CÁSSIO ROGERIO CELESTINO DOS SANTOS

GERÊNCIA DE RISCO NA MODERNIZAÇÃO DE SISTEMAS LEGADOS

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Engenharia de Software

São Paulo

2004

CÁSSIO ROGERIO CELESTINO DOS SANTOS

GERÊNCIA DE RISCO NA MODERNIZAÇÃO DE SISTEMAS LEGADOS

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Engenharia de Software

Área de Concentração:
Engenharia de Software

Orientadora:
Profa. Dra. Lúcia Vilela Leite Filgueiras

São Paulo
2004

Aos meus pais, Nourival e Luzia pelo amor e carinho que tem me proporcionado,
e o exemplo de dedicação e esforço.

A minha querida esposa Ariadney pelo amor correspondido,
e por fazer-me mais feliz.

Ao meu amado filhinho Ulisses pelo seu carinho e seu sorriso
que são minha fonte de motivação.

A Deus por me proporcionar principalmente saúde.

Agradecimentos

A minha orientadora Dra. Lúcia Vilela Leite Filgueiras pelas preciosas recomendações e diretrizes e incentivo na elaboração deste trabalho.

A minha sogra Marlene Riami Almicci que de várias formas tem colaborado para que fosse possível realizar este trabalho.

Aos meus irmãos Abner e Aline pelo incentivo e apoio nesse trabalho.

Ao demais familiares que cada um com pequenos gestos também têm participado na elaboração deste trabalho.

Aos meus colegas de trabalho, Celso Bugni e Cristina Nakata que também de forma significativa têm me auxiliados para que fosse possível a realização desta monografia.

RESUMO

GERÊNCIA DE RISCOS NA MODERNIZAÇÃO DE SISTEMAS LEGADOS

Os sistemas corporativos conhecidos como legados são sistemas grandes e complexos, e são freqüentemente considerados estratégicos para as organizações, porém são sistemas antigos que resistem às mudanças tecnológicas. As empresas que os utilizam já perceberam que modernizá-los é uma questão de sobrevivência perante as novas regras tecnológicas e mercadológicas. Entretanto o processo de modernização dos sistemas legados é muito temido devido às incertezas relacionadas à funcionalidade do legado e à adequação das novas tecnologias. Esse trabalho visa aplicar as técnicas de gerência de riscos de forma a auxiliar a tomada de decisão do gerente quanto à escolha de uma técnica de modernização. O objetivo é identificar os possíveis riscos associados às tecnologias de modernização dos legados e avaliá-los, contribuindo para tornar o processo de modernização previsível e com maior probabilidades de sucesso.

ABSTRACT

Risk Management in the Modernization of Legacy System

Corporate systems known as Legacy Systems are large and complex computer systems and are frequently considered of strategical importance for corporations. However, they are often obsolete and resistant to technological evolution. Companies that use these systems have already realized that it is essential to modernize them in order to meet technological and marketing demands. Nevertheless, legacy systems modernization process is feared due to uncertainties concerning the development of the new system and its functionality. This work applies risk management techniques to help managers decision making towards selecting a modernization technique identify and assess the uncertainties mentioned above. The objective of this work is to determine the potential risks associated to legacy systems modernization technique and to evaluate them, thus contributing to make the modernization process more predictable and successful.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

INTRODUÇÃO

Este trabalho trata da gerência de riscos na modernização de sistemas legados. Sistemas legados são sistemas de grande porte desenvolvidos em ambiente mainframe.

As empresas investem muito dinheiro em sistemas de software, e para que elas obtenham um retorno desse investimento o software deve ser utilizado por vários anos. O tempo de duração dos sistemas de software é muito variável, os sistemas legados permanecem em uso por mais de 10 anos. Muitos desses antigos sistemas ainda são fundamentais para as empresas, pelo fato de que muitos processos de negócios das organizações foram encapsulados em lógicas de aplicação mainframe, sendo eles o repositório de fato da experiência e do conhecimento das organizações.

Fatores internos e externos às empresas, como o estado da economia nacional e internacional, as modificações nos mercados, as alterações nas leis, as mudanças de gestão e organização estrutural das empresas, fazem com que elas sofram mudanças contínuas, significando que seus sistemas devem ser **continuadamente** atualizados para refletir as práticas de negócios e acompanhar a evolução da tecnologia.

Com a evolução da computação, os usuários passaram a exigir mais dos sistemas de informação, inclusive dos sistemas legados: interfaces gráficas, tempo de resposta pequeno, extração on-line de informações gerenciais são alguns exemplos. Por conta dessas exigências, os sistemas legados devem ser modernizados, porém estes sistemas tornam-se demasiadamente frágeis para modificar e demasiadamente importantes para serem ignorados.

1.1 Motivação do trabalho

De maneira geral, observa-se que o processo de modernizações em sistemas legados é bastante temido, devido às incertezas próprias ao meio do desenvolvimento do novo sistema e suas funcionalidades.

Este trabalho visa aplicar as técnicas de gerência de riscos ao processo de modernização do sistema legado. Essas incertezas são identificadas e mapeadas na forma de riscos. A aplicação das técnicas de gerência de riscos torna o processo de modernização previsível e gerenciado.

1.2 Objetivos

O objetivo principal deste trabalho é estabelecer um roteiro para consideração dos riscos na seleção de estratégia de modernização de sistemas legados. Para tanto os objetivos secundários são:

- Compreender as dificuldades da modernização do sistema legado;
- Compreender a gerência de riscos;
- Identificar os tipos de problemas relacionados à gerência de riscos na modernização do sistema legado;
- Levantar, para cada tipo, os principais fatores que dificultam o processo de modernização do sistema legado;
- Apresentar um roteiro para identificação e quantificação dos riscos que seja aplicável e de fácil utilização.

1.3 Delimitação do estudo

Esta monografia está baseada no estudo das técnicas de gerência de riscos aplicadas à modernização dos sistemas legados. Diversas áreas usam o tema da gerência de riscos, porém este trabalho está relacionado com a abordagem usada na Engenharia de Software.

Nesta área de conhecimento, da Engenharia de Software, este trabalho está direcionado para a área de manutenção de sistemas, e ainda as de análise e desenvolvimento.

1.4 Perspectiva de Contribuição

Espera-se que com base neste trabalho seja possível identificar os principais passos para a gerência de riscos dos sistemas relacionados à modernização de legados. A contribuição desse trabalho é fornecer uma ferramenta de consulta para suporte a decisões na escolha da tecnologia a fim de modernizar os sistemas legados.

1.5 Metodologia

O desenvolvimento desse trabalho tem como base às técnicas de modernização do sistema legado e a interação com a área de gerência de riscos da Gerência de Projetos aplicada à Engenharia de Software.

Foram identificadas as principais dificuldades observadas na modernização dos sistemas legados e, em seguida, através do estudo da Gerência de Riscos, foram levantadas as principais atividades e suas aplicações na modernização de legados.

Os problemas observados foram identificados e classificados segundo sua categoria e seqüencialmente desenvolveu-se um roteiro para consideração dos riscos na seleção de estratégia de modernização de sistemas legados.

1.6 Organização do Trabalho

O trabalho é dividido em 5 capítulos e apresenta a seguinte estrutura:

- Capítulo 1 - Introdução: Consiste neste capítulo, onde são apresentados o escopo, os objetivos e a metodologia utilizada no trabalho.
- Capítulo 2 - Modernização de Sistemas Legados: é composto por definições do sistema legado e apresentação das técnicas de modernização consideradas neste estudo.
- Capítulo 3 - Gerência de Riscos: é composto por breve apresentação conceitual do tema.
- Capítulo 4 - Proposta de um Roteiro de Gerência de Riscos: é o principal capítulo, onde são apresentados os principais riscos aos quais as empresas estão expostas, justificando a gerência dos riscos envolvidos neste

processo e onde se propõe um roteiro de identificação e quantificação dos riscos.

- Capítulo 5 - Conclusão: Apresenta uma visão geral da pesquisa, um resumo das contribuições deste trabalho e as propostas de trabalhos futuros.

CAPÍTULO 2

SISTEMAS LEGADOS E TÉCNICAS DE MODERNIZAÇÃO

Neste capítulo, são apresentadas as definições dos sistemas legados. Essas definições esclarecem o significado dos sistemas legados para as organizações, os porquês desses sistemas sobreviverem até hoje, e ainda quais as necessidades de modernizá-los. Em seguida são apresentadas algumas técnicas de modernização, que serão a base para o roteiro proposto neste trabalho.

Muitas das publicações que abordam o tema "Modernização de Sistemas Legados" referem-se à reengenharia e/ou engenharia reversa com enfoque em orientação a objetos[Rechia et al, 2002] [Cagnin, 1999] [Fontanette et al, sd] [Bergey et al, 2001] entre outros. Porém este trabalho aborda outras alternativas de modernização, sendo baseado no relatório técnico do SEI *(Software Engineering Institute)* publicado em 2000, com o Título *"A Survey of Legacy System Modernization Approaches"* [Comella-Dorda et al, 2000]. Neste relatório o autor faz uma breve apresentação de algumas técnicas de modernização de legados, as quais são identificadas e avaliadas nesta monografia sobre o ponto de vista dos riscos na modernização.

2.1 Definição de Sistemas Legados

Segundo o Dicionário On-line de Computação (*Free On-line Dictionary of Computing*), sistema legado é um sistema ou um programa de aplicação que continua em uso por causa do custo proibitivo de substituir ou de modernizar. A implicação é que o sistema é grande, monolítico e de difícil entendimento [FOLDOC, 1996].

Segundo Bennett [Bennett, 1995], sistemas legados são sistemas grandes e complexos que geralmente são difíceis de se lidar, mas são vitais para a organização.

Segundo Brodie [Brodie, 1995], sistemas legados são sistemas de informação que resistem significativamente às modificações e às evoluções tecnológicas e mercadológicas.

Segundo Pinheiro[Pinheiro, 1996], sistemas legados são sistemas de informação desatualizados, que resistem às mudanças tecnológicas, e executam funções críticas dentro das organizações.

De acordo com Comella-Dorda [Comella-Dorda et al, 2000] esses sistemas são grandes e complexos, pois as organizações que suportam são grandes e possuem lógicas de negócios complexas, que sempre estão encapsuladas nos sistemas. Esses sistemas são sistemas heterogêneos, pois geralmente constituem um ambiente misto de tecnologias, fruto das constantes evoluções feitas no decorrer dos anos.

De uma forma geral, sistemas legados são sistemas críticos que desempenham funções vitais para a organização, são sistemas antigos que incorporam um grande número de alterações e que refletem a evolução do negócio durante esses anos.

2.2 Sobrevivência dos Sistemas Legados

Um dos principais fatores da sobrevivência dos sistemas legados é a importante função que esses sistemas desempenham, pois representam a experiência e o conhecimento das organizações, contam com informações valorosas e as regras fundamentais dos negócios estão encapsuladas nele. Segundo Comella-Dorda [Comella-Dorda et al, 2000] esses sistemas são comparados ao cérebro da organização.

Manter os sistemas legados em uso evita os riscos de substituição. Segundo Seacord [Seacord, 2001], o processo de modernização desses sistemas é um desafio devido ao tamanho e a complexidade dos sistemas envolvidos.

2.3 Motivos da Modernização de Sistemas Legados

Com o passar do tempo, os antigos sistemas vão se desatualizando e se tornando de difícil operação e integração, devido às mudanças na tecnologia e nas operações de negócios, que são refletidas em sistemas mais novos. Segundo Comella-Dorda [Comella-Dorda et al, 2000], as organizações devem estar atentas em controlar as evoluções dos sistemas, enquanto as práticas de negócio mudam, as novas tecnologias de informação podem fornecer vantagens em relação ao competidor.

Um bom exemplo é o setor financeiro; um banco que hoje não oferece serviços de *internet banking*, por exemplo, está bem atrás na concorrência, enquanto que no passado, o banco que oferecia esses mesmos serviços, tinha vantagens em relação ao seu competidor.

Portanto modernizar os sistemas legados é uma questão de sobrevivência da empresa perante as novas regras e exigências do mercado. As organizações devem ter sistemas "flexíveis" que se adaptem rapidamente às mudanças.

Além de outras vantagens, a modernização desses sistemas torna o processo de manutenção menos dispendioso, pois os principais problemas na manutenção dos sistemas legados são:

- Parte do sistema ou todo o sistema foi implementado com linguagem de programação obsoleta, sendo a mão de obra escassa e cara.
- De acordo com Recchia [Recchia et al, 2002] a documentação do sistema é inadequada e desatualizada. Em alguns casos a única documentação é o código fonte do sistema.
- Em geral, muitos anos de manutenção podem ter corrompido a estrutura do sistema, tornando-o cada vez mais difícil de ser compreendido.

De acordo com Belloquim [Belloquim, 1999] dar manutenção em sistemas mal feitos e sem documentação é como "patinar numa poça de lama", gasta-se muito esforço e não chega a lugar nenhum.

Com a modernização dos sistemas legados a fase de manutenção desses sistemas será facilitada. Em geral, não será difícil encontrar profissionais que conheçam essas tecnologias novas, e por se tratar de sistemas mais novos, presume-se que as práticas

de engenharia de software foram aplicadas, refletindo em uma boa documentação de sistemas.

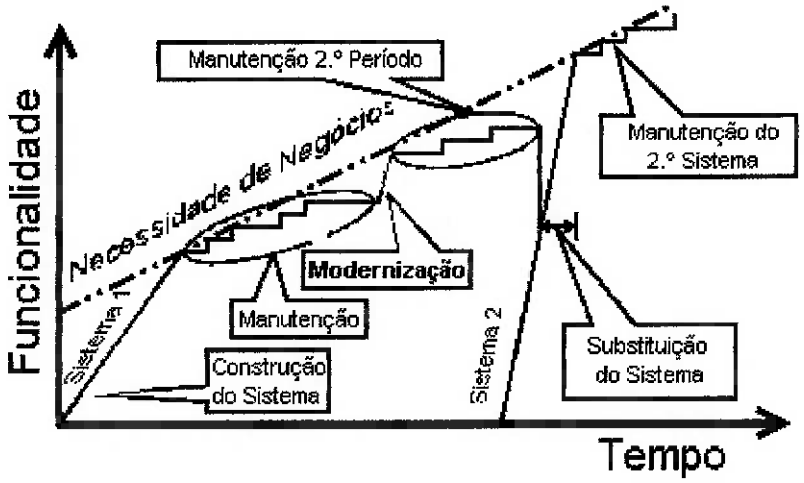
2.4 Alternativas para Evolução de Sistema Legado

De acordo com Weiderman [Weiderman, 1997a], o processo de evolução dos sistemas pode ser dividido em três categorias: Manutenção, Modernização, e Substituição.

- **Manutenção** – é um processo incremental e interativo onde são realizadas pequenas mudanças tais como correções de erros e/ou inclusões de algumas novas funções no sistema. A manutenção ajuda suportar a evolução de alguns sistemas, mas tem limitações como, por exemplo, o não envolvimento de grandes mudanças estruturais.
- **Modernização** - envolve modificações maiores que na manutenção, podendo ocorrer alterações na estrutura do sistema, importantes funções estruturais podem ser adicionadas, porém, os valores e/ou regras de negócio são preservadas.
- **Substituição** - esse processo normalmente é usado quando o sistema é muito antigo, onde os custos da manutenção e modernização são inviáveis; geralmente são sistemas sem documentação e completamente desatualizados.

A figura 2.1 ilustra a funcionalidade dos sistemas e as necessidades dos negócios no decorrer do tempo.

***Figura 2.1** – Necessidades dos negócios x Sistemas x Tempo*



Fonte: Adaptado de [Comella-Dorda et al, 2000]

A linha pontilhada indica as necessidades dos negócios, e a linha cheia indica a evolução dos sistemas. Note-se que a manutenção nos sistemas atende às necessidades dos negócios por algum tempo, porém com as constantes mudanças nos negócios surgem novas funcionalidades e mudanças estruturais maiores são necessárias. Para suportar essas alterações é necessário modernizar o sistema.

O fluxo de manutenção seguido de modernização se repete no decorrer do tempo, e finalmente quando o sistema fica muito desatualizado, a documentação fica incompleta e custos de manutenção cada vez mais altos, surge então a necessidade de substituir o sistema.

Este trabalho está focado na **modernização** dos sistemas legados. Como mencionado no começo desse capítulo, a monografia foi desenvolvida focando as técnicas de modernização apresentadas em [Comella-Dorda et al, 2000]. Essas técnicas são abordadas a seguir.

2.5 Modernização

O processo de modernização é aplicado quando os sistemas legados requerem mudanças significativas as quais o processo de manutenção não suporta, porém preservando as regras de negócio. Segundo Weiderman [Weiderman, 1997b], o processo de modernização pode ser dividido em dois níveis de detalhamento: Modernização de Caixa-Branca e Modernização de Caixa-Preta.

- ***Modernização de Caixa-Branca*** - Requer conhecimento detalhado da estrutura interna do sistema como, por exemplo, entender os códigos de programas e seus módulos de relacionamentos internos e externos.
- ***Modernização de Caixa-Preta*** – Apenas se preocupa com as interfaces externas dos sistemas.

A escolha da técnica de modernização, e o que se deseja modernizar são fatores importantes para decidir se o processo precisa ser mais detalhado ou não. Segundo Cormella-Dorda [Cormella-Dorda et al, 2000], a modernização dos sistemas legados podem ocorrer nas seguintes categorias:

- Relação com Usuários – (Interfaces)
- Dados
- Relação funcional (Lógica)

2.6 Modernização das Interfaces do Usuário

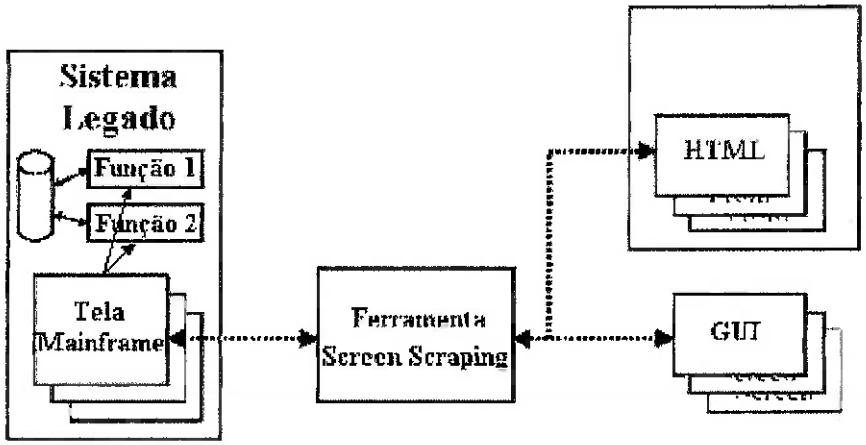
A Interface do Usuário *(IU)* é a apresentação de um sistema. Geralmente o objetivo da modernização de interfaces é melhorar a usabilidade, facilitando assim a utilização do sistema pelos usuários finais. Segundo [Carr 98 apud Comella-Dorda et al, 2000], a técnica comum para modernização de IU é *a Screen Scraping*. Essa técnica de modernização de interfaces não toma conhecimento da estrutura interna do sistema. Portanto, trata-se de modernização de caixa-preta. No tópico de modernização funcional (lógica) será tratada a modernização de interfaces de forma mais abrangente.

