PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

COORDENADORIA GERAL DE EXTENSÃO, APERFEIÇOAMENTO E ESPECIALIZAÇÃO

ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

MODERNIZAÇÃO DE SISTEMAS LEGADOS FAZENDO USO DE UM PROCESSO DE REENGENHARIA

Rafael Marcelino

São Paulo

2013

Rafael Marcelino

MODERNIZAÇÃO DE SISTEMAS LEGADOS FAZENDO USO DE UM PROCESSO DE REENGENHARIA

Monografia apresentada como requisito parcial
para a obtenção do título de Especialista em
Engenharia de Software da PUC-SP

Prof. Dr. Daniel Couto Gatti
Orientador

Prof. Dr. Carlos Eduardo de Barros Paes
Coordenador

São Paulo
2013

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

RESUMO

Com o passar do tempo os softwares que suportam os processos de negócios das empresas vão sofrendo modificações nos seus requisitos originais conforme as necessidades do negócio mudam. Essas mudanças acabam comprometendo a qualidade do sistema e tornando-o mais difícil de manter, até o ponto em que eles começam a ter dificuldade em atender o negócio. Quando um sistema atinge este estágio deve ser feita uma avaliação a fim de identificar os candidatos à modernização e chegar à melhor estratégia aplicável naquele momento, ou seja, se é mais viável continuar dando manutenção, substituir ou modernizar o sistema. Chegando à conclusão de que a melhor estratégia para o sistema é modernizá-lo, deve ser feito um estudo detalhado a fim de obter o entendimento e definir a arquitetura-alvo, assim como a estratégia de modernização aplicável a ele e validar se ela é viável ou não. Este processo visa minimizar os riscos do projeto de modernização pois inclui uma avaliação do legado, da tecnologia utilizada, da tecnologia alvo e da solução proposta antes de partir para o projeto de modernização. O processo para o projeto de modernização consiste em aplicar os conceitos de engenharia reversa e engenharia para a frente dentro de um processo de reengenharia em espiral que visa entregar novas partes do sistema modernizado a cada iteração, onde o legado e o sistema modernizado funcionam em paralelo ao mesmo tempo em que os dados vão sendo gradativamente migrados com a utilização de pontes para convertê-los ao novo formato, até que todo o legado tenha sido substituído.

Palavras-chave: Sistemas legado, reengenharia, engenharia reversa, modernização, avaliação, riscos

ABSTRACT

Over time the software that support the business processes of the companies will suffer modifications in their original requirements as the business needs change. These changes end up compromising the quality of the system and making it more difficult to maintain, to the extent that they begin to have difficulty meeting the business. When a system reaches this stage, an assessment must be made in order to identify candidates for modernization and reach the best strategy applicable at that time, i.e. whether it is more feasible to continue giving maintenance, replace or modernize the system. Coming to the conclusion that the best strategy for the system is modernize it, a detailed study must be done in order to obtain the understanding and define the target architecture, as well as the modernization strategy applicable to it and validate if it is feasible or not. This process aims to minimize the risks of the modernization project since it includes an assessment of the legacy system, of the technology used, the target technology and the proposed solution before start the modernization project. The process for the modernization project is to apply the concepts of reverse engineering and forward engineering within a spiral process of reengineering that aims to deliver new parts of the modernized system to each iteration, where the legacy and modernized system run in parallel at the same time that the data are being gradually migrated with the use of bridges to convert them to the new format, until all of the legacy has been replaced.

Keywords: Legacy systems, reengineering, reverse engineering, modernization, assessment, risks

1 INTRODUÇÃO

Nos dias atuais grande parte das corporações tem gasto uma porção importante de seus recursos para software com tarefas de correção de erros ao invés de novas implementações. Para que as organizações se mantenham cada vez mais competitivas seus sistemas devem suportar mudanças em um tempo que atenda às necessidades dos clientes e do negócio. Sendo assim, faz-se necessário a adoção de cultura de melhoria contínua para a constante evolução de sistemas que podem estar fazendo com que a empresa fique presa em processos obsoletos, impactando negativamente na dinâmica que o mundo de negócios atual necessita.

Essas mudanças nos processos de negócio das empresas acabam por modificar os requisitos dos sistemas que os suportam, logo esta cultura de melhoria contínua deve visar manter a qualidade técnica desses sistemas para que a sua manutenção e constante evolução ocorram a um custo aceitável ao mesmo tempo em que a flexibilidade para suportar novas mudanças deve ser mantida.

Com o passar do tempo esses sistemas acabam por carregar não somente os requisitos iniciais que contemplam o propósito na qual ele foi desenvolvido, mas também todas as solicitações de mudanças que acabaram por modificá-lo para que ele passasse a atender novos requisitos. Além disso, com o tempo a tecnologia utilizada e os processos na qual ele foi desenvolvido podem se tornar obsoletos, tornando a sua manutenção cara e custosa. Esses sistemas são o que chamamos de Sistemas Legados.

Considerando as dificuldades que as empresas tem com os seus sistemas legados devemos adotar uma estratégia que nos permita modernizá-los a um custo aceitável, ao mesmo tempo em que visamos conseguir isto minimizando os riscos. Para isto é necessário a adoção de uma estratégia pragmática para o levantamento do portifolio de sistemas legados da empresa, iniciando com uma avaliação da qualidade técnica e do valor para o negócio desses sistemas a fim de se chegar a uma lista de prioridades dos sistemas candidatos à modernização.

Com uma lista dos sistemas candidatos à modernização em mãos devemos fazer uma análise de qual a estratégia de modernização mais viáveis a cada um deles, dado o estágio do seu ciclo de vida, qualidade técnica e valor para o negócio a fim de identificarmos qual a melhor estratégia para a modernização desses sistemas: continuar mantendo o sistema, modernizar, substituir por um novo ou mesmo descontinuá-lo.

O presente trabalho tem por objetivo analisar como esses sistemas podem ser modernizados de uma maneira viável através das técnicas de engenharia reversa, reengenharia e engenharia para a frente, assim como apresentar uma análise de como esse portifolio de sistemas legados pode ser levantado e como eles podem ser avaliados a fim de se estudar a viabilidade de um projeto de modernização que minimize os riscos e seja economicamente acessível para a organização.

São apresentadas algumas técnicas, abordagens, metodologias e processos existentes para a avaliação de sistemas legados, avaliação da viabilidade de um projeto de modernização em cada um deles e estratégias de modernização que podem ser adotadas no sistema dependendo do estágio atual em seu ciclo de vida.

No capítulo 2, Fundamentos e Estado da Arte, o trabalho começa com uma revisão bibliográfica apresentando o que os autores falam sobre sistemas legados, a natureza do software, sobre o ciclo de vida e modernização de sistemas legados.

