação, 2005. p. 246-252.

KELLER, T.; SCHNEIDEWIND, N. F. **Successful Application of Software Reliability Engineering for the NASA Space Shuttle**, Software Reliability Engineering - Case Studies, 1997. Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97), p.112, 2-5 Nov. 1997.

MACDONALD, A.; RUSSELL, D.; ATCHISON, B. **Model-driven development within a legacy system**: an industry experience report, Software Engineering Conference, 2005. Proceedings. 2005 Australian, no.pp. 14- 22, 29 Mar. - 1 Apr. 2005.

MACHADO, C. Â. F. et al. **Aderência do RUP à Norma NBR ISO/IEC 12207**. In: II Simpósio Internacional de Melhoria de Processo de Software, 2000, São Paulo. Anais do II Simpósio Internacional de Melhoria de Processo de Software, 2000.

MACKE, S. et al. **An industry/academic partnership that worked:** an in progress report, Software Engineering Education, 1996. Proceedings Ninth Conference, no.pp.234-245, 21-24 Apr. 1996.

MACLEAN, A. et al. **Questions, Options and Criteria: Elements of Design Space Analysis, in 'Design Rationale: Concepts, techniques, and Use'**, Moran and Carroll eds., Lawrence Erlbaum, 1996.

MARKUS, M. L.; TANIS, C.; VAN FENEMA, P. C., **Multisite ERP Implementations**. Communications of the ACM, 2000. 43(4): p. 42-46.

MAZZONI, O. S. **Bridging the gap between education and industry expectations in Africa** - Technology and Society Magazine, IEEE, Vol.9, Iss.3, Sep/Oct. 1990 Pages: 27-29.

MCCONNELL, S. **Avoiding classic mistakes [software engineering]**, Software, IEEE , vol.13, no.5pp.112, 111-, Sep. 1996.

MEHANDJIEV, N. et al. **Thirteen Knights and the Seven-headed Dragon:** an interdisciplinary software engineering framework, Software Technology and Engineering Practice, 2002. STEP 2002. Proceedings. 10th International Workshop, no.pp. 46- 54, 6-8 Oct. 2002.

NIKULA, U. et al. **Establishing the Current Practice in Industry as the Baseline for Educational Infrastructure:** Case South-East Finland, Software Engineering Education and Training, 2005. CSEE&T 2005. Proceedings. 18th Conference, no.pp. 173- 180, 18-20 Apr. 2005.

OHBA, M.; SATO, Y. **Experimenting component-based technology in industry settings**, Assessment of Software Tools and Technologies, 1997, Proceedings Fifth International Symposium, no.pp.98-99, 2-5 June 1997.

OLALEKAN, A. S. **Conducting empirical software engineering research in Nigeria:** the posing problems, Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference, no.pp. 633- 634, 15-21 May 2005.

PARR, A.N.; SHANKS, G., **A taxonomy of ERP implementation approaches**, System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference, 10 pp. vol.1-, 4-7 Jan. 2000.

POORE, J.H. **A tale of three disciplines and a revolution [software engineering]**, Computer , vol.37, no.1pp. 30- 36, Jan. 2004.

POTTS, C. **Software-engineering research revisited**, Software, IEEE , vol.10, no.5pp.19-28, Sept. 1993.

RITTEL, H.; WEBBER, M. **Planning Problems are Wicked Problems**, in Developments in Design Methodology, Nigel Cross ed., John Wiley and Sons, 1984.

ROJAS, T.; PEREZ, M. **The capabilities and maturity model (CMM): a case study**, Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, 1997 IEEE International Conference, no.pp. 1285-1290 vol.2, 12-15 Oct. 1997.

RONKKO, K.; LINDEBERG, O.; DITTRICH, Y. **'Bad practice' or 'Bad methods' are software engineering and ethnographic discourses incompatible?** Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium, no.pp. 204- 210, 2002.

SHAW, M. **Prospects for an engineering discipline of software**, Software, IEEE , vol.7, no.6pp.15-24, Nov. 1990.

SHIH-HAO, L.; BAW-JHIUNE, L. **IBO: an issue based object model for software design**, TENCON '93. Proceedings. Computer, Communication, Control and Power Engineering.1993 IEEE Region 10 Conference, no.0pp.270-274 vol.1, 19-21 Oct. 1993.