2.6.1 Screen Scraping

A técnica *Screen Scraping*, também conhecida como navegação baseada em telas ou captura de dados baseados em telas, lê o fluxo de dados destinado a um terminal de mainframe e os transforma em uma apresentação gráfica moderna [Attachmate, 2000]

O fluxo de dados destinado a um terminal mainframe é um conjunto de telas de texto que são processadas pelo mainframe, portanto a proposta da técnica *screen scraping* é a transformação desta tela de texto para uma tela padrão mais amigável denominada GUI - Interface Gráfica do Usuário (*Graphical User Interfaces*), ou mesmo para uma página HTML (*Hypertext markup language*). Essa nova interface gráfica comunica-se com a interface do sistema legado através de uma ferramenta de *scraping*[1] especializada em gerar uma nova tela a partir do mapeamento da tela velha, como mostra a figura 2.1.

Figura 2.1- Modernização de Interfaces usando Screen Scraping



Fonte: Adaptado de [Comella-Dorda et al, 2000]

Sendo assim, a *screen scraping* evita alterações no aplicativo mainframe e o torna mais fácil de ser usado. Também pode permitir modificações nas informações apresentadas ao usuário combinando várias telas em uma única apresentação gráfica,

---

[1] Exemplo, OC://WebConnect Enterprise Integration Server™ do Open-Connect Systems, ou QuickApp™ do Attachmate.

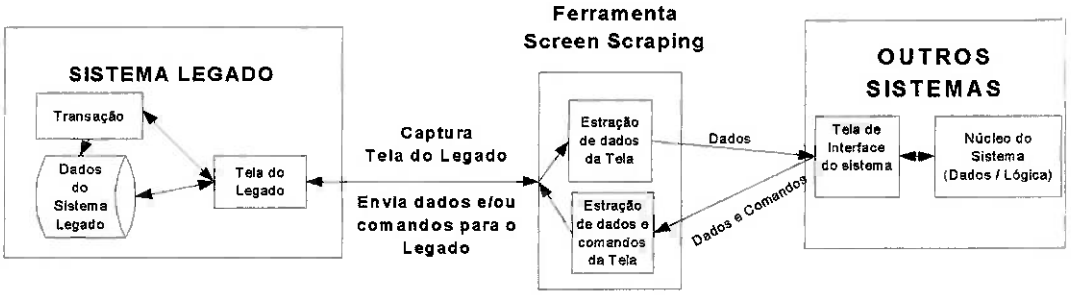
sem modificar a lógica da aplicação. Essas modificações de telas dão a impressão que a aplicação mainframe foi modificada.

As principais formas de utilização da técnica *screen scraping* são:

- Fornecer ao aplicativo de mainframe uma nova aparência, ou seja, substituir as telas de emulação do mainframe por interfaces gráficas do usuário [Attachmate, 2000].
- Fornecer uma outra funcionalidade, considerando o fato de que a lógica de um aplicativo está exposta pela seqüência de telas chamadas pela ação do usuário. O *screen scraping*, neste caso, captura e encapsula esta lógica seqüencial, usando os conceitos de componentização, afim de que essas telas fiquem mais bem agrupadas para serem mais utilizáveis [Attachmate, 2000].
- Outra aplicação interessante segundo Comella-Dorda [Comella-Dorda et al, 2000], é a utilização da técnica *screen scraping* na integração entre sistemas. As telas geradas pela ferramenta *screen scraping* são usadas como sendo uma nova interface de acesso ao legado (API) para comunicação e integração entre os sistemas.

De acordo com Comella-Dorda, essa última forma de utilização da técnica *screen scraping* foi utilizada em um projeto de integração no Instituto de Defesa dos Estados Unidos, integrando um Sistema de Gestão da Empresa (ERP) com outros sistemas. Neste projeto de integração, o ERP utiliza as telas *screen scraping* como entrada e saídas de dados para integração entre sistemas, como mostra a figura abaixo.

Figura 2.2- Utilização da Ferramenta Screen Scraping na Integração de Sistemas



Portanto a maior vantagem da *screen scraping* é a capacidade de codificar as interfaces independentemente do aplicativo mainframe, além da possibilidade de gerar APIs, provendo fácil integração entre sistemas.

2.7 Modernização de Dados

A modernização dos dados permite que dados do sistema legado sejam acessados por diferentes interfaces ou protocolos que foram projetados inicialmente. Segundo Comella-Dorda, o grande estímulo à modernização de dados é aumentar à conectividade e permitir a integração dos dados do legado em modernas estruturas. Esse aumento da conectividade e facilidade de integração de dados com outros sistemas é possível com o surgimento de algumas técnicas de integração, desde *gateways* até replicação de dados. A modernização de dados é considerada modernização de caixa-branca.

2.7.1 Gateways e Bridges

Um *gateway* de banco de dados é um tipo específico de software (*middleware*) que traduz a "conversa" entre dois ou mais protocolos de acesso [Altman, 1999 apud Comella-Dorda et al, 2000]. O propósito dos *gateways* foi criar uma maneira comum de acesso ou protocolos de acessos, independente dos

fornecedores, como por exemplo, Oracle Databases, SQL Server (Microsoft), DB2 (IBM) entre outros.

Aplicações e Plataformas Modernas

O *gateway* de banco de dados normalmente traduz um protocolo específico de acesso a dados em um protocolo padrão de mercado. Essa tradução é útil porque as aplicações e plataformas modernas geralmente suportam um ou mais destes protocolos padrão.

Os sistemas de mainframe podem fazer uso de *gateways* para aumentar sua conectividade suportando a integração com sistemas modernos, e disponibilizar acessos remotos à suas bases de dados.

Existem muitos protocolos, porém apenas alguns são constituídos como padrão de mercado, a saber:

2.7.1.1 ODBC - Open Database Connectivity (Conexão a Banco de Dados)

É uma interface da Microsoft para acesso a dados em sistema de gerenciamento de banco de dados heterogêneos [Umar, 1997]. Essa interface está baseada na especificação do Consórcio da Comunidade de Banco de Dados denominada SAG - *Standard Query Language Access Group*, que definiu um padrão unificado de linguagem para acesso a banco de dados remotos denominado CLI - *Call Level Interface*. A CLI não é uma linguagem de consulta, é simplesmente uma interface procedural para a Linguagem Padrão de Consultas a banco de dados (SQL - *Standard Query Language*), que não adiciona nem subtrai poder ao SQL, sendo apenas um mecanismo de submissão de instrução SQL.

O ODBC fornece um caminho de acesso a bases de dados em computadores pessoais, minicomputadores e em mainframe.

Acrescente-se a esta categoria de técnicas de modernização de dados a tecnologia OLE DB como sendo uma evolução da tecnologia ODBC.

O OLE DB foi especificado tendo por base o ODBC embora o objetivo neste caso foi de desenvolver algo que não estivesse limitado pelo modelo relacional. Assim o OLE DB foi desenhado para permitir o acesso tanto a dados relacionais como não relacionais (Ex: Bancos de Dados Hierárquico, e-mail, arquivo de sistemas, arquivos textos, etc).

De acordo com Bragança [Bragança, 1999], o OLE DB define uma coleção de interface COM (*Component Object Model*) que encapsulam vários serviços de gestão de bases de dados. Estas interfaces permitem a criação de componentes de software que implementam esses tipos de serviços. Os componentes OLE DB consistem em *data providers*, que armazenam e fornecem dados; *data consumers*, que utilizam dados; e *services components*, que processam e transportam dados [Bragança, 1999].

2.7.1.2 JDBC - *Java Database Connectivity* (Conexão de Bancos de dados - Java)

É um padrão de mercado definido pela Sun Microsystems, trata-se de uma API única para acesso à base de dados. O objetivo é prover uma interface orientada a objeto, para acessar uma base de dados relacional.

JDBC é uma API no nível de SQL, ou seja, pode construir instrução SQL e embuti-las dentro de uma API Java. Essas APIs Java de acesso a banco de dados baseou-se sobre os aspectos da API de ODBC, sendo assim a JDBC é um conjunto de classes e interfaces em Java, que proporciona uma interface similar a ODBC para bases de dados SQL. [JDBC] [Umar, 1997]

Outra característica importante são os metadados, que permite descobrir informações sobre os dados retornados, por exemplo qual o tipo de dado e o nome de cada coluna da tabela resultante.

2.7.1.3 ODMG – Object Data Management Group (Grupo de Gerência de Banco de Dados)

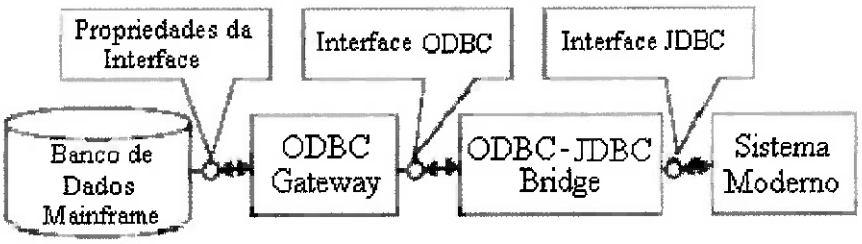
Trata-se de um padrão de armazenamento persistente de objetos, em estrutura totalmente configurada em cima dos padrões e paradigmas de objeto definido pelo ODMG.

Essas estruturas ou aplicações geralmente são complexas, pois requerem um sistema de banco de dados que manipule informações como: tipo de dados complexos; encapsulamento e abstração da estrutura de dados e também prover novos métodos para indexação, manipulação e consulta desses dados [OMG] [Barry, 1998 apud Comella-Dorda et al, 2000].

Exemplo

A figura 2.3 mostra um exemplo de uma aplicação mainframe que suporta *gateway* ODBC e pretende-se integrar com um sistema de baixa plataforma, desenvolvido em linguagem orientada a objetos, que suporta interface JDBC. É possível esta integração? A resposta é sim, a solução é: utilizar um *gateway* especial chamado ponte (*bridge*) que vai por sua vez traduzir um protocolo padrão em outro.

Figura 2.3 – Gateway e Bridge



Fonte: Adaptado de[Comella-Dorda et al, 2000]

2.7.2 Integração XML – Extensible Markup Language

A internet é o maior repositório de informação em todo o mundo. A maior parte dessas informações está no formato HTML. No entanto, esse formato preocupa-se apenas com a formatação das informações e não com o conteúdo. Assim, surge a necessidade de uma linguagem voltada tanto para a estrutura quanto

para o conteúdo das informações. Desde de 1986 existe um padrão internacional capaz de satisfazer tal necessidade, o SGML (*Standard Generalized Markup Language*) (ISO 8879). Entretanto, a complexidade de implementação e o alto custo fazem com que seu uso fique restrito às grandes empresas [Cerqueira, 2000].

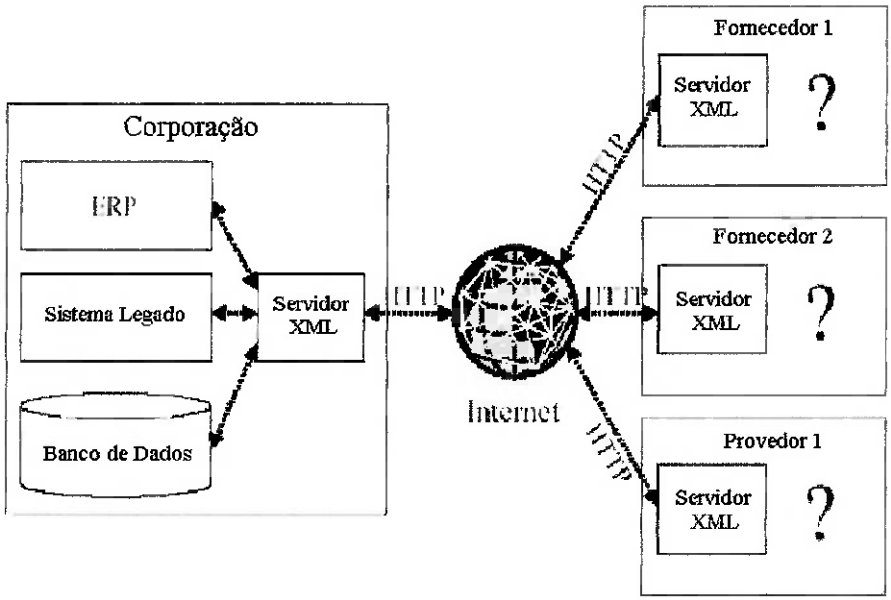
Com o objetivo de tornar a SGML viável para o uso da internet, o W3C (*World Wide Web Consortium*) criou um subconjunto apropriado da SGML, denominado XML (*eXtensible Markup Language*) [XML].

A linguagem XML se destaca em duas grandes áreas: a primeira é a grande importância no intercâmbio de documentos, a segunda pela sua capacidade de comunicação na troca de dados entre sistemas e/ou organizações usando o protocolo padrão HTTP, destacando-se no segmento de comércio eletrônico, especificamente na integração de aplicações e nos sistemas B2B – *Business to Business* (Negócio para Negócio) [Turban et al, 1999].

A flexibilidade na formatação XML possibilita às organizações compartilharem informações com consumidores e parceiros sem a necessidade de negociar detalhes técnicos, pois nesta tecnologia a organização não precisa conhecer os sistemas do cliente e vice-versa, apenas precisam definir qual o padrão XML que será trocado.

Segundo Comella-Dorda [Comella-Dorda et al, 2000], o XML *Server* (Servidor XML) tem a importante função de fornecer contato entre a infraestrutura da empresa e o mundo exterior. O XML Server pode comunicar internamente com as infraestruturas de banco de dados, sistemas legados, ERP entre outros, integrando-os com outros sistemas internos ou externos se for o caso, como apresentado na figura 2.4.

Figura 2.4 - Integração com XML – Extensible Markup Language



Fonte: Adaptado de [Comella-Dorda et al, 2000]

Outro ponto importante é a portabilidade da linguagem XML, isto é, um documento XML pode ser usado para armazenar e transferir a informação completamente independente da plataforma, do sistema operacional e da maneira pela qual será apresentada. Como o documento XML é texto puro, pode ser transmitido facilmente sobre qualquer tipo de rede. Sendo assim as aplicações se comunicam fazendo com que apenas os dados trafeguem [XML].

Um bom exemplo de aplicação para ilustrar a ligação entre o sistema legado e XML Server na figura 2.4, é o Sistema de Pagamento Brasileiro (SPB). O SPB é um sistema moderno que integra as instituições financeiras com o Banco Central, envolvendo praticamente todos os agentes atuantes da nossa economia (pessoas, governos, empresas, instituição financeira, Banco central, etc.) [SPB].

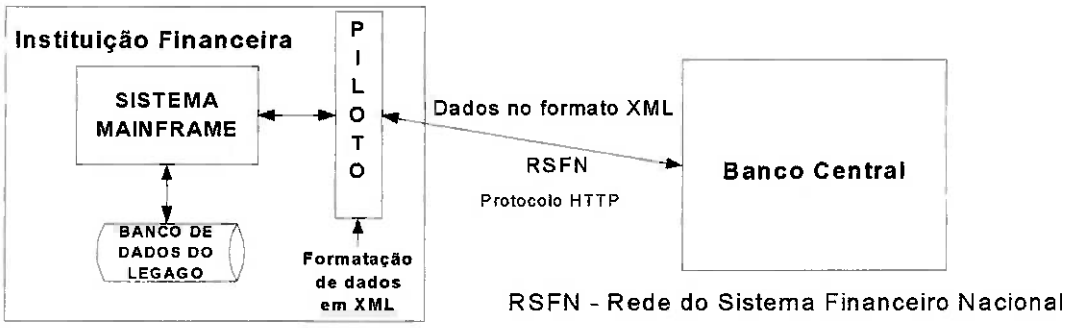
A instituição financeira na qual o autor trabalha passou por um grande esforço para desenvolver e integrar esse moderno sistema de pagamento, uma vez que os sistemas nas instituições financeiras na sua grande maioria são desenvolvidos em ambiente mainframe.

Como solução integradora em aplicações B2B nesta instituição foi utilizada a tecnologia XML, pela flexibilidade na padronização de trocas de dados e a não

necessidade de conhecer detalhes técnicos de sistemas alheios envolvidos no processo. É necessário definir o padrão XML de mensagens a serem trocadas.

Resumidamente a solução adotada nesta instituição financeira em questão foi, o desenvolvimento de um "piloto" que é o XML Server, tendo por função "conversar" com o sistema legado, transformar os dados no padrão XML conhecido e via HTTP trafegar essa informação até o destino (Banco Central).

Figure 2.5 – Exemplo da utilização de XML na Integração com Sistema Legado



2.7.3 Replicação de Dados

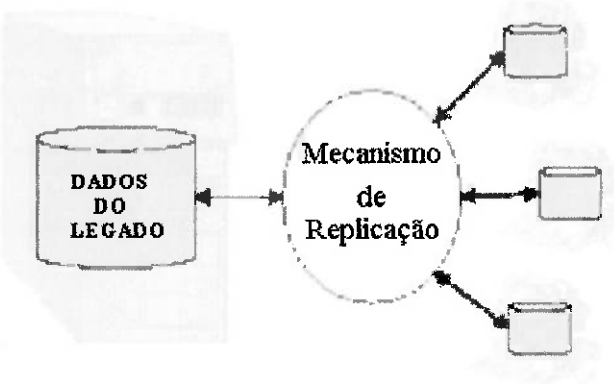
A replicação de banco de dados é um processo em que a base de dados central é "espelhada" em diversas bases de dados locais. Essa técnica permite acessar localmente os bancos de dados, garantindo acesso mais rápido, além de prover maior disponibilidade à aplicação, pelo fato que os dados estão descentralizados, isto é, mesmo se em uma das réplicas os dados estiverem indisponível a aplicação vai acessar estes dados mapeando outros caminhos alternativos.