No capítulo 3, Evolução do Legado, são apresentados os conceitos de manutenção de software dentro do ciclo de vida do legado, assim como as abordagens que podem ser adotadas para modernizar esses sistemas dado o seu ciclo de vida como continuar mantendo o sistema, modernizá-lo ou substituir o sistema. É apresentado como o portifolio de sistemas legados da empresa pode ser avaliado e priorizado, assim como uma abordagem para a análise da viabilidade de modernização desses sistemas que visa minimizar os riscos e como eles podem ser mitigados e classificados.

No capítulo 4, Técnicas de Modernização, analisamos as técnicas que podem ser utilizadas para a modernização desses sistemas abordando a Engenharia Reversa, Reengenharia e a Engenharia para a Frente, assim como um processo de reengenharia que pode ser adotado para modernizar um sistema. São analisados os tipos de modernização que podem ser adotados como a modernização da interface com o usuário, dos dados ou uma modernização lógica do sistema, e apresentado como o sistema modernizado pode ser entregue funcionando em paralelo com o legado, de uma maneira que permita que dados sejam migrados gradualmente para o novo sistema durante este processo.

2 FUNDAMENTOS E ESTADO DA ARTE

2.1 A NATUREZA DO SOFTWARE

Uma das principais causas das dificuldades que as empresas enfrentam com seus softwares se dá devido a sua própria natureza, pois de acordo com (SOMMERVILLE, 2011), sistemas de software são abstratos e intangíveis. Eles não estão limitados pelas propriedades de materiais, governados por leis físicas ou processos de manufatura. Isto simplifica a engenharia de software, umas vez que não existem limites físicos para o potencial do software. Entretanto, justamente pelo motivo da falta de limites físicos, sistemas de software podem rapidamente se tornar extremamente complexos, difíceis de entender e caros para mudar.

O software é intangível, ou seja, nós não podemos vê-lo ou tocá-lo, ele está vivo enquanto seus dados e informações existirem dentro da memória de um computador. Essa na natureza o faz uma entidade abstrata, pois os usuários sabem que ele existe enquanto suas saídas estiverem sendo exibidas na tela de um computador, fornecendo relatórios, processando dados e automatizando os processos da organização. Isto de certa forma podemos concordar que simplifica a engenharia de software pois faz com que não existam limites ou restrições físicas para a criação de sistemas. O seu desenvolvimento consiste em criar uma entidade que vive alimentada pelas entradas de dados necessárias, processa-as e exibe algum resultado satisfatório na tela de um computador ao usuário.

O software é composto do hardware que o suporta e quem o faz ser vivo são os as entradas de dados e informações que ele processa.

Ao mesmo tempo em que a própria natureza do software o simplifica ela também pode complicá-lo consideravelmente. Ela simplifica o seu desenvolvimento pois isto faz com que não existam as restrições físicas que restringiriam ou limitariam o software como o produto de alguma outra engenharia, como por exemplo a civil. Se tomarmos como exemplo a construção de um prédio, vários fatores podem limitar a sua engenharia, que podem ir desde o âmbito legal até questões geográficas ou de clima. Já o software não é regido por essas condições físicas pois ele é intangível, não depende de clima ou geografia alguma para ser feito, seu desenvolvimento não está sujeito a parar por causa de alguma tempestade ou fenômeno natural, pois ele está apenas na memória de um computador. Não existem limites físicos para o software, ele pode ser de qualquer tamanho ou natureza e desempenhar qualquer tipo de função automatizando os processos de qualquer área.

Apesar desta natureza trazer benefícios para a engenharia de software e seus usuários ela própria é quem também estabelece os maiores desafios da nossa área. O software sendo intangível pode estar sujeito a fugir do controle e ficar extremamente grande e complexo. Mais uma vez ao contrário de outras engenharias, com o software muitas vezes pode acontecer das principais estruturas não terem sido devidamente desenhadas, entendidas e codificadas, e isto na maioria das vezes é percebido somente na utilização pelos usuários quando o mesmo é entregue. Diferentemente da construção de um prédio, se você está comprando um apartamento e vai acompanhar a obra, facilmente você verá se a parede que você queria está no lugar, se os ambientes estão com fácil acesso e com toda a sua construção adequada. Já o software está sujeito a erros que muitas vezes não serão percebidos tão cedo pelos usuários.

Na construção de um prédio toda a sua estrutura é projetada e ela não muda até ser entregue, a sua fundação foi toda desenhada para aquela necessidade e aquele será o projeto final. Isto não acontece com o software, pois ele muda. Muda pois os negócios das empresas são dinâmicos e estão sempre mudando para que a mesma se mantenha sempre competitiva ou muda para atender novos requisitos dos usuários, por exemplo. Ao contrário do prédio, que após o início da construção não pode ser desfeito, o software mudará durante o seu projeto. É aí que ele pode começar a fugir do controle. Aquela estrutura inicialmente desenhada previa o software de um tamanho, mas os requisitos mudaram e não foi alterada a estrutura. Ela vai passar a suportar mais carga do que devia, e na manutenção é que seus problemas vão ficar visíveis se o projeto chegar até lá. Por isso devemos ter cuidado para que as alterações e a complexidade do software não fujam do controle tanto durante o projeto quanto na manutenção, sempre pensando no que vai trazer mais benefícios no futuro.

2.2 SISTEMAS LEGADOS

Sistemas legados geralmente possuem pouca ou nenhuma documentação, são desenvolvidos em plataformas ou linguagens de programação obsoletas, são extensos e difíceis de entender, a equipe de desenvolvimento não conta com as mesmas pessoas que conheceram o seu projeto, o entendimento pode estar centralizado em poucos recursos, durante o seu desenvolvimento não foram aplicadas as melhores práticas de desenvolvimento, porém cumprem bem a sua função de suportar o processo de negócios na qual foi designado. Mas não podemos incluir somente nesta lista os sistemas muito antigos, podemos considerar que a partir

do momento em que um sistema entra em produção ele já pode ser considerado legado. Por ele cumprir bem as suas tarefas existe um grande risco de se efetuar alguma alteração nele, pois não existem na empresa alguma documentação ou as pessoas que conhecem suas particularidades.

Um sistema pode ser considerado legado quando ele não pode mais ser entendido, quando sua complexidade aumentou até chegar em um certo ponto em que qualquer mudança se torna uma tarefa inviável. A cada mudança realizada durante o ciclo de vida do software os requisitos originais do sistema vão sendo alterados, e se isto não for bem documentado e gerido, sem a utilização de um modelo de desenvolvimento moderno e sem algum esforço de reengenharia pensando na arquitetura, novas linhas de código resultarão no aumento da complexidade do sistema. Lehman (1997) atribui este fato à lei da complexidade crescente, afirmando que conforme um programa evolui, ele se torna mais complexo, e recursos extras são necessários para preservar e simplificar a sua estrutura.