SHULL, F.; TURNER, R. **An empirical approach to best practice identification and selection**: the US Department of Defense acquisition

best practices clearinghouse, Empirical Software Engineering, 2005 - International Symposium, no.pp. 8 pp.-, 17-18 Nov. 2005.

SJOBERG, D. I. K. et al. **Conducting realistic experiments in software engineering, Empirical Software Engineering**. Proceedings. 2002 International Symposium, no.pp. 17-26, 2002.

SZITAS, Z., **Technical requirements in enterprise resource planning systems**, Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004. 27th International Spring Seminar, vol.3, no.pp. 461- 466 vol.3, 13-16 May 2004.

THEMISTOCLEOUS, M. et al. **ERP problems and application integration issues:** an empirical survey, System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference, 10 pp.-, 3-6 Jan. 2001.

TOWHIDNEJAD, M.; SALIMI, A. **Incorporating a disciplined software development process in to introductory computer science programming courses:** initial results, Frontiers in Education Conference, 1996. FIE '96. 26th Annual Conference, vol.2, no.pp.497-500 vol.2, 6-9 Nov. 1996.

TULLY, C. **A failure of management nerve and vision [software engineering]**, Software Engineering, 1991. Proceedings of the 13th International Conference, no.pp.154-155, 13-16 May 1991.

WESTERHEIM, H.; HANSSEN, G. K. **The introduction and use of a tailored unified process - a case study**, Software Engineering and Advanced Applications, 31st EUROMICRO Conference, no.pp. 196- 203, 30 Aug. - 3 Sept. 2005.

WIEGERAAD, S. **Development of a design history information system**. 1999. Acesso em: 02/07/2006. DisponÍvel em: <http://alexandria.tue.nl/extra3/proefschrift/boeken/9902517.pdf>.

YE BIN; MA ZHONG-GUI; TU XU-YAN, **Research on the architecture of ERP system based on intelligent autonomous decentralized system**, Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings, no.pp. 616- 619, 4-8 Apr. 2005.

APÊNDICE A – Pesquisa enviada por e-mail

Pesquisa – Projeto de sistemas ERP

O questionamento que estou enviando está relacionado ao meu trabalho de mestrado na Poli-USP.

Busco exemplo de **decisões** tomadas durante a fase de análise de projeto de software. Para cada decisão gostaria de saber quais foram as alternativas consideradas e pontos positivos **(+)** e negativos **(-)** de cada uma.

De preferência:

- está relacionado a ERP e não a TI em geral;
- estas decisões foram tomadas após discussões em um grupo (pois provavelmente haverá mais alternativas consideradas);
- mais de uma vez voltou-se à discussão da solução;
- depois de certo tempo, houve questionamento da decisão tomada e até tentou-se outra alternativa (as razões da decisão raramente são documentadas);

Podem ser coisas simples, como os exemplos fictícios abaixo:

1- Estilo da Interface[1]
   - Clássica: aceitação pela maioria das pessoas**(+)**, impacto**(-)**, facilidade de implementação**(+)**
   - Moderna: aceitação pela maioria das pessoas**(-)**, impacto**(+)**, facilidade de implementação**(-)**

   O estilo escolhido foi a Moderna pois foi a aceita pelo Presidente da empresa.

2- Tipo de contabilização (financeiro)
   - On-line: dados disponíveis imediatamente**(+)**, complexidade de implementação**(-)**, uso de integridade referencial**(-)**, flexibilidade do sistema**(-)**
   - Off-line: dados disponíveis imediatamente**(-)**, complexidade de implementação**(+)**, uso de integridade referencial**(+)**, flexibilidade do sistema**(-)**
   - On-line e off-line: dados disponíveis imediatamente**(+)**, complexidade de implementação**(-)**, uso de integridade referencial**(+)**, flexibilidade do sistema**(+)**

   O tipo escolhido foi o on-line e off-line, pois a flexibilidade do sistema foi a característica mais importante no momento.

Se achar que é mais prático conversar pessoalmente, por favor, me avise.
Obrigada,

Sandra Kawamoto

[1] Exemplo não muito bom pois não está ligado especificamente a ERP.