As mudanças aplicadas em uma base local são capturadas e armazenadas localmente antes de serem enviadas ao repositório central. Um mecanismo de replicação é o responsável pela integridade dos dados entre a base central e as bases locais.

Segundo Comella-Dorda [Comella-Dorda et al, 2000], em mainframe a técnica de replicação de dados é usada para acesso descentralizado aos dados armazenados do legado, portanto instâncias locais de banco de dados modernizados são partes replicadas do banco de dados central.

É importante observar que os sistemas legados que provêem replicações de dados beneficiam as novas aplicações no processo de integração, pois estas aplicações utilizarão acesso local a um banco de dados moderno, em vez dos problemas de acesso remoto a um repositório de dados antigo. A figura 3.6 ilustra a descentralização dos dados no processo de replicação.

Figure 2.6 - Replicação de dados



Fonte: Adaptado de [Comella-Dorda et al, 2000]

A vantagem neste processo é a alta disponibilidade dos sistemas computacionais. Essa vantagem é muito valorizada principalmente em aplicações comerciais que, tipicamente, envolvem transações e acessos a dados. Essas aplicações compreendem sistemas bancários e comércio eletrônico, onde a indisponibilidade de um serviço pode representar substanciais perdas financeiras. Como desvantagem é preciso um mecanismo de replicação não muito simples que garante a integridade dos dados, acesso de escrita simultânea, etc.

2.8 Modernização da Lógica Funcional

A modernização funcional é abrangente por tratar-se tanto da modernização de dados, cobrindo portanto todas as alternativas discutidas nos tópicos anteriores, como também da modernização da lógica funcional do negócio.

Trata-se de uma modernização de "caixa branca", pois é necessário analisar o código fonte geralmente escrito em COBOL para conhecer a fundo as funções e regras de negócio embutidas. Uma vez conhecidas essas funções são agrupadas logicamente a fim de serem modernizadas.

2.8.1 Common Gateway Interface (CGI)

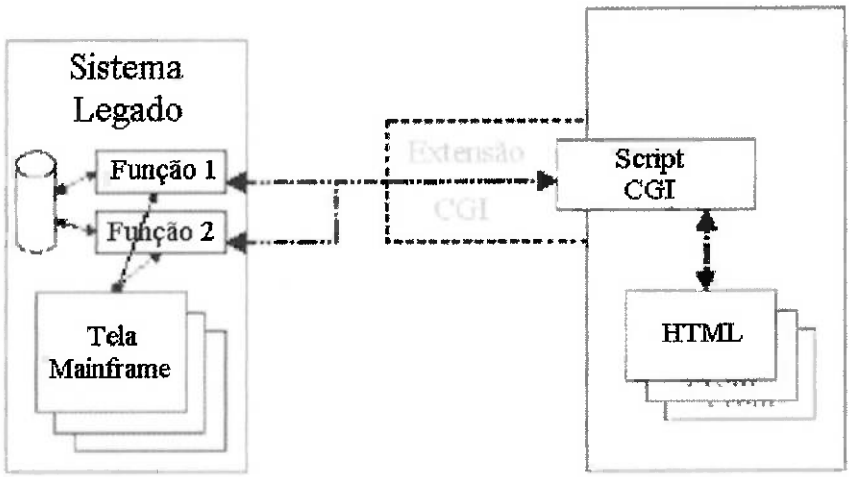
Segundo Comella-Dorda [Comella-Dorda et al, 2000], o CGI é um padrão de interfaces entre as aplicações externas e as informações dos servidores. São programas executados nos servidores (HTTP ou Servidores WEB) que respondem solicitações de usuários criando páginas HTML dinamicamente [Tittel, 1998].

A modernização de sistemas legados usando CGI visa a transformação de telas mainframes, por exemplo, telas de interação on-line mainframe (Monitor Transacional IMS/CICS) por páginas dinâmicas HTML, ou seja, é uma forma de prover acesso rápido de WEB aos recursos existentes no mainframe.

A nova tela gerada pela aplicação CGI não precisa estar "casada" com a tela antiga, como acontece na técnica de *screen scraping* discutida anteriormente. Neste caso as aplicações CGIs interagem diretamente com o negócio, ou os dados do legado. Entretanto é relativamente difícil a implementação CGI, sendo geralmente escrito em linguagem C ou Perl (linguagem de script).

A figura 2.7 mostra uma típica aplicação na qual um servidor *WEB* com extensões CGI solicita algumas funções para o sistema mainframe, das quais são respondida e servida remotamente no navegador (browser). Neste processo a aplicação CGI faz a "tradução" dos dados para HTML.

Figura 2.7 - Sistema Legado x CGI



Fonte: Adaptado de[Comella-Dorda et al, 2000]

A principal desvantagem de se utilizar *scripts* CGI é que para atender a cada requisição do usuário o servidor *Web* precisa criar um novo processo, ou seja, cada processo produz uma nova conexão com o banco de dados utilizado e o servidor *Web* tem que esperar até que os resultados sejam processados. Outro ponto negativo é com relação à segurança, uma vez que os arquivos (*scripts*) não ficam inteiramente protegidos, pois devem ser armazenados nos sub-diretórios CGI-bin do servidor.

Comella-Dorda [Comella-Dorda et al, 2000] trata o CGI como uma forma geral de comunicação entre servidores Web e programas mainframes, que respondem a solicitação de usuários formatando páginas HTML dinamicamente. Em adição a esse material, é relevante comentar a solução da Microsoft para a criação de páginas dinâmicas, a tecnologia ASP(*Active Server Pages*).

O ASP (*Active Server Pages*) fica embutido nas páginas HTML, não sendo pré-compilado. O servidor *Web* processa um arquivo utilizando uma DLL (ASP.DLL) que irá interpretar os comandos ASP embutidos na página HTML. As principais vantagens do ASP sobre o CGI são: segurança na execução, facilidade na programação (pode utilizar comandos em VBScripts, JavaScript e HTML), utilização de uma linguagem interpretada e não compilada, facilidade de acesso as bases de dados e proteção do código fonte da página, pois o servidor retorna somente o resultado HTML.

2.8.2 Encapsulamento Orientado a Objetos

Tem se usado muito a orientação a objetos para implementar sistemas complexos. Um sistema orientado a objeto é mais fácil de entender, devido à possibilidade de mapear sistemas do mundo real em objetos.

Os sistemas legados são sistemas centralizados. A modernização destes sistemas usando a tecnologia orientada a objetos pode ser sinônimo de descentralização portanto, de acordo com Comella-Dorda [Comella-Dorda et al, 2000], surge à necessidade de um mecanismo de comunicação para suportar a distribuição dos objetos.

De acordo com [Wallnau 1997 apud Comela-Dorda et al, 2000], a Tecnologia de Objetos Distribuídos (DOT – *Distributed Object Technology*) é a combinação da tecnologia distribuída com a Orientação a Objetos.

A DOT estende a tecnologia de objetos para o ambiente distribuído visando maior escalabilidade das aplicações usando objetos *middleware*. Os *middleware* são conjunto de componentes de softwares residentes acima do sistema operacional, que possibilita expandir as aplicações e redes, tornando-se uma ótima ferramenta para lidar com a complexidade de integração das aplicações.

Esses *middlewares* são os responsáveis por fornecer serviços de redes especializados, que são compartilhados entre as aplicações. O *middleware* mais conhecido é o CORBA – *Common Object Request Broker Architecture*, padrão desenvolvido pela OMG - *Object Management Group* (Grupo de gerenciamento de Objetos). O CORBA possui um bom mecanismo de chamada de procedimento remoto e um rico conjunto de serviços [CORBA] [OMG].

Uma das grandes dificuldades nesta técnica está na tradução do código procedural e das estruturas de dados do legado para o modelo conceitual da Orientação a Objetos. Por exemplo, uma aplicação individual é representada por objetos, serviços comuns são representados por objetos, os dados do negócio são representados por objetos, entre outros. Essa tradução não é trivial de se obter, pois

requer análise criteriosa do código fonte, decomposição e abstração do modelo orientado a objeto.

De acordo com Comella-Dorda [Comella et al, 2000], dentre as inúmeras dificuldades no encapsulamento dos sistemas legados duas em especial são relevantes:

- ***Definição apropriada das relações entre os objetos*** - trata-se das dificuldades encontradas na decomposição dos objetos partindo de uma estrutura procedural do sistema legado, e ainda as dificuldades de separar as interfaces da lógica de negócio;
- ***Necessidade para integrar estruturas de serviços*** – em ambiente distribuído a maioria dos *middlewares* provê um conjunto de serviços tais como segurança, transação, persistência etc, entretanto para obter algum serviço esperado só é possível com a integração de dois ou mais *middlewares*. Essa integração é no mínimo problemática, segundo Seacord [Seacord, 1999], podendo haver incompatibilidade entre os produtos.

A fim de solucionar estes problemas relacionados com integração de serviços, as indústrias de software têm desenvolvido os Servidores de Aplicações ou *Application Server*. Um servidor de aplicação é um produto que integra um conjunto de serviços e define um modelo de componente. Um componente desenvolvido segundo um respectivo modelo provê todos os serviços encapsulados a aquele modelo.

2.8.4 Componentização

Um componente de software é um código encapsulado que implementa a lógica do negócio e tem interfaces bem definidas, não são aplicações complexas e não podem rodar sozinhos, ou seja, isoladamente. Esses componentes são reusáveis e deve ser usado como peça de quebra-cabeça para resolver uma grande variedade de problemas.

A idéia de componente de software é muito poderosa, pois através das combinações desses componentes é possível criar soluções para problemas complexos. Componentes reusáveis são bem vindos, pois promovem um rápido desenvolvimento da aplicação.

O processo de criação dos componentes é um processo orientado a objetos. Os objetos que futuramente serão componentes devem seguir um modelo ou padrão pré-determinado. Inicialmente existiam inúmeros modelos de componentes todos incompatíveis; atualmente há alguns padrões de mercado, a saber:

- DNA – *Distributed Internet Architecture* (Arquitetura Distribuída na Internet) – é uma solução antiga da Microsoft, que de uma certa forma foi "substituída" pela tecnologia .NET. É de fato padrão de mercado pela forte influência da empresa Microsoft na área da tecnologia da informação [Comella-Dorda et al, 2000].
- CORBA 3 – *Common Object Request Broker Architecture* – é um modelo para a especificação CORBA, desenvolvido pelo Grupo de Gerenciamento de Objetos (OMG) que é um grupo de padrões aberto cuja missão é fornecer padrões para guiar as indústrias, e construir padrões para gerenciamento de objetos, provendo uma estrutura de trabalho comum para o desenvolvimento [OMG].
- EJB – *Enterprise JavaBeans* – é uma solução da Sun Microsystems, trata-se de uma arquitetura para componentes no lado servidor que possibilita e simplifica o processo de construir aplicações de objetos distribuído em Java [Deitel et al, 2001][SUN 2003].

Para exemplificação será discutido o uso do EJB para ilustrar a modernização do legado através de componentes, não significa que EJB é superior aos outros modelos, esse exemplo pode ser aplicado usando outros modelos de componentes.

Componentes EJB são conhecidos como *Enterprise Javabeans* (pode ser referenciados como *beans*). De acordo com Comella-Dorda [Comella et al, 2000], cada *beans* pode encapsular um pedaço da lógica de negócio, eles são disponibilizados juntos com o Servidor de Aplicação [SUN 2003].

Servidores de Aplicação é o ambiente *framework* de execução dos componentes, e tem como função gerenciar o componente e provê um conjunto de serviços comuns aos componentes (acesso remoto, segurança, persistência, transações, concorrência e acesso a recursos). Isso permite que os desenvolvedores de componentes foquem na solução dos problemas do negócio, não se preocupando com os demais serviços.

O Servidor de Aplicação também é responsável para manter os componentes de negócio disponível para serem utilizados quando necessário. È importante mencionar sobre o RMI – *Remote Method Invocation* ou Invocação Remota de Método, responsável por permitir a comunicação entre os processos e prevêem outros serviços relacionados a comunicação.

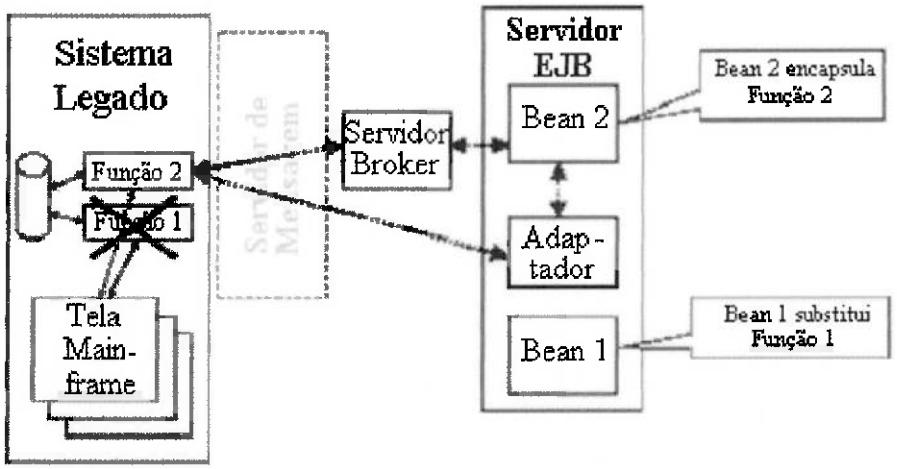
2.8.4.1 Exemplo de Modernização usando Componentes

O primeiro passo para o encapsulamento da lógica dos sistemas legados para criação dos componentes usando EJB é separar a interface das unidades de lógicas de negócio, como mostra a figura 2.8. As dificuldades encontradas para separação das interfaces dependem muito de como elas foram definidas no sistema legado, em alguns sistemas as interfaces estão mais segregadas da lógica de negócio tornando-se mais fácil a separação.

O próximo passo no encapsulamento da lógica de negócio é construir um único ponto de contato com o sistema legado. Isso é uma boa forma para centralizar todo o conhecimento da comunicação em um único lugar. Este único ponto de contato com o legado pode ser implementado usando componentes *beans* chamado Adaptador (*Adapter)* ou Serviço Broker (*Service Broker)*, que é um componente de software externo ao EJB Server.

E finalmente o último é implementar um "encapsulador" *bean* para cada módulo do sistema legado, como mostra a figura 2.8.

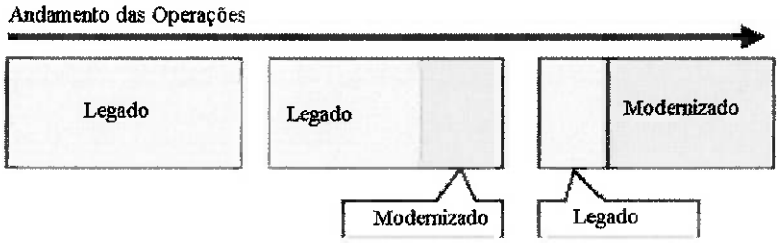
Figure 2.8 – Modernização de legados Usando componentes EJB



Fonte: Adaptado [Comella-Dorda et al, 2000]

Segundo Comella-Dorda [Comella et al, 2000], existem várias vantagens na componentização quando se trata de encapsulamento da lógica de negócio. Uma vez entendido e separado as funcionalidades do sistema, pode-se re-implementar o "encapsulador" *bean* para cada função de forma incremental flexível até que todo o antigo sistema legado seja substituído. Pode dizer que o encapsulamento da lógica de negócio provê um esquema para substituir o antigo sistema legado de forma incremental [Comella-Dorda et al, 2000] [Seacord, 2001].

Figura 2.9 – Modernização Incremental



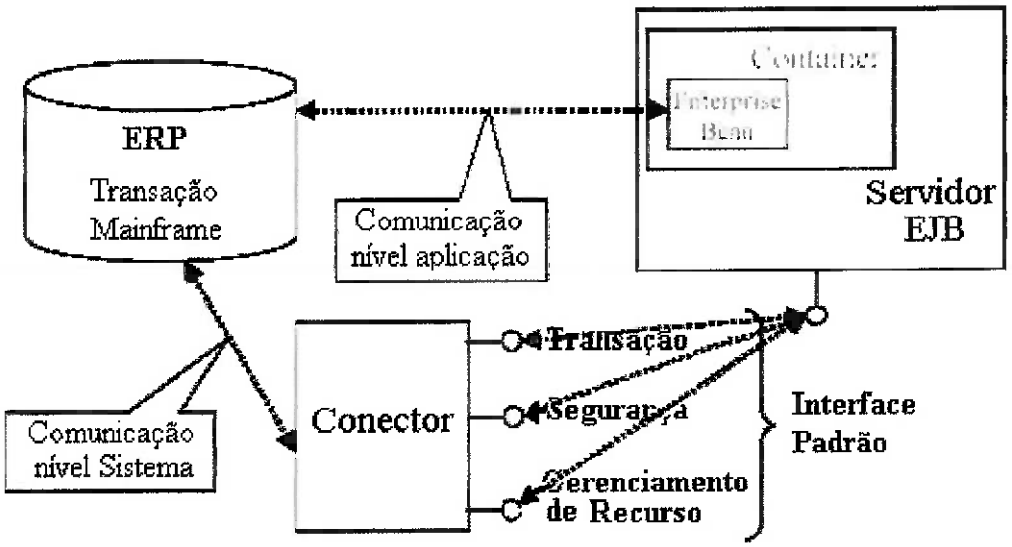
Fonte: [Seacord et al, 2001]

2.8.4.2 Conector

A modernização dos sistemas legados usando componentes beans significa que a lógica do negócio será encapsulada por componentes EJB, componentes externos ao sistema legado. Portanto surge a necessidade de um mecanismo para coordenar o gerenciamento dos objetos (*EJB Server*) e o sistema legado. Esse mecanismo é chamado de Conector [Comella et al, 2000] [Seacord, 2001].