Segundo (LEHMAN, 1997), um grande programa que é utilizado deve sofrer melhoria contínua, ou caso contrário com o tempo ele se tornará cada vez menos utilizável. Isto se dá pelo motivo de que conforme os processos e objetivos de negócios das empresas mudam, seus sistemas devem acompanhar essas mudanças na mesma velocidade, e a qualidade técnica desse sistemas deve ser constantemente melhorada a fim de facilitar a sua manutenção.

Em (LEHMAN, 1997) afirma que um grande programa que é continuamente alterado se deteriora conforme as mudanças vão sendo feitas a menos que um trabalho para manter ou diminuir a complexidade seja feito. É aí que a cultura de melhoria contínua se torna necessária, pois ela nos ajuda a dar o valor adequado ao legado ao mesmo tempo em que são feitas atividades para melhorá-lo ou migrá-lo.

Em (SEACORD, 1997), vemos que os sistemas se tornam legados a partir do momento em que começam a resistir a necessidade de modificações e evolução. Entretanto, o conhecimento embutido no legado constitui um ativo muito importante. Assumindo que esses sistemas continuam a fornecer valor para o negócio, eles devem ser modernizados ou substituídos.

Em (DEMEYER, 2008), o autor afirma que na maior parte dos casos o código legado não é de todo ruim. A única razão para existirem problemas com o código legado é quando as necessidades e requisitos do negócio mudaram desde que desde que o sistema original foi desenvolvido e implantado. Sistemas que foram adaptados muitas vezes devido as mudanças

nos requisitos sofrem de um deslize de desenho – chamado pelo autor de “design drift”. Isto se dá quando a arquitetura original do sistema se torna na maioria das vezes impossível de se reconhecer, e se orna impossível de se fazer maiores modificações, exatamente como previsto por Lehman nas leis da evolução do software.

Um sistema legado não é de todo ruim, pelo contrário, eles são um ativo muito importante das organizações e muitas vezes estão suportando processos vitais da empresa, por isso a dificuldade em descartá-los. Como descartar esses sistemas não é possível temos outras alternativas, como por exemplo substituí-lo a um custo muito alto ou então modernizá-lo. Conforme exposto acima, o sistema legado só passa a ser um problema quando ele precisa ser alterado mas a sua estrutura já não comporta mais melhorias ou mudanças a um custo razoável em termos de tempo e riscos de se quebrar outras funcionalidades. Em um ambiente onde são feitas mudanças desenfreadas, sem nenhuma documentação de apoio ou devidos cuidados com a qualidade do código, rapidamente podemos chegar ao que o autor se refere como “desvio de desenho” (design drift).

2.3 CICLO DE VIDA DO LEGADO

Em (SOMMERVILLE, 2011), o autor afirma que o desenvolvimento de um software não termina quando o sistema é entregue mas continua durante todo o ciclo de vida do sistema. Depois que um sistema é entregue, é inevitável ter que muda-lo para ele permanecer útil. Os negócios mudam e mudanças para atender as expectativas dos usuários geram novos requisitos para o software existente. Partes do software podem ser modificadas para corrigir os erros encontrados na operação, para adaptar ele a mudanças na sua plataforma de hardware e software ou para melhorar seu desempenho ou outras características não-funcionais.

Durante o ciclo de vida serão feitas correções de bugs e implementação de melhorias para atender a necessidade dos clientes e do negócio irão gerar novos requisitos para o software existente. Partes desse software deverão ser alteradas para suportar seu desempenho ou outras características não funcionais. Sendo assim torna-se de extrema importância que a estrutura do software esteja constantemente sendo avaliada e melhorada pensando no futuro, de uma forma que se estabeleça uma cultura de refatoração e reestruturação pró-ativa dentro da equipe de desenvolvimento sempre pensando na melhoria contínua. Essas atividades irão fazer com que cada vez menos o sistema corra o risco de sofrer o desvio de desenho, pois por mais que novos

requisitos tenham sido adicionados ao sistema depois de pronto, a sua estrutura já está apta a ser estendida de uma forma que novos componentes podem ser adicionados.

As empresas podem relutar em fazer alguma modificação devido aos riscos em se mexer em determinada parte do software, então as melhorias que poderiam ser feitas no seus processos de negócios que são suportadas pelo legado não podem ser implementadas fazendo com que o processo fique obsoleto. Pelo motivo de termos todos esses riscos embutidos, para implementar uma cultura de melhoria contínua é necessário um processo que suporte uma constante análise, avaliação, categorização e mitigação de riscos.

Nossa proposta é essa cultura de melhoria continua e gerenciamento de riscos dentro de um processo em espiral, conforme proposto por Boehm em (BOEHM, 1988), onde a cada interação os incrementos no software modernizado são documentados, desenvolvidos, testados, entregues e integrados, até que todo o software esteja modernizado. E este modelo em espiral pode ir além do projeto de modernização, podemos pensar nele em todo o ciclo de vida do software para os desenvolvimentos e adaptações futuras que ele poderá sofrer para suportar novos requisitos de usuários, do negócio e do ambiente.

Segundo (DEMEYER, 2008), o software que você utiliza para suportar o seu negócio deve ser de alguma forma adaptável, mas um sistema legado não pode ser substituído ou atualizado exceto através de um alto custo. O alvo da reengenharia é reduzir a complexidade do sistema legado suficientemente para que ele possa continuar sendo utilizado e adaptado a um custo aceitável.

O software deve ser adaptável no sentido de que permitir que os novos requisitos que ele poderá passar a possuir, originados pelas área de negócio ou dos usuários, não vão degradar ou alterar a estrutura da arquitetura existente, desta forma se tornando um sistema que vai se moldando adequadamente às necessidades de negócio da companhia. Quando esta adaptabilidade não existe mais, o software deve então ser modernizado ou substituído.

A reengenharia busca reestruturar focando os esforços na reestruturação do legado, no intuito de reduzir a complexidade, onde o legado continua a ser utilizado em pararelo às entregas do sistema modernizado em ambos trabalham em conjunto, preparando a arquitetura do novo sistema enquanto as implementações vão sendo utilizadas pelos usuários.

2.4 A CRISE DO LEGADO

Em (SEACORD, 2003), o autor chama se refere a “crise do legado” como o momento em que a organização nota que o esforço para correções ou manutenção no sistema está tomando boa parte dos recursos que seriam utilizados para novos desenvolvimentos, pois algumas tarefas de manutenção são inevitáveis, logo o custo vai aumentando até o momento em que podem não existir mais recursos para os novos desenvolvimentos.

Esta crise do legado é o que vemos como o resultado de uma das leis de Lehman, a Lei da Complexide Crescente, que diz que na medida em que o sistema evolui ele vai ficando mais complexo, e recursos extras são necessários para preservar e simplificar a sua estrutura. Desta maneira torna-se necessário o investimento no legado visando reduzir custo futuros.

Porém esta crise do legado pode ser contra-atacada com algumas forças. Por exemplo com a adoção de um modelo de desenvolvimento evolucionário, fazendo com que a manutenibilidade seja sempre mantida e cuidada, desta forma economizando recursos em todo gerenciamento do legado para correção de erros e desenvolvimento de melhorias. As organizações devem investir tempo e recursos para manter um bom nível de manutenibilidade durante todo o ciclo de vida dos seus sistemas. Uma outra força pode ser a reengenharia e a modernização do sistema, que foca a melhora nos custos de manutenção de um sistema legado existente. Com este esforço as organizações tem um trabalho de reestruturar ou modernizar os seus sistemas tendo um investimento agora para reduzir os custos futuros.

Em (SOMMERVILLE, 2011), vemos que é extremamente importante que as empresas invistam recursos para a evolução dos seus softwares, uma vez que elas investiram uma grande porção de dinheiro nesses softwares e ela é completamente dependente desses sistemas. Sendo assim, as empresas devem sempre evitar que os custos para manutenção ultrapasse a quantidade de recursos disponíveis para o desenvolvimento de novos sistemas, se isto acontecer temos a crise do legado. É inevitável que um sistema irá evoluir com o tempo pelo motivo que os negócios evoluem e mudam constantemente, e como os processos de negócio das empresas são suportados por esses sistemas eles também deverão mudar. As empresas que não investem nas melhorias em seus sistemas e apenas fazem cresce-los desenfreadamente inevitavelmente em algum momento passarão pela crise do legado ou pior ainda, a organização pode perder oportunidades de negócios simplesmente pelo fato de não conseguirem fazer com que seus sistemas que suportam os seus processos sejam evoluídos com facilidade para suportar novos requisitos dos usuários e do negócio.

2.5 PROCESSO DE MODERNIZAÇÃO DE SISTEMAS LEGADOS

Podemos combater esses problemas de projeto com o uso de um processo adequado, uma metodologia que possa ajudar a adequar essas mudanças dentro das entregas feitas constantemente ao cliente, utilizando técnicas de reengenharia e refatoração, para fazer com que as peças que consistem na estrutura do software possam ser independentes e reutilizáveis. Se nada for feito durante o projeto, o software pode entrar em produção já sofrendo da crise do legado. Inevitavelmente, cedo ou tarde esses sistemas terão que ser substituídos ou modernizados.

Nossa proposta para lidar com esses sistemas é a utilização de um modelo de desenvolvimento incremental, que suporte o desenvolvimento de novas funcionalidades adequando as mudanças que ocorrerem durante o percurso dentro de um processo em espiral, conforme proposto por Boehm em (BOEHM, 1988), com uma constante análise de riscos numa cultura de melhoria contínua, refatoração e reengenharia. Se pensarmos um pouco mais além, esta proposta de abordagem pode ser utilizada em um projeto de software desde a sua concepção até na manutenção.

Entendemos grande parte dos problemas com o legado se dão pois para as empresas o projeto termina quando o sistema é entregue, adotando um outro pensamento para a manutenção. Todo software precisa ser mantido, e isto implica em custos, e se as empresas não se atentarem a isso rapidamente sofrerão com os seus legados. A mesma metodologia utilizada no projeto deveria ser mantida na manutenção, mas o que vemos é o projeto ser entregue e toda a equipe que o fez vai embora, ficando a manutenção a cargo de uma outra equipe com seus processos particulares.

Nos demais capítulos abordaremos como as empresas podem implementar esta cultura de melhoria contínua através do uso do mesmo processo utilizado para a modernização do sistema durante a sua manutenção e por todo o seu ciclo de vida, pois ele pode fornecer entregas incrementais com o risco gerenciado, entregando sempre algo de valor para o cliente, com base nas análises de portifolio e prioridades identificadas.

3 EVOLUÇÂO DO LEGADO

No capítulo anterior vimos que todo software em algum momento do seu ciclo de vida atinge um estágio onde não é mais possível melhorá-lo ou alterá-lo sem degradarmos a sua estrutura, aumentando a complexidade ou perdendo em manutenibilidade ou desempenho. Neste capítulo vamos estudar o que faz o software atingir este estágio, o que pode ser feito para aumentarmos o seu período de estabilidade, e como podemos avaliar o portifolio de sistemas de uma empresa para determinarmos qual a estratégia e evolução mais adequada dado o momento do seu ciclo de vida.

Em (COMELLA-DORDA, 2000), o autor afirma que a evolução de um sistema corporativo com o tempo vai se tornando mais difícil, e o mesmo acaba por ficar desatualizado. Gerenciar a evolução de sistemas obsoletos requer que periodicamente esses sistemas sejam modernizados para suportar a evolução das práticas de negócio da companhia e incorporar tecnologias mais modernas de desenvolvimento e processos.

De acordo com (PRESSMAN, 2010), independentemente do domínio da aplicação, do seu tamanho, ou da sua complexidade, o software irá evoluir com o tempo. As mudanças guiarão este processo, e elas podem ocorrer quando erros são corrigidos, quando o software é adaptado a um novo ambiente, quando o cliente solicita novas funcionalidades ou quando a aplicação é reestruturada a fim de fornecer benefícios em um contexto mais moderno. Podemos considerar que um software pode ser considerado como alterado cada vez que uma nova versão é implantada em produção, por qualquer um dos motivos expostos acima.

Quando um sistema é entregue ele estará por um certo tempo atendendo as necessidades do negócio, mas será natural que elas mudem fazendo com que seja necessário melhorar o sistema. Por um certo tempo apenas a manutenção convencional consegue viver em harmonia com as melhorias feitas, mas chegará um momento em que a mesma se tornará inviável. Conforme o sistema vai sendo melhorado, a menos que ele receba alguma reengenharia, este limiar do seu ciclo de vida vai se tornando mais próximo.

3.1 MANUTENÇÃO DE SOFTWARE

Em (COMELLA-DORDA, 2000), o autor afirma que repetidas manutenções no sistema suportam as necessidades do negócio suficientemente por um certo tempo, mas como o sistema vai se tornando desatualizado, a manutenção falha em atender as necessidades do

negócio. Quando este momento chega um esforço maior de modernização se torna necessário, tanto em tempo quanto em funcionalidades entregues pelas atividades convencionais de manutenção. Finalmente, quando o sistema antigo não pode mais ser evoluído, ele deve então ser substituído.

De acordo com (PRESSMAN, 2010), a manutenção de software corrige defeitos, adapta o software para cumprir as necessidades de uma mudança de ambiente e melhora funcionalidades existentes para atender as necessidades dos usuários.