Neste caso os conectores são desenvolvidos pela Sun e promete simplificar a integração com os recursos do legado [SUN_CONN]. De acordo com Comella-Dorda [Comella-Dorda et al, 2000], conector é um padrão de interfaces de sistemas que gerência transações, segurança, e a comunicação entre os recursos do legado e do servidor de aplicação. O conector é "plugado" no *Application Server* provendo a conectividade entre o sistema legado e o *Application Server*, como mostra a figura 2.10 abaixo.

Figure 2.10 – Modernização de legados usando conectores



Fonte: Adaptado [Comella-Dorda et al, 2000]

2.8.5 Técnicas Procedurais

Uma outra técnica utilizada na prática é a programação usando linguagem de baixa plataforma de forma procedural.

Esta técnica consiste em traduzir o código procedural dos sistemas mainframe para um novo código também procedural, porém em linguagem de baixa plataforma (por exemplo Visual Basic, Delphi, etc.). Essa técnica provê soluções rápidas com interfaces modernas, dando uma nova "cara" para as aplicações mainframes. Na prática está técnica é muito utilizada, pois os próprios analistas de mainframe têm condições de entender e fazer essa modernização, sem grandes esforços.

Este capítulo apresentou um conjunto de técnicas de modernização de legados, a maioria delas obtida de referência bibliográfica complementada com a experiência profissional do autor. A seleção de uma técnica para modernização de legados envolve riscos e é sempre uma atividade preocupante para o gerente de projeto. O próximo capítulo trata-se da gerência do risco e como pode ser aplicada essa gerência em um processo de modernização.

CAPÍTULO 3

GERÊNCIA DE RISCOS

Neste capítulo são apresentados os principais objetivos da gerência de risco e sua importância. A gerência de risco abrange os riscos do projeto de forma geral, não limitados apenas em riscos técnicos. Apresenta-se a técnica de gerência de riscos segundo o PMBOK, com ênfase especial nos subprodutos de cada fase do processo de gerência.

3.1 Introdução

Segundo Machado [Machado, 2002], a gerência de risco começou a ser largamente utilizada no século XX principalmente nas áreas de saúde, finanças e seguros de vida. O negócio das empresas que atuam nessa área é a gerência de riscos. Todos os projetos tratam riscos, pois o lucro depende do balanceamento entre as oportunidades e os riscos bem calculados.

Segundo [BERNSTEIN, 1997], a palavra "risco" deriva do italiano antigo *risicare*, que significa "ousar". O Dicionário Aurélio define risco como sendo uma situação em que existe probabilidade mais ou menos previsível de perda ou ganho.

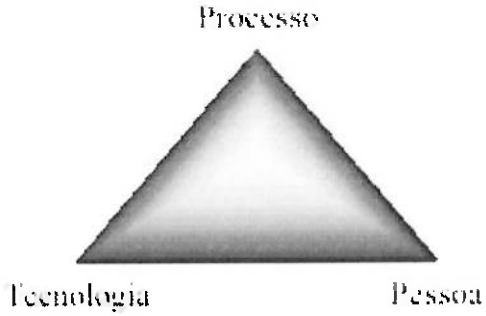
O sucesso do projeto está baseado em oportunidades que tratam o valor do produto a ser entregue; e os riscos, que tratam das incertezas de se obter esse produto dentro de algumas condições pré-estabelecidas (prazo, custo, qualidade, etc.).

Em tecnologia de informação é comum lidar com um alto nível de incertezas. Segundo Lister [Lister, 1997], o desenvolvimento de software é composto por três forças:

- Processos utilizados;
- Pessoas envolvidas no projeto;
- Tecnologia empregada.

Qualquer deficiência em alguma dessas áreas será refletido no risco do projeto [LISTER, 1997].

Figura 3.1 – Triângulo da força do desenvolvimento de software



As incertezas no projeto de software são fontes de riscos em potencial. Dentre elas podem-se citar, a imaturidade nos processos, adequação de pessoas com qualidade técnicas e perfil apropriado para exercer uma atividade e a constante mudança na tecnologia.

3.2 Importância da Gerência de Riscos

A gerência de riscos é um conjunto de procedimentos para se resolver os riscos, ou seja, com a aplicação ou execução da gerência de risco é possível minimizar falhas de projeto de software [BOEHM, 1991].

A gerência de riscos é uma ferramenta eficiente com o objetivo de controlar os efeitos de eventos negativos e também pode potencializar os efeitos positivos. A gerência de risco tem dois objetivos principais: prevenção e mitigação de riscos. Com a execução das técnicas da gerência de risco, os problemas passam a ser

preventivamente identificados e resolvidos antes de se tornarem problemas mais graves.

De acordo com Boehm [BOEHM, 1997], o principal objetivo da gerência de risco é auxiliar os analistas de software a identificar e analisar os focos que são fortes candidatas a gerar problemas e formular soluções preventivas.

3.2 Gerência de Risco segundo o PMBOK [PMI, 2000]

Segundo o PMBOK, a gerência de risco inclui os seguintes processos:

- ***Identificação dos Riscos*** – Consiste em determinar quais os riscos que podem afetar o projeto e documentar suas características.
- ***Análise Qualitativa dos Riscos*** – Executar uma análise qualitativa dos ricos e impactos priorizando seus feitos nos objetivos do projeto.
- ***Análise Quantitativa dos Riscos*** – Medir e avaliar numericamente a probabilidade de ocorrência e as conseqüências dos riscos e estimar as suas implicações nos objetivos do projeto.
- ***Desenvolvimento das Respostas aos Riscos*** – Desenvolver procedimentos e técnicas para o aproveitamento de oportunidades e reduzir ameaças.
- ***Controle e monitoração dos Riscos*** – Consiste na monitoração dos riscos residuais, identificar novos riscos, execução do plano de gerência de riscos, e avaliar seus efeitos em todo o projeto.

Embora todas as atividades da Gerência de Riscos sejam relevantes, o escopo desta monografia está restrito em identificar e quantificar os riscos, portanto serão abordados os três primeiros tópicos: Identificação dos riscos; Análise Qualitativa e Análise Quantitativa dos riscos.

As próximas seções resumem estas atividades, com ênfase especial nos produtos por ela gerados, que serão o foco da proposta desta monografia.

3.3 Identificação dos Riscos

É a atividade inicial da gerência de risco, que consiste em documentar todas as possibilidades de riscos existentes no projeto, descrever a característica de cada um deles sem a preocupação de ter uma solução imediata para as questões identificadas. A documentação formal dos riscos é muito importante, pois torna os visíveis a todos, não podendo ser ignorados.

3.3.1 Categorias de Riscos

Segundo o PMBOK [PMI, 2000], os riscos podem ser identificados e organizados em categorias de riscos. Essas categorias podem ser:

- Riscos Técnicos – envolve os riscos técnicos, tais como dificuldade de implementação, tecnologia complexa, problemas com desempenho, qualidade, etc.
- Riscos de Gerência de Projeto – problemas com a administração de projetos, tal como alocação precária de tempo e recursos, plano de projeto inadequado, etc.
- Risco Organizacional – problemas como conflitos de recurso com outros projetos dentro da organização, competências mal definidas, etc.
- Risco Externo – envolve os riscos sobre os quais a equipe de projeto não tem domínio, tal como mudanças nas leis, mudança de prioridades do responsável pelo projeto, etc.

Serão considerados nesta monografia os riscos técnicos de uma forma geral e os riscos de custo e de cronograma associados à escolha da tecnologia de modernização de legados.

3.4.2 Métodos para Identificar os Riscos

Existem vários métodos para a identificação de riscos, dentre eles o Instituto de Gerência de Projeto (PMI) destaca *checklists*, *brainstorming*, Delphi, fluxogramas, entrevistas entre outros.

O objetivo desta monografia é mapear os riscos na modernização de sistemas legados de uma forma bem abrangente, ou seja, contemplar custo, esforço, prazo e qualidade. Portanto todos os métodos citados abaixo poderiam ser aplicados a esta monografia. O autor usou a técnica de entrevistas, fez pesquisas literárias que abordam o assunto de modernização de legados e sua própria experiência profissional.

3.4.1.1 Entrevistas

A entrevista orientada a risco, com a participação de vários especialistas com perfis diferentes pode auxiliar na identificação dos riscos ainda não percebidos, e até mesmo na obtenção de diversas visões dos riscos, de acordo com o PMBOK [PMI, 2000].

O primeiro passo para a aplicação desse método é a identificação dos entrevistados e a preparação da lista de perguntas que serão feitas durante a entrevista. O segundo passo é a realização da entrevista tendo como foco as perguntas preparadas pelo entrevistador. As vantagens desse método são a identificação de riscos ainda não percebidos, a obtenção de diversas visões dos riscos, pois os entrevistados podem ter perfis diferentes, e a facilidade para sua aplicação. A desvantagem é que esse método é dependente do entrevistador e do entrevistado, pois ambos não devem definir ou responder as perguntas de modo que limitem a entrevista.

3.4.1.2 Pesquisas literárias

Para elaboração desta monografia pesquisou-se diversos trabalhos relacionados com o tema modernização de sistemas legados. Entre eles é importante mencionar o relatório base desta monografia que é o relatório técnico da SEI

(CMU/SEI-2000-TN-003) publicado em 2000. Todos os trabalhos pesquisados estão referenciados na bibliografia.

3.4.1.3 Experiência do autor

O autor trabalha com sistema mainframe em uma grande instituição financeira e já participou de projetos de modernização de sistemas legados. Histórias de sucesso e de insucesso contribuem para apontar fatores que podem ser riscos de projetos semelhantes.

3.4.2 Produto final do Processo de Identificação dos Riscos

Segundo a proposta deste trabalho, no final desta fase de identificação dos riscos tem como resultado uma lista das possíveis situações comprometedoras do projeto associado às alternativas de modernização.

3.5 Priorização Qualitativa dos Riscos

Esse processo avalia o impacto e probabilidade dos riscos identificados no passo anterior. A análise qualitativa dos riscos determina a importância de se tratar riscos específicos [PMI, 2000].

A probabilidade e as conseqüências de um risco podem ser descritas como:

- **Escala de probabilidade** – Escalas cardinais, por exemplo, onde os extremos da escala são: 0.0 (nenhuma probabilidade) e 1.0 (certeza), também indicam a probabilidade da ocorrência do evento de risco.
- **Escala de impacto de riscos** – reflete a severidade de seu efeito no objetivo do projeto. É possível construir uma tabela de impacto

usando escala com numeração cardinal ou ordinal, indicando o impacto no projeto se um determinado risco ocorrer.

- **Matriz de graduação da probabilidade / impacto de risco** – pode ser construída uma matriz com as graduações de riscos qualitativos (muito alto, alto, moderado, baixo e muito baixo) indicando a probabilidade de ocorrer o risco.

Para cada projeto individual o objetivo deste processo é fazer uma triagem de quais são os riscos mais relevantes do ponto de vista qualitativo.

3.5.1 Produto final do Processo de Priorização Qualitativa dos Riscos

O resultado deste processo é uma lista priorizada dos riscos segundo critérios específicos. Os riscos mais significativos devem ganhar atenção especial do gerente e da organização.

3.6 Priorização Quantitativa dos Riscos

O processo de priorização quantitativa dos riscos analisa numericamente a probabilidade de cada risco e seu impactos nos objetivos do projeto [PMI, 2000].

Esse processo requer a lista de erros identificada da etapa anterior. Os processos de análise qualitativo e quantitativo podem ser usados juntos ou separados. Algumas variáveis tais como orçamento e tempo são fatores determinantes na escolha dos processos a serem utilizados.

Segundo PMI [PMI, 2000], as probabilidades quantitativas podem ser mapeadas por:

- **Entrevistas** – Entrevista técnicas são bem vindas para quantificar a probabilidade e a conseqüência dos riscos no projeto. Uma boa prática para

dar início na quantificação dos riscos é uma entrevista com as partes envolvidas. As informações obtidas devem ser quantificadas.

- **Análise de Sensibilidade** – Determinam quais são os riscos que tem o maior impacto no projeto
- **Árvore de Decisão** - Trata-se de um diagrama que descreve uma decisão sob consideração e as implicações de escolher uma ou outra das alternativas disponíveis. A resolução da árvore de decisão deve mostrar o melhor caminho para o tomador de decisões.
- **Simulação** – É um modelo que traduz as incertezas especificadas a um nível detalhado. Esse método simula o projeto completo, considerando os impactos dessas incertezas. Simulações de projetos são, tipicamente, executadas usando a técnica de Monte Carlo, que não é escopo desse trabalho.

3.6.1 Produto final do Processo de Priorização Quantitativa dos Riscos

O resultado deste processo é um método de critérios capaz de mapear qual risco tem maior impacto sobre o outro. Com isso tem-se uma lista priorizada dos riscos quantificados, segundo os critérios definidos.

Como o objetivo deste trabalho é produzir um roteiro de orientação ao gerente que precisa selecionar uma técnica de modernização de legados, no próximo capítulo será detalhado cada produto resultado das atividades descritas neste capítulo, considerando-se as técnicas de modernização apresentadas no Capítulo 2.

CAPÍTULO 4

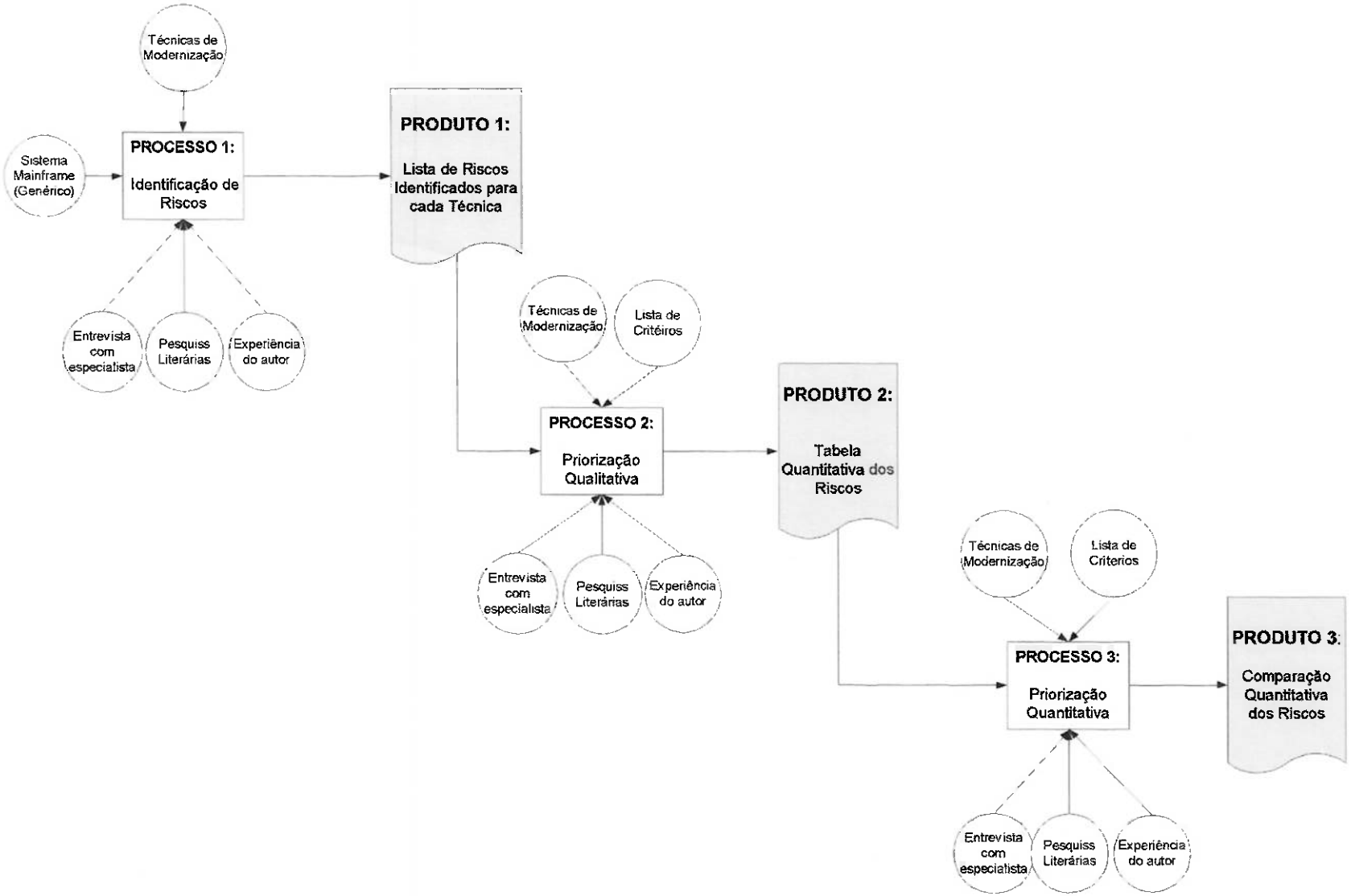
ROTEIRO PARA APOIO À SELEÇÃO DE ESTRATÉGIA DE MODERNIZAÇÃO DE UM SISTEMA LEGADO

O objetivo deste capítulo é fornecer uma ferramenta de acesso rápido para ajudar na escolha de uma estratégia de modernização dos sistemas legados. Essa ferramenta, na forma de um roteiro, tem os seguintes produtos:

- Lista de identificação dos riscos
- Priorização qualitativa dos riscos
- Comparação quantitativa entre as técnicas de modernização

A geração desses produtos pode ser resumida na figura abaixo, e será explorado na seção seguinte. No final desse capítulo é apresentado o roteiro e como utilizá-lo.

Figura 4.1 – Produto da Gerência de Risco



4.1 Identificação dos Riscos

Neste trabalho, a identificação dos riscos consiste em determinar quais os riscos poderiam afetar o projeto de modernização dos sistemas legados, segundo uma determinada técnica de modernização. Portanto, dadas as técnicas de modernização vistas no capítulo 2, identificaram-se os riscos de cada técnica em um projeto de modernização do sistema legado.

Conforme ilustra a figura 4.1, dado uma determinada técnica, os riscos foram identificados através de:

- Pesquisa literária - pesquisas referentes ao assunto relacionado com modernização de sistemas legados para elaboração deste trabalho
- Experiência do autor – por trabalhar com sistema mainframe em uma grande instituição financeira do país e ter participado de projetos de modernização desses sistemas
- Entrevistas - entrevistas[2] na fábrica de software da instituição financeira em que o autor trabalha.

Neste trabalho, para melhor entendimento, as técnicas de modernização seguiram a divisão apresentada na seção 2.5:

Os riscos identificados estão descritos na tabela abaixo. Esses riscos foram escolhidos com base na lista de riscos da instituição financeira em que o autor trabalha.

---

[2] Foram consultados o coordenador e participantes da Fabrica de software departamento responsável por todos os projetos de modernização de sistemas dentro da Instituição Financeira, bem como o histórico de projetos.

Riscos/Oportunidades		Possível perda/ganho
1	Risco de desempenho	Perda de desempenho, lentidão na execução.
2	Risco de manutenção	Esforço envolvido na Manutenção
3	Risco do conhecimento técnico da equipe	Introdução de problemas ou atrasos no projeto em função do conhecimento da tecnologia pela equipe envolvida.
4	Risco da complexidade da tecnologia	Introdução de problemas ou atrasos no projeto em função da complexidade da tecnologia envolvida
5	Risco de inflexibilidade da aplicação	Perda de liberdade de modernizar no sentido de dependência da estrutura do legado (interface, dados ou lógica)
6	Risco de integridade de dados	Possível divergência de dados entre as bases de dados.
7	Risco de indisponibilidade da aplicação	A aplicação pode ficar parada por alguma falha.
8	Risco de implementação	Introdução de problemas ou atrasos no projeto em função do conhecimento inadequado da parte modernizada.
9	Risco de segurança	Perda de segurança da aplicação
10	Risco de qualidade	Introdução de novos defeitos na aplicação, ou não resolver os erros existentes na aplicação antiga.
11	Risco de sincronismo	Perda de sincronismo entre a parte modernizada e a aplicação legada
12	Risco de custo	Alto custo para implementar o projeto de modernização
13	Risco de cronograma	Solução demorada para implementar o projeto de modernização

Embora não seja escopo dessa monografia analisar as contra medidas de cada risco, em alguns casos são citadas algumas dessas medidas, a título de ilustração e complementação.

4.1.1 Identificação de Riscos da Técnica de Interfaces

A técnica *Screen Scraping* é representante de técnica de interfaces. Os riscos identificados nesta técnica estão descritos a seguir.

A identificação de riscos na técnica de *screen scraping* pode ter dois enfoques. O primeiro trata da aquisição da ferramenta *scraping*, e o segundo trata-se do desenvolvimento dessa ferramenta. Neste ponto é analisado o primeiro enfoque. Na seção 4.3 é feita a comparação quantitativa entre os 2 enfoques.

1 - Risco de Performance:

Nesta técnica há alta probabilidade de perda em performance. Segundo [Attachmate, 2000] esse risco está relacionado com a velocidade com a qual o aplicativo legado pode apresentar telas consecutivas e não exatamente com o programa de captura de dados. Por exemplo, ser for preciso 3 segundos para se apresentar cada uma das 5 telas de uma operação, o tempo de resposta da apresentação gráfica completa será de 15 segundos.

É importante lembrar que no fluxo das informações entre o legado e a tela do usuário existe um componente a mais, que é um software (ferramenta de *scraping*) responsável pela tradução dos dados ou telas do mainframe para o formato moderno (HTML/GUI), o que demanda tempo.

2 - Risco de Manutenção:

Outro risco identificado está relacionado com a manutenção da aplicação *screen scraping*. Conforme Comella-Dorda [Comella, et al, 2000], como nesta técnica a tela nova deve estar "casada" com a tela antiga do mainframe, qualquer alteração na tela do mainframe exige alterar o código de controle das interações entre as aplicações. Neste caso existe um protocolo de comunicação entre as telas do mainframe e a nova tela, com uma determinada seqüência de passagem de parâmetros.Se essa seqüência for alterada, ou, por exemplo, mudar o tamanho de um campo, a aplicação *screen scraping* deve ser alterada também, pois os parâmetros devem estar bem ajustados para execução da aplicação.

3 - Risco de Conhecimento Técnico da Equipe:

Considerando o caso de aquisição da ferramenta *screen scraping*, a equipe técnica não precisa conhecer detalhes técnicos desta ferramenta. Se for necessário desenvolvimento, a programação é coisa simples apenas trata-se de programação.

4 - Risco de Complexidade da Tecnologia:

Não envolve tecnológica complexa, pois a utilização dessa ferramenta *scraping* é fácil, basta associar cada campo da tela *scraping* com os dados da tela do

mainframe. E no caso de desenvolvimento trata-se apenas de linguagem de programação.

5 - Risco de Inflexibilidade da Aplicação:

Visto que a maioria das aplicações *screen scraping* envolve a tradução da velha tela do mainframe pela ferramenta de *screen scraping* em uma tela nova e moderna baseada nos padrões GUI ou HTML, essa nova tela deve estar "casada" com a tela do mainframe. Isso significa inflexibilidade no processo de modernização, ou seja, a liberdade de solução é restrita, uma vez que a tela nesse caso deve ter a mesma informação da tela antiga. Não se cogita a mudança da tela nesta técnica, e por tratar-se de um processo um tanto inflexível, esta técnica é mais bem aplicada em sistemas com alto índice de estabilidade cujas interfaces gráficas já estão bem sedimentadas, tendo como principal objetivo melhorar a usabilidade e a aparência.

6 - Risco de Integridade de dados:

Esse risco não se aplica para a técnica de *screen scraping*.

7 - Risco de Indisponibilidade da aplicação:

O risco de indisponibilidade da aplicação existe, uma vez que se tem mais um software envolvido no processo. Portanto podem ocorrer problemas associados ao novo software de diversas naturezas (falhas de comunicação, servidor Web com problema, etc.) eventualmente deixando a aplicação indisponível.

8 - Risco de Implementação:

No caso de aquisição da ferramenta, o risco de implementação não se aplica. No caso de desenvolvimento da ferramenta, o mecanismo de sincronismo deve ganhar atenção especial.

9 - Risco de Segurança:

Esse risco não se aplica para a técnica de *screen scraping*, uma vez que toda a execução do processo ocorre no mainframe, e os parâmetros passados para o legado são previamente consistidos.

10 - Risco de Qualidade:

De acordo com a explicação do risco de inflexibilidade (5), há probabilidade de transferência dos problemas antigos para a nova aplicação.

Segundo Comella-Dorda [Comella-Dorda et al, 2000], *screen scraping* é basicamente uma "casca" (*makeover*) para o sistema legado, não resolvendo os problemas ainda existentes nas interfaces. Se o sistema legado tiver problemas de interface relacionados à seqüência dos parâmetros, isso não será resolvido por esta técnica.

11 - Risco de Sincronismo:

Nesta técnica também foi identificada a probabilidade de riscos em perda de sincronismo entre a aplicação do usuário (HTML / GUI) e a aplicação mainframe. Por exemplo, a aplicação do usuário envia uma solicitação para a aplicação mainframe, essa aplicação mainframe deve estar processando a tela correta (ou seja, a mesma tela da aplicação do usuário) para responder corretamente à solicitação. Caso essas comunicações não estiverem bem ajustadas, as aplicações irão se comportar de maneira divergente, resultando em mensagens de erro, ou em um resultado não esperado.

Isso ocorre porque em ambiente on-line mainframe cada tela pode executar um programa diferente e em cada tela a tecla de função[3] se comportará de maneira diferente, uma vez que se trata de programas diferentes executando funções diferentes. No entanto, devido à interface restrita, existe uma alta probabilidade de que a mesma tecla de função faça sentido em outra tela. A navegação das telas é controlada pelo próprio ambiente on-line do mainframe.

Esse risco pode ser mitigado com um bom mecanismo de manipulação de erros e exceções implementados na aplicação cliente, ou seja, na aplicação *screen scraping*. Essa é a parte mais delicada e crítica das aplicações *screen scraping*. Pois no ambiente on-line do mainframe a navegação de telas pode ocorrer através de

[3] Para executar as funções do menu são usadas as teclas de funções de <F1> até <F24>

teclas de navegação ou teclas de funções (onde ocorre processamento). São possíveis essas teclas se confundirem de uma tela para a outra. Ou seja, em uma tela uma determinada tecla é apenas tecla de navegação e em outra, essa mesma tela envolve processamento, ou seja, tecla de função.

12 - Risco de Custo:

O custo envolvido nesse processo é baixo, considerando que as organizações que utilizam sistemas mainframes são grandes organizações. O custo está prioritariamente associado ao esforço do pessoal envolvido, que não é grande pela baixa complexidade da tecnologia. O esforço é reduzido também porque basta o conhecimento da interface (que é visível e dominada pelos usuários), não requerendo a análise profunda dos procedimentos do legado.

13 - Risco de Cronograma:

A implementação usando *screen scraping* é rápida, pois não precisa entendimento detalhado da ferramenta *scraping*.

A tabela abaixo resume os possíveis riscos e oportunidades identificados na técnica de *screen scraping*.

Tabela 4.1 – Identificação de riscos da Técnica de Interfaces

Modernização de	Técnica	Categoria	Ident	Fatores potenciais de riscos
INTERFACES	Screen Scraping	RISCOS		
		Técnico	1	Baixa performance
		Técnico	2	Impacto na manutenção. Tela "amarrada" com a tela do mainframe.
		Técnico	5	Baixa flexibilidade na modernização
		Técnico	7	Pode ocorrer problema relacionado a indisponibilidade da aplicação.

		Técnico	10	baixa qualidade no sentido de resolver o problema existente.
		Técnico	11	Possibilidade de problemas de sincronismo entre a aplicação mainframe e a tela nova do usuário
		OPORTUNIDADES		
		Técnico	3	Equipe de desenvolvimento não precisa estar familiarizada com a ferramenta no caso de aquisição da ferramenta scraping
		Técnico	4	Tecnologia não complexa
		Técnico	6	Não oferece risco de integridade de dados
		Técnico	8	Fácil implementação (no caso de aquisição da ferramenta)
		Técnico	9	Não oferece risco relacionado a segurança
		Organizacional	12	Baixo custo
		Organizacional	13	Solução rápida

4.1.3 Técnicas de Modernização de Dados

As técnicas envolvidas neste tópico são: técnicas usando *Gateways*, técnica de replicação de dados e XML. A modernização de dados consiste normalmente em retirar dados de uma base de dados usando SQL e disponibilizar esses dados para outra aplicação ou banco de dados.

Modernizar dados quando o esquema de banco de dados é complexo se torna difícil, já que normalmente o esquema de banco de dados do mainframe não está documentado [Recchia et al, 2002]. Os riscos identificados nessas técnicas estão descritos a seguir:

1 - Risco de Performance:

Os *gateways* de acessos à banco de dados que permitem acesso remoto, utilizam protocolos síncronos, tornando lento o acesso aos dados legados, dependendo de como esses dados estão relacionados. Por exemplo, se as tabelas estão todas entrelaçadas, para obter um dado é necessário fazer *joins* entre as tabelas.

Isso com certeza terá impacto em performance. Pelo mesmo motivo podem-se ter problemas de performance na utilização de XML.

A modernização de dados usando a técnica de replicação de dados vista na seção 2.7.3 apresenta uma boa performance, visto que os dados serão acessados localmente (bases replicadas).

2 - Risco de Manutenção:

Os riscos na manutenção de dados do sistema (alteração, inclusão, deleção) são menos prováveis nos *gateways*, pois as alterações dos dados indicam simples alterações na nova aplicação. Em XML pode-se ter mudança na formatação da estrutura ou estilo de dados que são transmitidos pelo XML. Não se trata de riscos graves, porem é mais um ponto com o que o gerente deve se preocupar.

Na técnica de replicação de dados são necessárias mudanças maiores, ou seja, deve-se alterar o mecanismo de replicação dos dados, para refletir essas alterações na base de dados replicadas. Essas alterações devem ser sincronizadas. Portanto existe maior probabilidade de risco na técnica de replicação de dados.

3 - Risco de Conhecimento Técnico da Equipe:

Geralmente a probabilidade de não se conseguir uma equipe com conhecimento de *gateways* é muito pequena, visto que é uma tecnologia muito utilizada hoje em dia, e de fácil utilização. A técnica XML é relativamente nova, portanto existe probabilidade de não conhecimento técnico da equipe.

A técnica de replicação é mais restrita, pelo seu alto custo, portanto não é comum vê-lo na prática, conseqüentemente há alta probabilidade de não conhecimento técnico da equipe.

4 - Risco de Complexidade da Tecnologia:

A complexidade da tecnologia envolvida é baixa no caso de *gateways*. No caso de XML e replicação de dados como visto na seção 2.7.2 e 2.7.3 respectivamente há complexidade tecnológica e complexidade na implementação.

5 - Risco de Inflexibilidade da Aplicação:

As aplicações mainframes que usam a tecnologia XML para integração e modernização dos dados têm maior flexibilidade na formatação dos dados, ou seja, os dados independem do formato origem, pois o XML pode definir livremente qual o padrão de formatação dos dados a serem trocados.

Em aplicações XML é possível às organizações trocarem informações sem necessidade de negociar detalhes técnicos.

Quanto à utilização de *gateways* e a técnica de replicação de dados, os programas resultantes são dependentes do formato de dados do mainframe. No caso de replicação de dados essa dependência é ainda maior, visto que as réplicas de dados são um "espelho" dos dados do legado.

6 - Risco de Integridade de dados:

O risco de integridade é relevante na técnica de replicação de dados, pois existe um mecanismo de replicação que deve ser bem implementando. Ele é responsável pela integridade dos dados entre a base central e as bases replicadas. Esse mecanismo implementa diversas funções complexas, entre elas, a lógica de escrita simultânea, etc. Os riscos de integridade são bem remotos para gateways e XML. Esses riscos existem em caso de mau entendimento do relacionamento dos dados e não por falha na aplicação.

7 - Risco de Indisponibilidade da aplicação:

A replicação de dados provê alta disponibilidade da aplicação pelo fato de que os dados estão descentralizados, isto é, mesmo se em uma das réplicas os dados estiverem indisponíveis, a aplicação vai acessar estes dados mapeando outros caminhos alternativos para acesso aos dados. No caso de XML pode se considerar uma remota probabilidade de indisponibilidade da aplicação por envolver o XML Server, que pode estar indisponível. Para os gateways esse risco não se aplica.

8 - Risco de Implementação:

Um dos maiores riscos na modernização de dados de sistemas mainframes envolve a compreensão da organização dos dados, onde é preciso entender como dados se relacionam, aqui classificado como um risco na implementação. Este risco afeta inclusive a modernização da lógica de negócio desses sistemas. A compreensão da organização é prejudicada pelo uso dos comandos COBOL *FILLER* e/ou *REDEFINES*. A dificuldade neste caso, segundo Seacord [Seacord, 2001], é que o *Filler* e o *Redefines* podem acessar a mesma área de dados usando nomes diferentes, de acordo com a lógica do programa, introduzindo um acoplamento entre módulos difícil de identificar. Essa facilidade de programação, usada extensivamente nos programas mainframe, dificulta e muito a modernização dos dados do legado, por que é difícil entender quais são os dados referenciados por diferentes nomes no contexto da lógica do programa, em um determinado instante. Resumindo são muitas variáveis que espelham o mesmo dado e são referenciadas em parte específicas do código.

9 - Risco de Segurança:

Esse risco não se aplica para a técnica que utiliza gateways e XML. No caso de replicação de dados, os dados podem estar replicados externamente ao ambiente mainframe.Isso indica uma probabilidade de risco relacionado com a segurança dos dados, pois o ambiente mainframe é mais seguro como mencionado na seção 2.2.

10 - Risco de Qualidade:

Esse risco não se aplica para as técnicas de modernização de dados

11 - Risco de Sincronismo:

Os riscos de sincronismo se tornam mais evidente na técnica de replicação de dados, pela existência do mecanismo externo de sincronização.

12 - Risco de Custo:

A técnica de modernização de dados de maior custo é a replicação de dados, XML e *gateways* respectivamente.Os fatores de custos atribuídos aqui são referentes a implementação e a aquisição do banco de dados.

13 - Risco de Cronograma:

A implementação mais rápida devido à facilidade de uso é a técnica que utilizam *gateways*. As outras técnicas demandam mais tempo. Esse risco é dependente do risco 3 (conhecimento da equipe técnica).

A tabela abaixo resume os possíveis riscos e oportunidades identificados na técnica de modernização de dados.

Tabela 4.2 – Identificação de riscos da Técnica de modernização de Dados

Modernização de	Técnicas	Categoria	Ident	Fatores de riscos potenciais
DADOS	Gateways (ODBC JDBC)	**RISCOS**		
		Técnico	1	Baixa performance. Dependência do relacionamento entre os dados do legado.
		Técnico	2	Não envolve grande esforço na manutenção.
		Técnico	5	Baixa Flexibilidade na formatação dos dados.
		Técnico	6	Risco de Integridade, não se aplica.
		Técnico	7	Risco de Indisponibilidade, não se aplica.
		Técnico	9	Risco de Segurança, não se aplica
		Técnico	10	Risco de Qualidade, não se aplica.
		Técnico	11	Baixo Risco de sincronismo.
		Técnico	4	As aplicações devem ter conhecimento técnico dos dados do legado.
		OPORTUNIDADES		
		Técnico	3	Não oferece risco de conhecimento técnico da equipe de desenvolvimento.
		Técnico	4	Tecnologia não complexa.
		Técnico	8	Baixo Risco de implementação.
		Organizacional	12	Baixo custo.