Podemos considerar que a manutenção é uma tarefa que todo sistema estará sujeito a ser submetido. É natural que correções de bugs ou pequenas melhorias sejam desenvolvidas depois que um sistema é entregue. Porém a manutenção convencional possui limites, quando for necessário um esforço maior em termos de alterações estruturais no sistema um esforço à parte deve ser feito. Essas melhorias com o tempo tendem a degradar a estrutura do sistema, tornando-o mais complexo e de difícil manutenção. Cabe a organização identificar este momento do ciclo de vida do software, fazendo com que ele se mantenha o máximo de tempo suportando a dinâmica que o negócio precisa mantendo-o atualizado. Dado um certo momento no seu ciclo de vida, a manutenção se tornará cara e custosa, tornando-se inviável, quando isto acontece, o sistema deve ser substituído ou modernizado.

Em (DEMEYER, 2008), o autor afirma que de um ponto de vista funcional, alguma coisa está quebrada quando não entrega mais a função que foi designada a desempenhar. Do ponto de vista da manutenção, entretanto, um pedaço de software está quebrado quando não puder mais ser mantido.

3.2 A MANUTENÇÃO DENTRO DO CICLO DE VIDA DO LEGADO

Entendemos que esta é a maior dificuldade e o que faz as empresas não conseguirem identificar tão facilmente quando o software vai chegando no fim do seu ciclo de vida, pois não importando o quão boa esteja a estrutura e ou a manutenção do sistema, eles estão utilizando as suas funcionalidades para cumprir suas atividades no negócio. Já se olharmos para o ponto de vista da manutenção podemos encontrar uma série de motivos que podem fazer com que uma parte do software esteja quebrada, como falta de documentação, complexidade alta ou mesmo falta do conhecimento dentro da equipe. Isto faz com que com o tempo ela se torne mais custosa e cara, muitas vezes durante o ciclo de vida o custo da manutenção ultrapassa o custo inicial do

sistema quando desenvolvido e as melhorias e correções de bugs vão ficando mais caras e custosas, fazendo com que as necessidades do negócio não sejam mais atendidas como a companhia gostaria.

Muitas companhias tem dificuldade em notar quando seus sistemas atingem este limiar em seu ciclo de vida em que a manutenção se torna muito cara e custosa e é difícil implementar melhorias ou mesmo correções de bugs sem degradar a estrutura interna do mesmo ou aumentar demais a sua complexidade, comprometendo o entendimento.

Para evitar com que um sistema se torne quebrado do ponto de vista da manutenção as empresas devem ter uma cultura de constante avaliação do seu portifolio de sistemas legados para determinar qual ou quais estão poeto do seu limite do ciclo de vida e traçar a estratégia necessária conforma a sua qualidade e valor para o negócio.

Em (SOMMERVILLE, 2011), o autor afirma que muitas organizações geralmente possuem um portifolio de sistemas legados que elas usam, com uma quantia de dinheiro limitada para manter ou modernizar esses sistemas, e elas devem decidir como obter o melhor retorno no seu investimento. Isto envolve efetuar uma avaliação realista dos seus sistemas legados e decidir qual a estratégia mais apropriada para evoluir esses sistemas. Para isto é extremamente necessário que a TI esteja alinhada com os stakeholders e com o negócio, pois é fundamental que todos estejam comprometidos com a estratégia de modernização que será adotada.

De acordo com (PRESSMAN, 2010), conforme o tempo passa as organizações se deparam com o fato de estarem gastando mais tempo e dinheiro mantendo seus sistemas do que desenvolvendo novas aplicações ou projetos. Ele afirma que não é raro para as organizações gastar cerca de 60% a 70% de todos os seus recursos na manutenção de software. Mudanças são inevitáveis em sistemas, entretanto devemos desenvolver mecanismos para avaliar, controlar e efetuar as modificações no software.

Na concepção de um novo software a manutenção é um ponto muito importante que geralmente é deixado de lado. Comumente a equipe do projeto não é a mesma equipe que irá manter o sistema, o que de início já pode causar problemas pela falta de conhecimento do time. Todo software que é concebido deveria levar em conta a manutenibilidade futura. Em (PRESSMAN, 2010), o autor afirma que tanto a análise quanto o desenho de um sistema devem visar atender algumas características de manutenibilidade. Em essência, ele afirma que a

manutenibilidade é uma indicação qualitativa do quão fácil é para o software existente ser corrigido, adaptado ou melhorado.

Em (PRESSMAN, 2010), o autor define um software com boa qualidade de manutenção um software que exibe modularidade, ou seja, um software bem estruturado e dividido em componentes que podem ser reutilizados, e que faz uso de design patterns para facilitar o entendimento. Ele deve ser construído utilizando-se padrões e de código e convenções bem definidas, guiando para um código que seja auto-documentável e entendível.

Esses fatores expostos acima devem fazer parte do pensamento do time do projeto no momento de conceber um novo software pois as decisões tomadas antecipadamente na criação, no momento do entendimento e no desenho da solução que resolverá o problema para um cliente certamente deverá ser mantido no futuro, e as decisões lógicas e estruturais devem ser bem pensadas no início pois serão as mesmas que ditarão o quão fácil o software é de ser mantido.

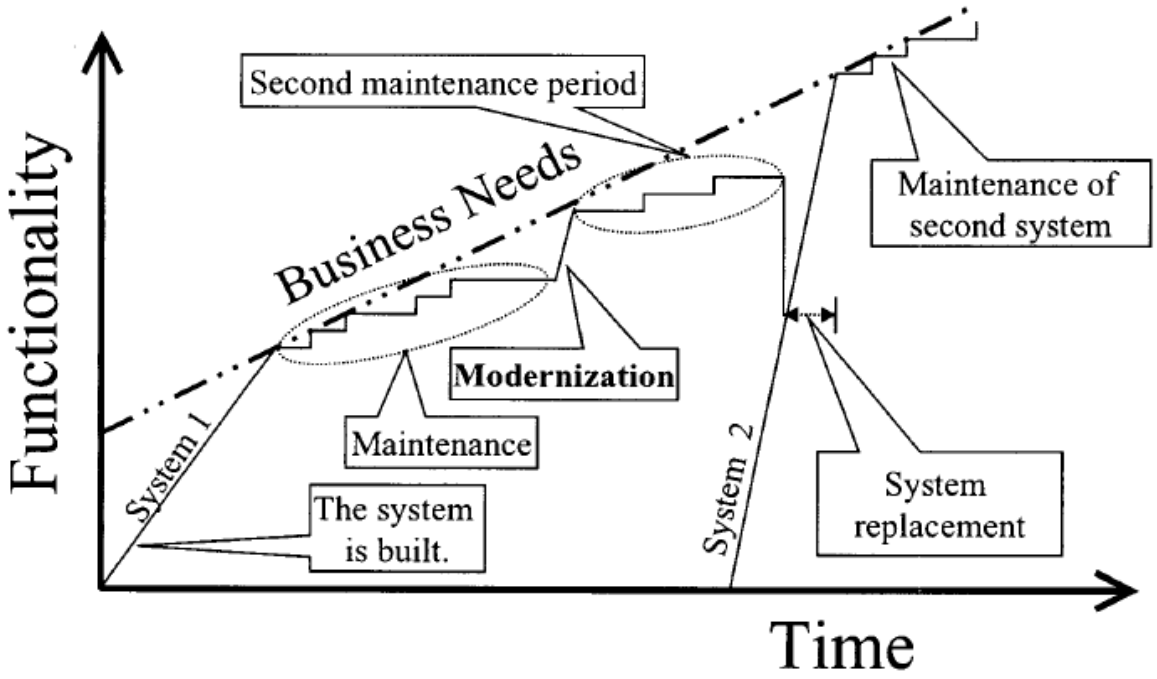

Figura 1: O ciclo de vida de um sistema de informação (COMELLA-DORDA, 2008)