		Organizacional	13	Rápida solução.
	XML	**RISCOS**		
		Técnico	1	Baixa performance. Dependência do relacionamento entre os dados do legado.
			2	Envolve esforço considerado na manutenção.
		Técnico	3	Oferece risco de conhecimento técnico da equipe de desenvolvimento. A equipe precisa conhecer a tecnologia
		Técnico	4	Tecnologia Complexa.
		Técnico	6	Risco de Integridade, não se aplica.
		Técnico	7	Risco de Indisponibilidade.
		Técnico	8	Dificuldade de implementação.
		Técnico	9	Risco de Segurança, não se aplica.
		Técnico	10	Risco de Qualidade, não se aplica.
		Técnico	11	Baixo Risco de sincronismo.
		Organizacional	12	Custo moderado, depende do conhecimento da equipe.
		Organizacional	13	O tempo de implantar depende do conhecimento da equipe.
		OPORTUNIDADES		
		Técnico	5	Alta Flexibilidade na formatação dos dados.
	Replicação De Dados	**RISCOS**		
		Técnico	2	Envolve esforço considerado na manutenção
		Técnico	3	Oferece risco de conhecimento técnico da equipe de desenvolvimento. A equipe precisa conhecer a tecnologia
		Técnico	4	Tecnologia Complexa
		Técnico	5	Baixa Flexibilidade na formatação dos dados
		Técnico	6	Alta risco de integridade de dados
		Técnico	8	Dificuldade de implementação

		Técnico	9	Risco de Segurança. Dados residem externamente ao mainframe
		Técnico	10	Risco de Qualidade, não se aplica.
		Técnico	11	Alto Risco de sincronismo
		Organizacional	13	O tempo de implantar depende do conhecimento da equipe
		OPORTUNIDADES		
		Técnico	1	Alta performance, devido acesso local aos dados.
		Técnico	7	Alta Disponibilidade

4.1.4 Técnicas de Modernização Funcional (Lógica)

Essas técnicas de modernização geralmente são mais abrangentes e detalhadas, pois requerem um bom entendimento do código interno e das estruturas internas dos sistemas. É importante mencionar que essas técnicas podem envolver modernização de dados, modernização da lógica interna e modernização das interfaces também. Como representante da técnica de modernização funcional tem-se:

- Técnica usando CGI /ASP
- Modelo OO (Encapsulamento Orientado a Objetos)
- Modelo de Componentes (Componentização).
- Técnicas Procedurais

A técnica de modernização usando CGI trata as interfaces externas da aplicação. Essa técnica não foi mencionada no item de modernização das interfaces porque a comunicação é diretamente feita com o núcleo da lógica do negócio ou com os dados da aplicação, e neste caso, a lógica pode sofrer alterações ou otimizações.

A técnica de modernização utilizando orientação objetos é bastante utilizada, uma vez que a maioria das publicações que aborda modernização do sistema legado trata de reengenharia e/ou engenharia reversa com enfoque de orientação objetos.

Na técnica de modernização utilizando componentes os riscos identificados são os mesmo que na técnica utilizando orientação a objetos, acrescido da necessidade de maior cuidado com a definição, armazenagem e reuso dos componentes.

É importante relembrar que as técnicas de modernização funcional podem ter dois enfoques: técnicas de caixa preta e técnicas de caixa branca conforme mencionado na seção 2.5.

Embora no relatório do SEI [Comella-Dorda et al, 2000] os autores abordem as técnicas de modernização funcional como sendo técnicas de caixa-preta, neste mesmo relatório os autores citam a modernização usando componentes EJB. Esses componentes são desenvolvidos para encapsular a lógica do negócio da aplicação. Portanto, neste caso presume-se que para encapsular a lógica de negócio é necessário o entendimento mais detalhado do sistema e seus módulos funcionais. Sendo assim os autores do relatório da SEI usaram o enfoque de caixa branca e não de caixa preta.

Portanto, nesta monografia, a identificação dos riscos das técnicas de modernização funcional foi tratada de forma mais abrangente, ou seja, de caixa branca, levando-se em consideração os riscos do entendimento da lógica do negócio e o seu mapeamento para o modelo de orientação a objetos.

4.1.4.1 Identificação de Riscos da Técnica Utilizando CGI / ASP

Essa técnica é usada para prover acesso rápido de Web aos recursos existentes no mainframe. Os riscos identificados estão descritos a seguir:

1 - Risco de Performance:

As aplicações CGI´s criam um processo no servidor para cada solicitação do usuário, isto é, para cada solicitação cria-se uma nova conexão com o banco de dados. Do ponto de vista do usuário, isso degrada a performance.

As aplicações ASP têm objetos predefinidos para criar aplicações complexas, como os que permitem acesso a banco de dados ou objetos que gerenciam sessões, ou seja, a conexão com o banco é controlada através desses objetos predefinidos de sessão, portanto as conexões com o banco de dados são otimizadas. Por tanto os riscos de degradação de performance são menos prováveis de ocorrer do que nas aplicações CGI.

2 - Risco de Manutenção:

Os riscos na manutenção são moderados, pois não se deve ter grandes problemas.Porém deve ser considerado como risco na manutenção o tempo e o esforço que será necessário para fazer pequenas alterações, uma vez que a implementação é lenta, a manutenção também será lenta, se comparado com ASP.

3 - Risco de Conhecimento Técnico da Equipe:

Por diversas razões descritas na seção 2.8.1 o ASP é mais utilizado do que o CGI. Portanto as técnicas que utilizam CGI's têm alta probabilidade de riscos de conhecimento técnico da equipe, pois essa tecnologia já está ultrapassada.

4 - Risco de Complexidade da Tecnologia:

Ambas as aplicações envolvem tecnologia complexas, portanto há alta probabilidade de risco de complexidade da tecnologia envolvida no processo de modernização.

5 - Risco de Inflexibilidade da Aplicação:

Essas técnicas são dependentes da estrutura de dados do mainframe, portanto existe uma certa probabilidade de riscos relacionados à inflexibilidade na modernização dos sistemas legados.

6 - Risco de Integridade de dados:

Esse risco não se aplica para a técnica de CGI e ASP.

7 - Risco de Indisponibilidade da aplicação:

Em ambas aplicações pode se considerar uma remota probabilidade de se provocar indisponibilidade da aplicação por envolver o *Internet Server*, por exemplo, que pode ficar indisponível.

8 - Risco de Implementação:

Outro risco a considerar é a dificuldade na implementação de aplicações CGI's, pois essas aplicações são geralmente escritas em linguagem C ou linguagem de script Perl.

As dificuldades na implementação da conexão com o banco de dados também devem ser levadas em consideração. As aplicações ASP usam componentes pré-definidos para fazer a conexão com o banco. Isso facilita e diminui o código de conexão com o banco. Enquanto que a conexão com o banco de dados nas aplicações CGI necessitam de mais código de programação complicando o entendimento.

9 - Risco de Segurança:

A probabilidade de risco relacionado com a segurança da aplicação é evidente em aplicações CGI, uma vez que os arquivos scripts não ficam inteiramente protegidos, pois eles são armazenados em sub-diretórios CGI-bin do servidor.

Nas aplicações ASP os riscos de segurança são menos prováveis, uma vez que o código fonte é protegido, e o servidor retorna somente o resultado da solicitação em formato HTML.

10 - Risco de Qualidade:

Existe uma certa probabilidade de risco relacionado com qualidade da aplicação, no entanto cada aplicação deve ser analisada com mais detalhes. O quesito

qualidade é dependente da equipe e dos processos utilizados no procedimento de modernização.

11 - Risco de Sincronismo:

Esse risco não se aplica para a técnica de CGI e ASP.

12 - Risco de Custo:

Essa técnica é de baixo custo, capaz de modernizar rapidamente as interfaces externas do sistema legado, suportando o padrão Web.

13 - Risco de Cronograma:

Essa técnica não tem grandes problemas com cronogramas, trata-se de desenvolvimento relativamente rápido dependente do conhecimento técnico da equipe.

Os possíveis riscos e oportunidades identificados nesta técnica são mostrados na tabela a seguir.

Tabela 4.3 – Identificação de riscos da Técnica de modernização Funcional (CGI/ASP)

Modernização de	Técnica	Categoria	Ident	Fatores potenciais de riscos
Lógica / Interfaces	CGI	RISCOS		
		Técnico	1	Baixa Performance
		Técnico	2	Risco na manutenção
		Técnico	3	Alta probabilidade relacionada ao conhecimento técnico da equipe. Tecnologia não muito utilizada atualmente.
		Técnico	4	Tecnologia Complexa
		Técnico	6	Risco de integridade, não se aplica.
		Técnico	7	Risco de Indisponibilidade
		Técnico	8	Dificuldade na implementação, utilização de linguagem C ou Perl
		Técnico	9	Baixa Segurança
		Técnico	10	Risco de Qualidade

		Técnico	11	Risco de Sincronismo, não de aplica.
		OPORTUNIDADES		
		Técnico	5	Flexibilidade de dados na modernização
		Organizacional	12	Baixo Custo
		Organizacional	13	Solução relativamente rápida dependendo do conhecimento técnico da equipe.
	ASP	**RISCOS**		
		Técnico	1	Risco de performance. Dependência do relacionamento entre os dados do legado
		Técnico	2	Risco na manutenção
		Técnico	3	Baixa probabilidade de riscos relacionado ao conhecimento da equipe. Tecnologia muito utilizada
		Técnico	4	Tecnologia complexa
		Técnico	6	Risco de integridade, não se aplica.
		Técnico	7	Risco de indisponibilidade
		Técnico	8	Risco de implementação, porem facilidade de implementação, se comparado com CGI.
		Técnico	10	Risco de Qualidade
		Técnico	11	Risco de Sincronismo, não de aplica.
		OPORTUNIDADES		
		Técnico	5	Flexibilidade de dados
		Técnico	9	Alta Segurança
		Organizacional	12	Baixo Custo
		Organizacional	13	Solução relativamente rápida dependendo do conhecimento técnico da equipe.

4.1.4.3 Identificação de Riscos na Técnica de Encapsulamento Orientado a Objetos

Para obter sucesso na modernização dos sistemas legados usando orientação a objetos são necessários alguns passos:

- Separar a interfaces da regra de negócios
- Entender a lógica do negócio embutida no código legado
- Entender o relacionamento dos dados do sistema

- Finalmente agrupar e encapsular as funções criando os objetos.

Os riscos identificados nessa técnica estão descritos a seguir:

1 - Risco de Performance:

A performance nessa técnica é dependente da qualidade do código desenvolvido pelo programador. Portanto essa dependência é encarada como oportunidade, ou seja, o programador tem condições de otimizar a performance.

2 - Risco de Manutenção:

Essa técnica é bastante flexível não se devendo ter graves problemas com a manutenção nessa técnica. O impacto maior é o conhecimento da técnica.

3/4 - Risco de Conhecimento Técnico da Equipe e Complexidade da Tecnologia

Como mencionado na descrição do risco anterior, a probabilidade de risco relacionado com o conhecimento técnico da equipe e a complexidade da tecnologia é alta, portanto um dos pontos decisivos para escolha dessa técnica é saber previamente o conhecimento técnico da equipe, caso essa não tiver não se deve usar essa técnica, pois a tecnologia orientada a objetos é complexa e o conhecimento detalhado desta técnica não é trivial.

5 - Risco de Inflexibilidade da Aplicação:

Trata-se de uma técnica totalmente flexível, a principio pode-se modernizar as interfaces, os dados e a lógica sem restrições. Portanto nessa técnica não há dependência nenhuma do legado, tudo pode ser customizado.

6 - Risco de Integridade de dados:

Esse risco não se aplica para as técnicas de orientação a objetos.

7 - Risco de Indisponibilidade da aplicação:

Esse risco não se aplica para as técnicas de orientação a objetos.

8 - Risco de Implementação:

A grande maioria dos riscos identificados está relacionada com os riscos de implementação. Esses riscos estão descritos a seguir

Um dos riscos identificados é a dificuldade de separação entre as interfaces e as regras do negócio. Como é sabido, os sistemas legados são sistemas desenvolvidos de forma procedural geralmente implementado em linguagem COBOL. Essa implementação legada tende a ser menos modular do que os códigos modernos, e é muito provável que os aplicativos estejam inter-relacionados de uma forma não clara. Esse acoplamento pouco evidente e a programação não modular contribuem para as dificuldades de segregar as interfaces das unidades lógicas de negócios. Essa segregação se torna tão mais fácil se alguém da equipe do projeto de modernização conhecer em detalhes o funcionamento do sistema legado.

Outro risco está relacionado com a necessidade de entender como os **dados** do sistema legado se relacionam. Freqüentemente, o relacionamento entre os dados é feito pelos procedimentos e em sistemas complexos há uma grande quantidade de arquivos de programas e de dados que dificulta o entendimento completo do sistema. Deve-se considerar o risco no mapeamento desses relacionamentos de dados para entendimento do funcionamento do sistema. Segundo [Rechia et al, 2000], esse entendimento dos dados é necessário na modernização para transformar o modelo de dados desenvolvido de forma procedural em um modelo de dados orientado a objetos.

Outro risco identificado está relacionado com a dificuldade de mapear as funções lógicas do sistema. Há uma grande variedade de interfaces nos sistemas legados e as funções lógicas ficam "perdidas" no meio de tantas informações. Por exemplo, em aplicações on-line (IMS/CICS), os sistemas legados utilizam telas de menu para acesso à funcionalidade[4]. Para cada opção do menu tem-se um ou mais

[4] Essas funções são acionadas através das teclas de Funções: de <F1> até <F24>.

programas fontes envolvidos, ou seja, cada programa executa uma parte da lógica do negócio. Isso implica em um esforço adicional, ou seja, é necessário o entendimento de cada programa juntamente com o pedaço da lógica de negócio envolvida, e depois reconstituir as partes para obter a função lógica daquela operação.

Os sistemas legados são sistemas complexos, portanto possuem inúmeras relações entre modulo de programas e arquivos de dados seqüenciais, arquivos indexados (VSAM) e/ou tabelas relacionais (DB2). Esses relacionamentos são complexos e dificulta ainda mais a tarefa de modelar a função lógica do sistema.

Uma vez mapeadas e identificadas as funções lógicas do sistema, ainda há risco na transformação dessa estrutura monolítica procedural em uma estrutura rica hierárquica e estruturada em orientação a objetos. Segundo [De Lucia apud Comella-Dorda et al, 2000], o risco envolvido neste processo é a dificuldade encontrada da decomposição dos objetos partindo de uma estrutura procedural do sistema legado.

Identifica-se ainda o risco de anomalias na estrutura orientada a objetos herdadas da estrutura procedural. Em sistemas construídos de forma procedural pode existir um número elevado de inconsistências e/ou anomalias no código fonte. Por exemplo, um mesmo procedimento aparece em vários pedaços do código fonte, todos implementados de maneiras diferentes, porém com o mesmo resultado. Essas anomalias devem ser eliminadas uma vez que, em sistemas orientados a objetos os métodos estão associados a uma única classe, para tanto surge a necessidade de transformar o código correspondente ao procedimento anômalo em métodos.

O autor dessa monografia participou de um projeto de modernização de um sistema legado[5] para um sistema alvo orientado a objetos na instituição financeira em

[5] Sistema de Tributação desenvolvido em COBOL em ambiente mainframe. Parte desse sistema foi modernizado em baixa plataforma, disponibilizando informações via intranet.

que trabalha. Percebeu-se na prática que esse processo de mapeamento e tradução das regras de negócio é trabalhoso e de alto risco, porém, um bom conhecimento do sistema legado ameniza bem esses riscos.

9 - Risco de Segurança:

Esse risco não se aplica para as técnicas de orientação a objetos.

10 - Risco de Qualidade:

Pelo mesmo motivo relacionado na seção 4.1.4.2, existe uma certa probabilidade de risco relacionado com qualidade da aplicação.

11 - Risco de Sincronismo:

Esse risco não se aplica para as técnicas de orientação a objetos.

12/13 - Risco de Custo e Risco de Cronogramas

Essa tecnologia é complexa, envolve alta probabilidade de riscos na implementação, exige mão de obra especializada, conseqüentemente pode envolver um alto custo e demanda tempo de implementação. Esse custo deve-se ao esforço da equipe.

Os possíveis riscos e oportunidades identificados nesta técnica são mostrados na tabela a seguir.

Tabela 4.4 – Identificação de riscos da Técnica de modernização Funcional (Modelo Orientação a Objetos)

Modernização de	Técnica	Categoria	Ident.	Fatores de riscos potenciais
FUNCIONAL (LÓGICA)	Encapsulamento Orientado a Objetos	**RISCOS**		
		Técnico	1	Risco de performance, não se aplica.
		Técnico	2	Risco na manutenção
		Técnico	3	Alta probabilidade de riscos relacionado ao conhecimento da equipe. Tecnologia complexa
		Técnico	4	Tecnologia Complexa
		Técnico	6	Risco de integridade, não se aplica.
		Técnico	7	Risco de Indisponibilidade, não se aplica.
		Técnico	8	Riscos de implementação, tais como: • Entendimento do complexo relacionamento entre programa versus dados do sistema • Estrutura Orientada a Objetos é complexas, tudo deve ser representado como sendo objetos (seviços, dados, e lógica) • Dificuldade na decomposição de objetos partindo de uma estrutura procedural do sistema legado. • Dificuldade no mapeamento das funções lógicas do sistema legado • Entendimento das aplicações mainframes On-line, onde as funções lógicas estão mais segregadas. • Associar os procedimentos como sendo métodos das classes. Retirar as anomalias.
		Técnico	9	Risco de Segurança, não se aplica.
		Técnico	10	Risco de Qualidade
		Organizacional	11	Risco de Sincronismo, não se aplica.

		Técnico	12	Alto custo
		Técnico	13	Solução demorada
		OPORTUNIDADES		
		Técnico	5	Alta Flexibilidade

4.1.4.4 Identificação de Riscos na Técnica de Componentização

Como mencionado na seção 4.1.4, nessa técnica temos os mesmos riscos identificados na técnica de orientação a objetos acrescidos de outros possíveis riscos classificados também como sendo riscos de implementação. Todos os demais riscos identificados na técnica de orientação a objetos são aplicados nesta técnica. Portanto serão listados abaixo apenas os riscos identificados na técnica de componentização em adição aos riscos identificados na técnica de orientação a objetos.

8 - Risco de Implementação:

O primeiro risco dessa técnica está na escolha do modelo de componentes, segundo os padrões da organização e o objetivo da aplicação. Deve-se pesquisar quais os tipos de serviços são suportados em cada modelo de componentes e selecionar o modelo de componentes que melhor atender a necessidade da aplicação.

Para minimizar os possíveis riscos de implementação é importante que se considere na escolha do modelo de componentes o conhecimento técnico da equipe com relação aos modelos padrões de componentes. Por exemplo, uma equipe que está acostumada a trabalhar com componentes Java não terá o mesmo desempenho quando se trata de componentes da Microsoft.