Segundo (COMELLA-DORDA, 2000), a evolução de um sistema é um assunto amplo que pode ser dividido em três categorias: manutenção, modernização e substituição. A Figura 1 ilustra como essas diferentes atividades de evolução são aplicadas durante as diferentes fases

do ciclo de vida de um sistema. Podemos notar que as necessidades do negócio por novas funcionalidades aumenta constantemente com o tempo, representado pelo linha pontilhada, enquanto a linha sólida representa as funcionalidades fornecidas pelo sistema.

Quando o sistema é entregue ele estará atendendo as necessidades do negócio por um tempo, durante um certo tempo a manutenção conseguirá cuidar dos novos requisitos que irão modificar o sistema original. Enquanto a arquitetura do sistema estiver mais parecida com aquela inicial, as entregas de novas funcionalidades executadas nas tarefas de manutenção estará em linha com o negócio pois será possível entregar novas versões do sistema incrementalmente, em um curto período de tempo e a um custo menor.

Porém conforme o sistema vai sendo modificado, ficando mais complexo e distante da arquitetura que foi inicialmente concebido, essas entregas passam a necessitar de um grande esforço da equipe e a um custo maior. As entregas passarão a não conseguir atingir completamente as funcionalidades que o negócio precisa e conforme o time trabalha para implementar as mesmas, novos requisitos vão surgindo.

A manutenção é necessária para suportar a evolução de qualquer sistema mas é limitada, pois entrega apenas correções de bugs, pequenas melhorias e funcionalidades mas não envolve melhorias que envolvam alguma mudança estrutural e seu custo aumenta com o tempo, então o sistema vai ficando desatualizado. Ela atende as necessidades do negócio por um tempo, mas em um determinado momento, quando a manutenção não estiver mais atendendo o negócio, o sistema deverá ser substituído ou modernizado.

3.3 MODERNIZAÇÃO DO LEGADO

Na modernização as novas entregas de funcionalidades serão interrompidas por um período de tempo pois nela serão feitas algumas tarefas que a manutenção não consegue entregar, por exemplo uma reestruturação do sistema para tornar ele mais enxuto através da refatoração, tornando-o menos complexo e menor. Este esforço de modernização deverá implicar na necessidade de um esforço maior tanto em tempo, custo e em funcionalidades do que as tarefas da manutenção.

Após o projeto de modernização entregar uma nova versão do sistema ele estará novamente alinhado com necessidades do negócio e entregará um conjunto de novas funcionalidades que não possuía antes, que apenas com as tarefas que a manutenção não estava

sendo possível executar. A entrega desta nova versão do sistema será então seguida por um novo período de manutenção, que novamente suportará as entregas de novas funcionalidades, pequenas melhorias e correções de bugs com as suas entregas incrementais.

A modernização de um sistema entregará uma nova versão do mesmo sistema anterior porém reestruturado, ele será remodelado para que a sua estrutura suporte por um certo tempo esse novo período de manutenção que seguirá. Ela não implicará na criação de um novo sistema com uma nova tecnologia, por exemplo, e deverá ter cuidado em manter o sistema parecido com o originalmente criado. A nova fase de manutenção que seguirá será semelhante a primeira, uma vez que como sistema ainda é muito parecido com o original o conhecimento da equipe deverá ser mantido, porém as suas entregas passarão a ter mais dificuldade em atender as necessidades do negócio tanto em tempo quanto em custos.

A segunda fase de manutenção demandará um esforço maior e deverá entregar menos funcionalidades em janelas maiores de tempo do que aquelas do primeiro período, pois apesar do trabalho de reestruturação, as novas mudanças feitas irão aumentar ainda mais a sua complexidade, fazendo que com o passar do tempo elas não estarão atendendo novamente as necessidades do negócio e então novamente o sistema estará desatualizado.

3.4 SUBSTITUIÇÃO DO SISTEMA LEGADO

É necessário então identificar se o sistema suporta uma nova reestruturação a ser seguida por uma nova fase de manutenção, mas conforme vemos na Figura 1 quanto mais distantes estamos daquele sistema que foi originalmente entregue deverá ser mais difícil de fazer com que o sistema se mantenha atualizado na velocidade que o negócio necessita. Quando o sistema se torna então desatualizado, as tarefas de manutenção difíceis novamente e uma nova reestruturação é inviável, o sistema deverá então ser substituído.

Segundo (COMELLA-DORDA, 2000), a substituição por um novo sistema deve então ser considerada quando as necessidades do negócio não conseguem viver em paz com o sistema atual e a sua modernização não pode ser realizada por ser de alto custo. A substituição normalmente é utilizada em sistemas não documentados, não extensíveis ou desatualizados.

A substituição do sistema irá entregar então um novo software atendendo as necessidades do negócio como na primeira versão, exatamente como quando o sistema original foi construído, e novamente será seguida das atividades de manutenção. O desenvolvimento

deste segundo sistema implicará num esforço ainda maior do que a modernização do primeiro em termos de tempo e custo.

3.5 ABORDAGENS ADOTADAS DADO O CICLO DE VIDA DO SISTEMA

É uma tarefa muito difícil para as empresas saber determinar qual tarefa é a mais apropriada para um determinado momento do ciclo de vida do sistema, dentre continuar mantendo, modernizá-lo ou substituí-lo. Torna-se então necessário efetuar uma avaliação consistente do sistema e de todos os legados para determinar qual a atividade de evolução é a mais adequada a ser adotada dado o estágio de evolução de cada um e avaliar o impacto de cada uma delas. Adiante iremos discutir algumas técnicas e como esta avaliação pode ser efetuada, mas antes disto iremos discutir o escopo das atividades de manutenção e modernização antes de adentrarmos na opção pela substituição do sistema.

3.5.1 Manter o sistema

Segundo (COMELLA-DORDA, 2000), a manutenção é o processo iterativo e incremental no qual pequenas modificações são feitas no sistema. Essas mudanças envolvem correções de bugs ou pequenas melhorias funcionais e nunca devem envolver mudanças mais significativas estruturalmente. Ela é necessária para suportar a evolução de qualquer sistema, mas possui algumas limitações:

- Vantagens competitivas derivadas da adoção de alguma tecnologia mais recente estão seriamente comprometidas, uma vez que a adoção de uma nova tecnologia muitas vezes pode implicar em mudanças na aplicação que não são consideradas tarefas de manutenção. Por exemplo, um sistema *web* obsoleto que não utiliza os recursos mais modernos disponíveis atualmente para melhorar a experiência do usuário na sua *user interface*, não consegue ser atualizado pois o desenvolvimento de uma nova tela com a adoção de novas tecnologias não se encaixa no escopo da manutenção.
- O custo da manutenção de sistemas legados aumenta com o tempo, pois, por exemplo, para se encontrar a expertise necessária em alguma tecnologia mais

obsoleta pode ser difícil e caro. Com a dinâmica do mercado de tecnologia dos dias atuais e com a adoção de muitas tecnologias mais recentes por parte das empresas em novos projetos, encontrar os profissionais com a experiência necessária em alguma tecnologia mais específica têm se tornado cada vez mais difícil e caro.

- E um terceiro ponto um pouco mais técnico, que diz que o impacto composto de muitas modificações pequenas acaba se tornando muito maior do que a soma individual das mesmas, devido a erosão da integridade conceitual do sistema. Isto porque os sistemas tendem a se expandir com o tempo, enquanto raramente algum esforço para remover código inutilizado é financiado. Adaptar um sistema para atender as novas necessidades do negócio vai se tornando cada vez mais difícil. Conforme falamos anteriormente, quanto mais o sistema vai sendo modificado ficando, ficando cada vez mais longe daquele sistema original que foi construído, sua complexidade aumenta consideravelmente tornando a manutenção uma tarefa difícil e cara.

Em (SOMMERVILLE, 2011), o autor define manutenção de software como o processo geral de alterar um sistema depois que ele foi entregue. As mudanças feitas no software pode ser simplesmente mudanças para corrigir erros de codificação, mudanças mais extensas para corrigir erros de desenho ou melhorias mais significantes para corrigir erros de especificação e acomodar novos requisitos. O autor classifica as tarefas de manutenção de software em três tipos: Correção de falhas, adaptação ao um ambiente e adição de funcionalidades.

As tarefas de correções de falhas são relativamente mais baratas de se corrigir quando for algum erro de codificação, já erros de desenho se tornam mais caros pois podem implicar em mudanças estruturais reescrevendo diversos componentes. A correção de algum erro de codificação não deve impactar grandes componentes do sistema e ser alguma coisa pontual, não envolvendo grandes riscos ou mudanças maiores no sistema. Já alguma tarefa de correção de algum erro de desenho é mais custosa por geralmente impactar componentes que podem estar relacionados a estrutura do sistema, tornando as mudanças mais significativas em esforço de tempo e custo.

A adaptação a um novo ambiente se torna necessária quando algum aspecto do ambiente do sistema é alterado como o sistema operacional que ele é executado ou alguma mudança de hardware, por exemplo. Isto envolve a adoção de uma versão mais recente de uma linguagem de programação, alguma atualização de framework ou migração do servidor de uma aplicação web para uma plataforma mais moderna.

A adição de funcionalidades é necessária quando os requisitos do sistema mudam em resposta a alguma mudança organizacional ou de negócio. A escala de mudanças deste tipo requeridas no sistema geralmente é maior do que a demanda dos outros tipos de manutenção. Neste tipo de manutenção o sistema vai sendo estendido para acomodar os novos requisitos, e isto pode envolver a criação de novos componentes, refatoração ou alguma mudança estrutural mais significativa.

Em (SOMMERVILLE, 2011), o autor ainda afirma, porém, que não há uma distinção clara entre esses tipos de manutenção. Quando você adapta um sistema a um novo ambiente, você pode adicionar funcionalidades para obter benefícios de alguns recursos deste novo ambiente. Alguns erros no software podem acontecer quando os usuários utilizam o sistema de alguma maneira que não foi prevista, e alterar o sistema para acomodá-lo a maneira que ele utiliza é a melhor maneira de corrigir esses problemas. Logo alguma alteração de ambiente necessária ao sistema pode fazer com que sejam gerados novos requisitos do negócio, então esses tipos de manutenção podem aparecer em conjunto e originar uma nova demanda, assim como as mudanças que são originadas para adaptar o sistema à maneira que seus usuários utilizam pode não necessariamente estar relacionada a algum erro na codificação ou desenho em si.

3.5.2 Substituir o sistema

Segundo (COMELLA-DORDA, 2000), a substituição do sistema é apropriada quando os sistemas legados da organização não conseguem mais conviver em paz com as necessidades do negócio e quando a modernização não é possível ou muito cara. A substituição normalmente é utilizada em sistemas não documentados, ultrapassados ou não-extensíveis.

A substituição do sistema por outro totalmente novo deve ser considerada quando os esforços para modernizar o legado forem inviáveis e o sistema atual não comportar mais alguma

reestruturação sem aumentar demasiadamente a sua complexidade e esforços posteriores de manutenção.

Em (SEACORD, 2003), o autor afirma que a substituição do sistema possui alguns riscos que devem ser considerados antes de se optar por essa técnica:

- A substituição basicamente é construir um sistema completamente novo e consome muitos recursos, com isto os recursos de TI que estão tipicamente todos alocados nas tarefas de manutenção podem não estarem familiarizados com as novas tecnologias que podem ser utilizadas no novo sistema.
- A substituição requer testar intensivamente o novo sistema. Sistemas legados geralmente são bem testados e otimizados, e embutem uma expertise considerável do negócio com o passar do tempo. Não há garantias de que o novo sistema será tão robusto ou funcional como o antigo. Conforme vimos na Figura 1, a substituição pode causar um período de degradação das funcionalidades que o negócio necessita.

O autor afirma que os sistemas podem ser substituídos todos eles de uma vez usando uma abordagem “big-bang” ou incrementalmente. Substituir um sistema incrementalmente pode ajudar se o sistema possuir algum grau de modularização ou coesão. Porém, o sistema legado deve sofrer um trabalho de reengenharia como um passo preparatório antes de iniciar um esforço de substituição incremental.

3.5.3 Modernizar o sistema

Segundo (SEACORD, 2003), a modernização envolve mudanças mais extensivas do que a manutenção mas conserva uma porção significante do sistema atual. Essas alterações geralmente incluem reestruturar o sistema, melhorar funcionalidades ou modificar alguns atributos do software.

A modernização é utilizada quando o sistema legado requer mudanças mais profundas do que as possíveis durante a manutenção, mas continua preservando todo o valor do conhecimento de negócio embutido no legado.

A modernização de um sistema pode ser diferenciada pelo nível de entendimento requerido para suportar o esforço de modernização (COMELLA-DORDA, 2000): White-box ou black-box. A modernização White-box requer conhecimento das estruturas internas do sistema legado, já a modernização black-box requer o conhecimento somente das interfaces externas do sistema.