Em adição aos riscos encontrados na técnica de orientação a objetos, a componentização se preocupa com a integração entre as estruturas de serviços dos componentes. De acordo com [De Lucia apud Comella-Dorda et al, 2000], a

necessidade para integrar estruturas de serviços é um risco relevante, pois pensando em aplicação distribuída, a maioria dos *middlewares* provê um conjunto de serviços tais como segurança, transação, persistência etc, entretanto, para obter algum serviço esperado pode ser necessária a integração de dois ou mais *middleware*. Essa integração é no mínimo problemática, segundo [Seacord apud Comella-Dorda et al, 2000], podendo haver incompatibilidade entre os produtos.

Segundo Seacord [Seacord et al, 2001], na técnica de modernização usando componentes é possível ter uma estratégia de modernização incremental, isto é, concentrar todo os esforços em uma determinada função ou módulo do sistema legado, entender, modernizar e implantar a modernização através de componentes. Uma vez implantando, repetir o procedimento de modernização de forma incremental para as outras funções do sistema. Essa característica pode ser encarada como sendo uma oportunidade, pois o cliente já vai recebendo partes operacionais do sistema modernizado durante o processo de modernização. E os riscos são mais controlados, pois todos os esforços estão concentrados apenas naquele módulo ou função que está sendo modernizado.

Neste caso as novas funcionalidades ou alterações são incrementalmente verificadas, e os riscos são mais gerenciados por estarem mais restritos.

Os possíveis riscos identificados nessa técnica estão relacionados na tabela abaixo.

Tabela 4.5 – Identificação de riscos da Técnica de modernização Funcional (Modelo de Componentes)

Modernização de	Técnica	Categoria	Ident.	Fatores de riscos potenciais
FUNCIONAL (LÓGICA)	Componentização	**RISCOS**		
		Técnico	1	IDEM A TÉCNICA DE ORIENTAÇÃO A OBJETOS
		Técnico	2	
		Técnico	3	
		Técnico	4	
		Técnico	6	
		Técnico	7	
		Técnico	8	Todos os Riscos de implementação da técnica de orientação a objeto mais: • Dificuldade em implementar os serviços, incompatibilidade de produtos. • Dificuldade de implementação dos conectores
		Técnico	9	IDEM A TÉCNICA DE ORIENTAÇÃO A OBJETOS
		Técnico	10	
		Organizacional	11	
		Técnico	12	
		Técnico	13	
		OPORTUNIDADES		
		Técnico	5	Alta Flexibilidade. Processo incremental de modernização. Componentes reusáveis (reuso).

4.1.4.5 Técnicas Procedurais

Uma outra técnica não comentada no relatório da SEI, porém muito utilizada na prática é a programação usando linguagem de baixa plataforma de modo procedural, ou seja, traduzir o código procedural dos sistemas mainframe para um

novo código também procedural, porém em linguagem de baixa plataforma (por exemplo Visual Basic, Delphi, etc.). Essas linguagens provêem soluções rápidas com interfaces modernas.

Os riscos identificados nessa técnica estão descritos a seguir:

1 - Risco de Performance:

O risco de performance que poderá apresentar nesta técnica diz respeito aos dados do legado. Conforme mencionado na seção 4.1.3, se os dados estiverem fortemente entrelaçados provavelmente haverá impacto de performance.

2 - Risco de Manutenção:

Os riscos de manutenção nesta técnica são mínimos, pela similaridade com o sistema legado.

3 - Risco de Conhecimento Técnico da Equipe:

A probabilidade de risco relacionado ao conhecimento técnico da equipe é muito baixa, visto que essa técnica consiste em usar ferramentas de 4ª. geração . Ou seja, a maneira de se implementar o código fonte é a mesma utilizada no sistema legado, geralmente desenvolvido em COBOL, portanto não há grandes problemas relacionados a esse risco.

4 - Risco de Complexidade da Tecnologia:

Essa técnica não envolve complexidade, pois as linguagens de 4ª geração são utilizadas como sendo linguagens procedurais como COBOL por exemplo.

5 - Risco de Inflexibilidade da Aplicação:

Essa técnica trata-se de copiar o código do legado para a aplicação nova, portanto tem-se uma alta probabilidade de inflexibilidade da aplicação, ou seja, a aplicação tem uma melhor apresentação devida os melhores recursos das ferramentas novas, porem é uma copia do legado.

6 - Risco de Integridade de dados:

Esse risco não se aplica, visto que não se fará mudanças nos dados da aplicação.

7 - Risco de Indisponibilidade da aplicação:

Esse risco não se aplica para essa técnica.

8 - Risco de Implementação:

Os riscos de implementação são mínimos visto que se trata de cópia do código fonte. Nesta técnica as probabilidades de riscos no mapeamento das funções lógicas são menores, visto que não será preciso a tradução do modelo procedural para outro modelo.

9 - Risco de Segurança:

Esse risco não se aplica para essa técnica.

10 - Risco de Qualidade:

O erros que ainda existem no sistema legado, em geral não serão corrigidos pois o principal objetivo dessa técnica é prover uma solução rápida de interfaces com algumas alterações na lógica da aplicação. Portanto tem-se uma certa probabilidade de se manter uma baixa qualidade da aplicação. E a introdução de erros novos?

11 - Risco de Sincronismo:

Esse risco não se aplica para essa técnica.

12 - Risco de Custo:

Essa técnica tem baixo custo. O custo aqui está atribuído ao esforço da equipe no processo de desenvolvimento ou tradução do código legado para a nova aplicação.

13 - Risco de Cronograma:

Essa técnica provê solução rápida, visto que se trata de cópia do código legado.

Tabela 4.6– Identificação de riscos de Técnicas Procedurais

Modernização De	Técnica	Categoria	Ident	Fatores potenciais de riscos
Lógica / Interfaces	Técnicas Procedimentais	**RISCOS**		
		Técnico	1	Risco de performance. Dependência do relacionamento entre os dados do legado
		Técnico	2	Risco na manutenção
		Técnico	3	Baixa probabilidade de risco relacionado ao conhecimento técnico da equipe.
		Técnico	5	Alto risco de flexibilidade da aplicação
		Técnico	6	Risco de integridade, não se aplica.
		Técnico	7	Risco de Indisponibilidade, não se aplica.
		Técnico	9	Risco de Segurança, não se aplica.
		Técnico	10	Alto risco relacionado a qualidade da aplicação.
		Técnico	11	Risco de Sincronismo, não de aplica.
		OPORTUNIDADES		
		Técnico	4	Tecnologia não complexa
		Técnico	8	Baixo risco de Implementação
		Organizacional	12	Baixo Custo
		Organizacional	13	Solução Rápida

4.2 Priorização Qualitativa dos Riscos Identificados

Foi criada uma tabela qualitativa dos dados identificados no processo anterior. Essa tabela tem como base uma escala sugerida por Pressman. É uma escala de probabilidades qualitativas que determina a probabilidade do risco. A tabela 4.6 demonstra essa escala.

Tabela 4.7 – Escala Probabilidades Qualitativas – adaptadas de [PRESSMAN, 1995]

Valor	Descrição
0	Altamente Improvável
1	Improvável
2	Moderado
3	Provável

O desenvolvimento dessa tabela qualitativa teve como base a experiência do autor, as entrevistas com coordenador com especialistas e pesquisas literárias referentes ao assunto, de acordo com a figura 4.1.

Com base na lista de riscos identificados no tópico anterior, foi criada a lista qualitativa dos riscos prevista como resultado da etapa de identificação de riscos do PMBOK, descrita na seção 3.3.

É importante mencionar que o processo de identificação dos riscos, assim como a priorização qualitativa e quantitativa desses riscos se estende a todos os sistemas legados de forma imparcial. Visto que durante a pesquisa em literatura e nas entrevistas com especialistas foram contemplados diversos sistemas legados com características bem específicas, espera-se que os produtos gerados nessa monografia sejam suficientemente abrangentes e imparciais para serem aplicados a qualquer sistema legado, não fazendo parte do escopo desta monografia a validação desta abrangência.

Tabela 4.8 - Priorização Qualitativa dos Riscos

	Screen Scraping	Gateways	XML	Replicação	CGI	ASP	OO	Componen-tização	Outras Téc. Proced.
1- Risco de Performance da aplicação	Moderado	Moderado	Moderado	Improvável	Provável	Moderado	Improvável	Improvável	Moderado
2- Risco na manutenção	Provável	Improvável	moderado	Provável	Provável	Moderado	Moderado	Moderado	Improvável
3- Risco do Conhecimento Técnico da Equipe	Improvável	Improvável	Provável	Provável	Provável	Moderado	Provável	Provável	Improvável
4- Risco da complexidade da Tecnologia	Improvável	Improvável	Provável	Provável	Provável	Provável	Provável	Provável	Improvável
5- Risco de Inflexibilidade na modernização	Provável	Moderado	Improvável	Provável	Moderado	Moderado	Improvável	Improvável	Provável
6- Risco na Integridade de dados	Improvável	Moderado	Moderado	Provável	Improvável	Improvável	Improvável	Improvável	Improvável
7- Risco de Indisponibilidade da aplicação	Moderado	Improvável	Moderado	Improvável	Moderado	Moderado	Improvável	Improvável	Improvável
8- Risco na Implementação	Improvável	Improvável	Moderado	Provável	Provável	Provável	Provável	Provável	Moderado
9- Risco de Segurança da Aplicação	Improvável	Improvável	Improvável	Moderado	Provável	Improvável	Improvável	Improvável	Improvável
10- Risco de Qualidade	Provável	Improvável	Improvável	Improvável	Moderado	Moderado	Moderado	Moderado	Provável
11- Risco de Sincronização da aplicação	Provável	Improvável	Improvável	Provável	Improvável	Improvável	Improvável	Improvável	Improvável
12- Risco de Custo	Improvável	Improvável	Moderado	Provável	Moderado	Moderado	Provável	Provável	Improvável
13- Risco de cronograma	Improvável	Improvável	Moderado	Moderado	Moderado	Moderado	Provável	Provável	Improvável

4.3 Comparação Quantitativa dos Riscos segundo as Técnicas de Modernização

A priorização quantitativa dos riscos envolve dois parâmetros. Primeiro a probabilidade de ocorrer o risco, e segundo o impacto do risco caso ele se verifique.

Para o primeiro parâmetro, que se refere à probabilidade de ocorrer o risco utilizou-se a parte numérica da escala sugerida por Pressman (tabela 4.6).

O segundo parâmetro deverá ser avaliado pelo gerente do projeto, que deve saber o impacto de cada risco no seu sistema, na sua equipe e em sua empresa. Utilizou-se a variável I para o impacto na equação abaixo. Estabelecer claramente os limites numéricos para a variável de impacto. A equação de risco é dada por:

Risco = Probabilidade * Impacto

Por exemplo, em modernização de interfaces uma equipe que tem conhecimento em ASP terá menos impacto dos riscos na utilização do ASP, que é uma tecnologia complexa, do que na utilização da técnica de *screen scraping*.

A comparação é feita dentro de cada categoria estudada anteriormente conforme seção 2.5, multiplicando-se o índice de probabilidade sugerido pelo autor pelo fator de impacto avaliado pelo gerente.

Nessa comparação quanto maior for o número relacionado a um determinado risco, maior é a probabilidade de ocorrência desse risco.

Os impactos podem ser baseados segundo a tabela abaixo:

Tabela 4.9 Tabela Sugerida de Impacto

Valor	Impacto
0.0	Mínimo
1.0	Menor
2.0	Razoável

3.0	Crítico
4.0	Catastrófico

4.3.2 Comparação Quantitativa entre as Técnicas de Interfaces

Entre as alternativas de técnicas de interfaces, podem ser comparadas a técnica *screen scraping* e a técnica de CGI / ASP. Como visto no capitulo 2, essa técnica de CGI / ASP também lida com a modernização de interfaces de uma forma mais abrangente, ou seja, podem ser feitas algumas otimizações na lógica da aplicação, porém o objetivo principal é a modernização das interfaces.

Observe-se que para técnica de *screen scraping* o risco foi analisado sob os dois enfoques, de aquisição e de desenvolvimento da ferramenta *scraping*.

De acordo com os critérios mencionados tem-se a tabela 4.9. Nota-se que o primeiro parâmetro é o número que indica a probabilidade do risco segundo a tabela 4.6 sugerida por Pressman. O segundo parâmetro é a variável I que indica o impacto de cada risco no projeto.

Para a obtenção final da quantificação dos riscos basta associar um número (de acordo com alguma tabela predefinida) para cada impacto e fazer a multiplicação da probabilidade e o impacto.

Tabela 4.10 – Comparação quantitativa das Técnicas de Interfaces

Riscos	Descrição dos Riscos	Screen Scraping (aquisição)	Screen Scraping (desenvolvimento)	CGI	ASP
Técnico	**Risco do conhecimento técnico da equipe de desenvolvimento**	0 * I	2 * I	3 * I	2 * I
	Risco de implementação	1 * I	3 * I	3 * I	3 * I
	Risco de Qualidade da aplicação	3 * I	3 * I	2 * I	2 * I
	Risco em Performance da aplicação	2 * I	2 * I	3 * I	2 * I

	Risco de Flexibilidade da aplicação	3 * I	3 * I	2 * I	2 * I
Organizacional	**Risco com cronograma (Tempo para implantar)**	1 * I	2 * I	3 * I	2 * I
	Risco relacionado a Custos (Recurso Financeiro)	1 * I	2 * I	2 * I	2 * I

4.3.2.1 Interpretação da tabela quantitativa das Técnicas de Interfaces

Observe-se que não é possível concluir por uma técnica apenas com o primeiro parâmetro (probabilidade) da composição dos riscos, avaliado pelo autor. Dependendo da distribuição do segundo parâmetro na tabela acima, a ser feita pelo gerente, toda essa análise pode não ser mais verdadeira. Por exemplo, nota-se que, a técnica de modernização de interfaces com maior probabilidade de riscos é a CGI. Entretanto em uma determinada equipe ou projeto os impactos dessa técnica são bem menores que nas outras, portanto o resultado final será bem diferente com a aplicação do segundo parâmetro.

Do ponto de vista de probabilidade de riscos percebe-se que a técnica *screen scraping* no primeiro caso (aquisição), é a que menos oferece probabilidade de riscos técnicos na modernização das interfaces, e também é a técnica de menor custo e que provê uma solução mais rápida, porém deve tomar alguns cuidados, como por exemplo:

- Embora a técnica de *screen scraping* seja a menos susceptível a erros, essa técnica oferece uma alta probabilidade de risco de qualidade, como mencionado anteriormente, essa técnica não corrige os erros da tela do mainframe, portanto os erros são herdados de uma tela para a outra.
- A técnica de *screen scraping* também não oferece flexibilidade na modernização, ou seja, a nova tela deve estar "casada" com a tela

antiga, não oferecendo nenhuma customizações ou novas funcionalidades.

- Nota-se também que o desenvolvimento de uma ferramenta *screen scraping* oferece mais probabilidade de riscos do que a aquisição da mesma. Pois, oferece alta probabilidade de risco na implementação devido ao complexo mecanismo de sincronismo entre as aplicações mainframes e a tela do usuário, enquanto que na aquisição, esse mecanismo já foi implementado. No entanto a ferramenta pode não estar disponível para aquisição!

É importante observar que as técnicas usando CGI / ASP oferecem menor probabilidades de risco quanto ao quesito qualidade e flexibilidade da aplicação, pois essas aplicações normalmente se comunicam diretamente com os dados ou lógica da aplicação, e não existe nenhuma amarração com a tela mainframe. Portanto os dados e telas podem ser customizados de acordo com a necessidade dos usuários.

O diferencial do ASP se comparado com CGI é que as pessoas têm mais familiaridade com o ASP por sua forte presença no mercado e por oferece um ambiente mais confortável de desenvolvimento. Em ASP é usual reutilizar componentes tornando o desenvolvimento mais rápido, portanto essa técnica tem menor probabilidade de risco com cronograma.

Em CGI o desenvolvimento normalmente é feito usando linguagem C e Perl (linguagem de script), essas linguagem são mais difícil e o desenvolvimento é lento.

4.3.3 Comparação quantitativa das técnicas de Dados

As técnicas de modernização de dados são: *Gateways*, Replicação de Dados e XML.

Os critérios usados foram os mesmos da comparação da técnica anterior, porém a única diferença é que o critério de qualidade foi trocado pelo critério de integridade dos dados.

De acordo com os critérios mencionados tem-se a tabela 4.10. Como descrito na seção 4.3.2, o primeiro parâmetro é o número que indica a probabilidade do risco segundo a tabela 4.6 sugerida por Pressman. O segundo parâmetro é a variável I que indica o impacto de cada risco no projeto.

A mesma regra para obtenção final da quantificação dos riscos da tabela anterior é válida nesta técnica. Basta associar um número (de acordo com alguma tabela predefinida) para cada impacto e fazer a multiplicação da probabilidade e o impacto.

Tabela 4.11 – Comparação quantitativa das Técnicas de Dados

Riscos	Descrição dos Riscos	Gateways	XML	Replicação de dados
Técnico	**Risco do conhecimento técnico da equipe de desenvolvimento**	1 * I	3 * I	3 * I
	Risco de implementação	2 * I	3 * I	3 * I
	Risco de Integridade de dados	1 * I	2 * I	3 * I
	Risco de Performance da aplicação	2 * I	2 * I	1 * I
	Risco Flexibilidade da aplicação	2 * I	1 * I	3 * I

Organizacional	**Risco com cronograma (Tempo para Implantar)**	1 * I	2 * I	2 * I
	Risco relacionado a Custos (Recurso Financeiro)	1 * I	2 * I	3 * I

4.3.3.1 Interpretação da tabela de comparação quantitativa das Técnicas de Dados

Analisando a tabela acima, apenas levando em consideração o parâmetro de probabilidade do risco, conclui-se que a técnica que menos oferece riscos técnicos na modernização de dados é a técnica que utiliza *gateways*, porém deve usar alguns critérios para a escolha da técnica.

Se o critério for custo, a técnica que oferece menor custo é a técnica que utiliza *gateways*, e a de maior custo trata-se da técnica de replicação de dados, por envolver um mecanismo complexo de replicação dos dados e naturalmente uma outra base de dados.

Os critérios de escolha devem estar claramente definidos para a seleção da estratégia de modernização. Por exemplo, no desenvolvimento do Sistema de Pagamento Brasileiro (SPB), o critério adotado para integração dos dados não foi custo nem tempo para implementar, o critério escolhido foi flexibilidade. Portanto optou-se pelo padrão internacional XML que suporta aplicações B2B, ou seja, pela sua flexibilidade na modernização e integração de sistemas.

Outro exemplo, na instituição financeira onde o autor trabalha foi implementado um projeto de modernização e integração de uma determinada aplicação mainframe, com uma aplicação cliente, de baixa plataforma voltada para usuários internos de fundos de investimentos. Essa aplicação tinha por premissa ser uma aplicação com alta performance. Portanto depois de analisar a aplicação, e analisar como os dados eram acessados ou atualizados, chegou-se à conclusão que a melhor solução seria a criação de modelo lógico simplificado de dados (VIEW) e acessar esses dados usando ODBC.

Analisando-se a tabela 4.10 para o exemplo citado, tendo como critério a alta performance, porém sem considerar o parâmetro de impacto, com certeza a opção selecionada seria a técnica de replicação de dados, porém , nesse caso, a aplicação oferecia alto risco de integridade dos dados, por se tratar de sistemas extremamente dinâmicos, com alta probabilidade de em um determinado instante os dados estarem desatualizados nas réplicas. Por essa forte razão não se optou por replicação de dados.

Portanto, a tabela 4.11 trata de uma referência rápida para priorização quantitativa dos riscos de modernização dos dados, porém deve ser avaliadas cuidadosamente as aplicações, considerando suas particularidades e também deve-se olhar para os efeitos colaterais de cada técnica. Por exemplo, a técnica de replicação de dados é a que menos oferece probabilidade de risco de performance, mas pode oferecer alto risco de integridade dos dados.

4.3.4 Comparação quantitativa das técnicas de modernização Funcional (Lógica)

As técnicas de modernização funcionais são: Modelo Orientado a Objetos, Modelo de componentes e técnicas procedurais.

Os riscos usados basicamente foram os mesmos das comparações anteriores, porém com algumas diferenças.

O risco de complexidade da tecnologia nesta técnica passou a ser mais importante que o risco de performance. Ambos os enfoques, orientação a Objetos e componentização são complexos, porém o modelo de componentes é considerado mais complexo.

De acordo com os critérios mencionados tem-se a tabela 4.11. Como descrito na seção 4.3.2, o primeiro parâmetro é o número que indica a probabilidade do risco

segundo a tabela 4.6 sugerida por Pressman. O segundo parâmetro é a variável I que indica o impacto de cada risco no projeto.

A mesma regra para obtenção final da quantificação dos riscos da tabela anterior é válida nesta técnica. Basta associar um número (de acordo com tabela 4.9) para cada impacto e fazer a multiplicação da probabilidade e o impacto.

4.12 – Comparação quantitativa das Técnicas Funcionais

Riscos	Descrição dos Riscos	Orientação a Objetos	Componentização	Outras Técnicas
Técnico	**Risco do conhecimento técnico da equipe de desenvolvimento**	2 * I	3 * I	2 * I
	Risco de implementação	3 * I	3 * I	2 * I
	Risco da Complexidade da tecnologia	3 * I	3 * I	2 * I
	Risco de Qualidade da Aplicação	2 * I	2 * I	3 * I
	Risco de Flexibilidade da aplicação	1 * I	1 * I	2 * I
Organizacional	**Risco com cronograma (Tempo para Implantar)**	3 * I	3 * I	2 * I
	Recurso Financeiro (Risco relacionado a Custos)	3 * I	3 * I	2 * I

4.3.4.1 Interpretação a tabela de comparação qualitativa das Técnicas Funcionais

Observa-se que a análise dessa tabela diz respeito apenas ao primeiro parâmetro (probabilidade) da composição dos riscos. As técnicas de modernização da lógica de negócio têm os maiores índices de risco, tanto técnico como

organizacional (prazo e custo). Em análise da tabela acima, conclui-se que a técnica de componentização por ser uma tecnologia complexa e possuir uma alta probabilidade de desconhecimento da técnica pela equipe de desenvolvimento, oferece o maior risco técnico e organizacional na modernização de sistemas legados.

A técnica de modernização funcional com a menor probabilidade de risco é a técnica descrita como sendo "técnicas procedurais", ou seja, a lógica de negócio do mainframe é modernizada usando uma linguagem de 4ª. geração (por exemplo, Visual Basic, Delphi, etc) de uma forma procedural, isto é, sem a utilização de modelos orientados a objetos e/ou modelo padrão de componentes.

Neste caso a tradução do código mainframe para o código novo é mais natural sem a necessidade de conversão do modelo procedural para outros modelos. Conseqüentemente o tempo para implementação essa solução é menor. Embora essa técnica oferece a menor probabilidade de riscos, é importante verificar que no quesito de flexibilidade essa técnica é deficiente, pois é extremamente dependente da seqüência dos procedimentos lógicos do mainframe, ficando de uma certa forma amarrada com a solução legada.

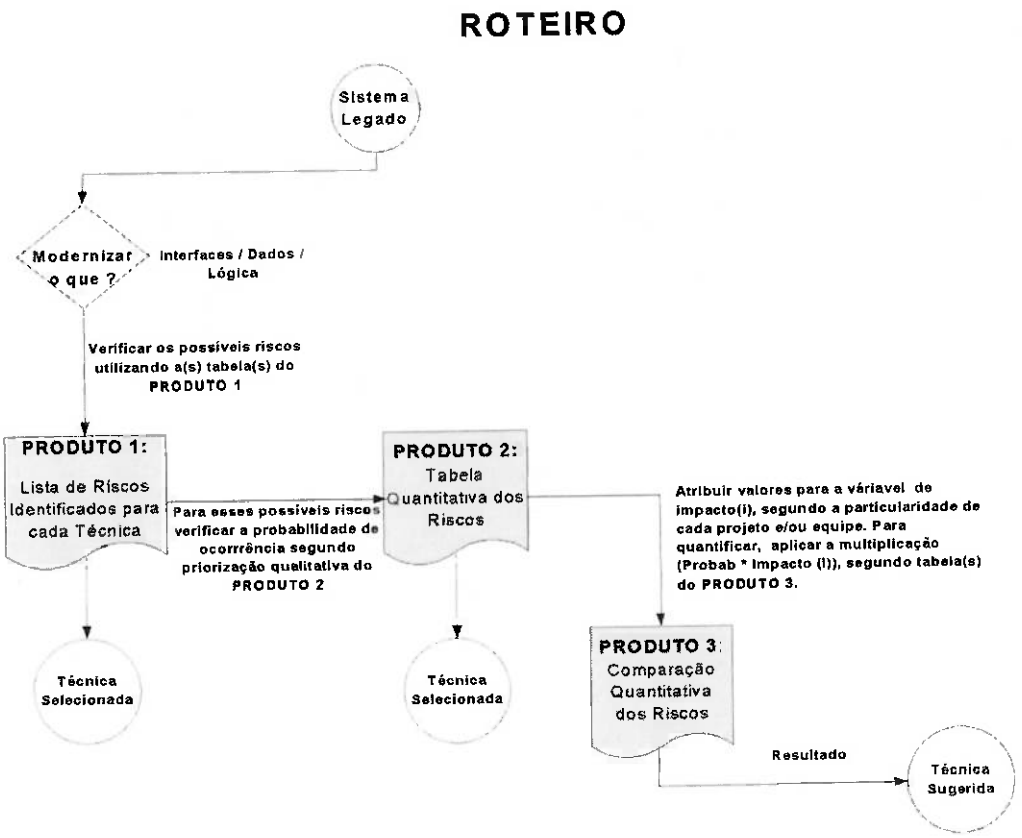
No critério qualidade, a técnica que oferece maior risco é "Técnicas procedurais", pois como a idéia é aproveitar ao máximo a estrutura procedural do sistema legado é bem provável que alguns erros existentes no código legado nem serão questionados. Diferentemente das técnicas orientadas a objetos, para fazer a tradução de um modelo para o outro é necessário conhecer detalhadamente as funções.

Embora não descrito na tabela acima, um outro fator que contribui para a complexidade da técnica de modernização usando componentes é a dificuldade da integração de serviço entre os componentes.

Neste capítulo, foi aplicada a gerência de riscos com foco nas técnicas de modernização do legado. Obteve-se no final de cada processo um produto que irá compor o roteiro proposto, que é o núcleo da contribuição dessa monografia. A utilização desse roteiro auxilia o responsável pelo projeto de modernização selecionar uma das técnicas segundo alguns critérios exposto neste capítulo.

A figura abaixo resume o roteiro proposto.

Figura 4.2 – Roteiro e Utilização



Obs: É possível o responsável pelo projeto de modernização selecionar uma das técnicas apenas com a verificação do PRODUTO 1 e/ou do PRODUTO 2. Isso depende da particularidade da aplicação que se deseja modernizar.

CAPÍTULO 5

CONCLUSÃO

Este capítulo apresenta a conclusão desta pesquisa e aponta algumas possibilidades de trabalho a serem conduzidos futuramente com o objetivo de ampliar o escopo desta monografia.

Essa monografia foi desenvolvida com o objetivo de auxiliar os gerentes de software na seleção de uma estratégia de modernização dos sistemas legados. As técnicas de modernização apresentadas foram baseadas no relatório técnico da SEI publicado em 2000 com o título "A survey of Legacy System Modernization Approaches". Adicionalmente a essas técnicas, o autor dessa monografia buscou outras técnicas usadas na prática que não foram abordadas no relatório da SEI. Essas outras técnicas foram baseadas na experiência técnica do autor, por trabalhar com sistemas mainframe em uma grande instituição financeira do país, e pelo processo de entrevistas com a equipe técnica responsáveis pela maioria dos projetos de modernização de sistemas legados, dentro dessa instituição financeira.

5.1 Resultados Alcançados

Os produtos dessa monografia foram:

- Lista de riscos identificados em cada técnica de modernização
- Tabela contendo os riscos priorizados qualitativamente em cada técnica
- Comparação quantitativa de risco entre as categorias de técnicas de modernização juntamente com a explicação de como aplicar essa quantificação na prática.

Esse conjunto de produtos foi chamado de Roteiro para Apoio à Seleção de Modernização dos Sistemas Legados.

Espera-se que com esse roteiro o gerente de projeto ou a pessoa responsável pela modernização do sistema legado possa balizar melhor sua escolha, devido ao suporte dado pelos produtos gerados nessa monografia. Esse roteiro é abrangente e se aplica a qualquer sistema legado, desde que os critérios que se deseja avaliar sejam coincidentes com os critérios do roteiro.

As tabelas de priorização quantitativas são tabelas contendo a fórmula para determinar a prioridade quantitativa dos riscos. Essa fórmula envolve dois elementos; a probabilidade, que foi estabelecida nesta monografia e o impacto (I). Esse último assumirá valores específicos que são dependentes da particularidade de cada equipe e/ou projeto de modernização.

5.2 Trabalhos Futuros

O processo de gerência de riscos empregado nesta monografia limitou-se às fases de Identificação dos Riscos, Analise Qualitativa e Quantitativa desses riscos, tendo como base o processo de Gerência de Risco do PMBOK. Visando ampliar o escopo dessa dissertação é interessante desenvolver os processos restantes da Gerência de Risco do PMBOK, ou seja, desenvolver o processo de planejamento de respostas a riscos, que implica em um maior detalhamento dos riscos bem como o Controle e a monitoração dos riscos.

Sendo assim, no futuro pode-se ter um processo completo de gerência de risco baseado no PMBOK com foco nas técnicas de modernização de sistemas legados. Ainda como próximo passo, é importante que, se desenvolva um estudo de caso com o objetivo de validação dos produtos gerados nesta monografia.

REFERÊNCIA BIBLIOGRÁFICA

[Attachmate, 2000]

Attachmate Coporation Outubro de 2000. **Mudando o Objetivo de Aplicativos Legados para Web: Avaliação do Acesso Baseado em Tela.** Disponível em: <http://br.attachmate.com/article/0,1012,3485_11_5554,00.html. Acessado em 15 outubro de 2003.

[Belloquim, 1999]

BELLOQUIM, Á. **A baixa produtividade e o custo de Desenvolvimento.** Revista Developers, n. 32, ano 3, p.62, Maio.

[Bennett, 1995]

BENNETT, K.H. **Legacy Systems: Coping With Success**, IEEE Software, January 1995, Vol 12, No. 1, pp 19-23.

[Bernstein, 1997]

BERSTEIN, Peter. **Desafio aos deuses:** a fascinante história do risco. Rio de Janeiro: Campus, 1997.

[Bergey et al, 2001]

BERGEY; JOHN; O'BRIEN, LIAM; SMITH DENNIS. **Options Analysis for Reengineering (OAR): A Method for Mining Legacy assets**. CMU/SEI-2001-TN-01, Carnegie Mellon University, EUA, June 2001

[Bragança, 1999]

BRAGANÇA, A. **Introdução ao ADO / OLE DB**
Engenharia de Informática Instituto Superior de Engenharia do

Porto, Portugal, 1999/2000 Disponível em:
<www.dei.isep.ipp.pt/~alex/publico/ bd2/bd2_ADO_OLEDB.pdf>
Acessado em 10 de Outubro de 2003.

[Brodie, 1995]

BRODIE, M.L; STONEBRAKER, M. ***Migrating Legacy Systems: Gateways, interfaces, and the incremental approach*** Morgan Kaufmann Publishers, Inc.
San Francisco, California, 1995.

[Cagnin, 1999]

CAGNIN, MARIA ISTELA. **Avaliação das vantagens quanto à facilidade e Expansão de sistemas legados à engenharia reversa e Segmentação.** UfSCAR- Universidade Federal de São Carlos. Tese, 1999.

[Cerqueira et al., 2000]

CERQUEIRA, Alisson Gomes; SILVA, Gláucia Braga; Cardoso, Olinda Nogueira P. **Uma Visão Geral de XML em Recuperação de Informação.** DCC - Departamento de Ciência da Computação
UFLA -Universidade Federal de Lavras - Minas Gerais. Artigo.

[Comella-Dorda et al, 2000]

COMELLA-DORDA, S. Et al. **A Survey of Legacy System Modernization Approaches** CMU/SEI-2000-TN-003, SEI,
Carnegie Mellon University, EUA, April 2000.

[CORBA]

Common Object Request Broker Architecture
Disponível em: <http://www.corba.org.br/corba.htm>.
Acessado em 11 de Novembro de 2003.

[Deitel, 2001]

Deitel, H.M; Deitel, P.J. **JAVA Como Programar**
Editora Bookman, 3.edição. Porto Alegre 2001.

[FOLDOC, 1996]

The Free On-line Dictionary of Computing
Editor Denis Howoe
Disponível em : < http://wombat.doc.ic.ac.uk >.
Acessado em 03 de Outubro de 2003.

[Fontanette et al, sd]

Fontanette, Valdirene; et al. **Reprojeto de Sistemas Legados Baseado em componentes de Software.** UFCAR –Universidade Federal de São Carlos . Artigo

[Homer et al, 2000]

HOMER, Alex et al. **Professional Active Server Pages 3.0**
Rio de Janeiro Editora Ciência Moderna Ltda,. 2000.

[JDBC]

JDBC Technology – The source for Java Developers
Disponível em: <http:// Java.sun.com/products/jdbc>.
Acessado em 05 de Novembro de 2003.

[Lister, 1997]

LISTER, Tim. ***Risk Management Is Project Management for Adults***
IEEE Software, Vol 14, N.3, p. 51-59, May/June, 1997.

[Machado, 2002]

Machado, Cristina Angela Filipak. **A-Risk: Um método para Identificar e Quantificar Riscos de Prazo em Projeto de Desenvolvimento de Software.**
PUCPR - Pontifícia Universidade Católica do Paraná. Curitiba, 2002. Tese

[OMG]

Object Management Group
Disponível em: <http://www.omg.org>.
Acessado em 10 Outubro de 2003.

[Pinheiro 1998]

PINHEIRO, P. **Um Estudo sobre a Migração de Sistemas Legados Centralizados para Ambientes Distribuídos** - Universidade Federal de Minas Gerais – UFMG, Tese 1996.

[PMI, 2000]

Project Management Institute – PMI. **A guide to the project Management body of Knowledge.**
PMI Publishing Division, 2000.Disponível em : <www.pmi.org>.

[Recchia et al, 2002]

RECCHIA,Edson Luiz; PENTEADO, Rosangela. **FaPRE/OO: Uma Familia de Padrões para Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais.** Artigo apresentado no SugarloafPLoP2002 – The Second Latin American Conference on Pattern Languages of Programming Agosto/2002, Itaipava-RJ.

[Seacord et al, 2001]

SEACORD, R. Et al. **Legacy System Modernization Strategies** CMU/SEI-2001-TN-025, SEI, Carnegie Mellon University, EUA, April 2000.

[SPB]

O SPB no Contexto da Reestruturação do Sistema Financeiro Nacional. Disponível em: <http://www.bcb.gov.br>. Acessado em 21 de Novembro de 2003

[SUN 2003]

Enterprise JavaBeans Technology
Disponível em: <http:// Java.sun.com/products/ejb>.
Acessado em 10 de Novembro de 2003.

[SUN_CONN]

J2EE Connector Architecture
Disponível em: <www.jcp.org/en/jsr/detail?id=16>.
Acessado em 21 de Novembro de 2003.

[Umar, 1997]

UMAR, AMJAD. **Object-Oriented Cliente/Server Internet Environments.**
Editora Prentice-Hall PTR, New Jersey 1997.

[Turban, 1999]

TURBAN,E.et al. **Eletronic Commerce A Managerial Perspective.**
Editora Prentice-Hall , New Jersey 1999.

[Tittel, 1998]

TITTEL, Ed.; James, S.N. **HTML 4 for Dummies**
Editora IDG Books Worldwide, Foster City 1998.

[Weiderman et al, 1997a]

WEIDERMAN, N.H. et al. **Approaches to Legacy System Evolution** CMU/SEI-1997-TN-014, SEI, Carnegie Mellon University, EUA, December, 1997.

[Weiderman et al, 1997b]

WEIDERMAN, N.H. et al. **Implications of Distributed Object Tecnology for Reengineering** - CMU/SEI-1997-TN-005, SEI, Carnegie Mellon University, EUA, June, 1997.

[XML]

Extensible Markup Language (XML)

Disponível em: <http://www.w3.org/XML/>.

Acessado em 06 de Novembro de 2003.