3.5.3.1 Modernização White-Box

A modernização White-box requer uma engenharia reversa inicial para obter o entendimento do funcionamento interno das operações do sistema. Os componentes do sistema e os seus relacionamentos são identificados, e uma representação do sistema em alto nível é produzida.

A técnica White-box é recomendada quando o projeto de modernização deverá ser feito por uma equipe que já conheça o funcionamento do sistema, desta forma um grande ganho na fase do entendimento inicial é obtido. A equipe que mantém o sistema conhece as suas particularidades e seu conhecimento é de grande valia num projeto deste tipo de modernização.

Esta técnica garante mantém todo o conhecimento de negócio embutido no sistema durante o seu ciclo de vida e produz um novo software que seja de fácil entendimento para ser posteriormente mantido, justamente por mantar boa parte deste conhecimento porém de uma maneira reestruturada.

O entendimento dos sistemas é a principal forma de engenharia reversa utilizada na modernização White-box. O entendimento dos sistemas envolve modelar o domínio, extrair informações do código utilizando mecanismos de extração apropriados e criar abstrações que ajudem a entender a estrutura do sistema em questão.

Analisar e entender código antigo é uma tarefa difícil pois com o tempo todo sistema entra em um colapso de complexidade com todas as manutenções realizadas sem algum esforço de refatoração. É uma tarefa que envolve riscos e um trabalho intensivo.

Após o código-fonte ser analisado e entendido, a modernização White-box geralmente inclui alguma reestruturação do sistema ou no código. (CHIKOFSKY, 1990) define a reestruturação do software como a transformação de uma forma de representação atual para outra relativamente no mesmo nível de abstração, enquanto preservamos o comportamento externo do sistema.

Esta transformação tipicamente é utilizada para melhorar algum atributo de software do sistema como manutenibilidade ou desempenho. Adiante discutiremos algumas técnicas de modernização que podem ajudar a melhorar as estruturas de um sistema legado, assim como discutiremos a engenharia reversa e a reengenharia e como as aplicamos num sistema legado a ser modernizado.

3.5.3.2 Modernização Black-Box

Modernização black-box, segundo “A survey of black-box modernization approaches for information systems”, envolve examinar as entradas e saídas do sistema legado dentro de um contexto para ganhar o entendimento das interfaces do sistema. Apesar de que adquirir um entendimento da interface de um sistema não é uma tarefa fácil, ela não atinge o nível de dificuldade associado a modernização White-box.

Uma modernização black-box é geralmente baseada no ato de envolver o sistema legado. Isto consiste em envolver o sistema legado em uma nova camada de software que esconde a complexidade não desejada do legado e exporta uma interface moderna. Idealmente, envolver é uma técnica de reengenharia black-box onde somente a interface legada é analisada, e a parte interna do sistema legado é ignorada. Esta técnica nem sempre é aplicável, e muitas vezes requer um entendimento dos módulos internos do sistema usando técnicas White-box.

Embora a modernização black-box produza um novo software modernizado nós não estamos livres dos problemas do legado, uma vez eu ele ainda existe em uma outra camada, e nos depararemos com eles novamente se em algum momento a parte do legado que foi envolvida necessitar de alguma mudança ou manutenção.

Esta abordagem de modernização não deve conseguir resolver os problemas do sistema antigo se o caso for de problemas de desempenho, ou mesmo se o legado possuir uma estrutura que não esteja minimamente modularizada permitindo que a técnica black-box extraia interfaces independentes.

A proposta deste trabalho é fornecer uma alternativa de modernização do tipo White-box, entregando sucessivas versões do sistema modernizado a cada interação, até que todo o sistema esteja modernizado.

3.6 ANÁLISE DO PORTIFOLIO DE SISTEMAS

Antes de iniciarmos qualquer esforço de modernização, precisamos levantar quais são os sistema legados da organização e classifica-los, fazendo uma avaliação e análise do portifolio a fim de identificarmos quais são os sistemas relevantes que necessitem de algum esforço de modernização.

Segundo (SEACORD, 2003), a análise de portifolio mede a qualidade técnica e o valor para o negócio de um conjunto de sistemas e avalia este conjunto sobre essas métricas.

3.6.1 Qualidade técnica x Valor para o negócio

Qualidade técnica é a medida do quão boa é a aplicação ou o sistema sobre um conjunto de critérios técnicos. Alguns exemplos de critérios técnicos para a qualidade técnica podem incluir a frequência de novos releases, facilidade de efetuar mudanças, segurança do hardware e software, infraestrutura organizacional, desempenho do sistema, precisão ou facilidade de operação. Muitos desses critérios podem facilmente ser avaliados pela equipe que mantém e conhece tecnicamente o sistema e pelos seus usuários.

O valor para o negócio é a medida da importância do sistema ou aplicação para a organização. Exemplos de critérios de valor para o negócio são contribuição em termos de receita, nível de uso, número de objetivos do negócio satisfeitos por ele, valor do sistema, satisfação dos usuários ou o valor da informação que o sistema armazena.

3.6.1.1 Avaliação do valor para o negócio

Em (SOMMERVILLE, 2011), o autor mostra que para avaliar o valor para o negócio de um sistema temos que identificar os seus stakeholders, como usuários finais e seus gerentes, e fazer uma série de perguntas sobre o sistema. Existem quatro problemas básicos que devemos discutir: a utilização do sistema, os processos de negócio que são suportados, a segurança do sistema e as suas saídas.

A utilização do sistema deve ser discutida para levantarmos quais sistemas são utilizados ocasionalmente ou por um número muito baixo de usuários, se isto ocorrer, ele pode ter um baixo valor para o negócio. Um sistema legado pode ter sido desenvolvido para uma

necessidade de negócio que pode ter sido alterada ou que hoje pode ser obtida de outras maneiras mais efetivamente. Devemos tomar cuidado com esta medida pois alguns sistemas podem ser pouco utilizados mas de grande valia por cumprirem papéis extremamente importantes, por exemplo, um sistema de matrículas de alunos que é utilizado somente o início de cada ano letivo mas desempenha uma tarefa extremamente essencial.

Os processos de negócio que o sistema suporta devem ser discutidos pois, quando um sistema é introduzido, processos de negócio são designados para explorar as capacidades desse sistema. Se o sistema é inflexível, mudar esses processos de negócio pode ser impossível. Entretanto, conforme o ambiente da organização muda, esses processos de negócio originais podem acabar ficando obsoletos. Este é o ponto que deve ser discutido aqui, do quão vitais ou extremamente importantes esses processos de negócio que o sistema suporta são importantes para a organização. Entretanto, devemos considerar que um sistema pode ter baixo valor para o negócio porque ele força o uso desses processos de negócios ineficientes, justamente pelo motivo de ser difícil modificá-lo.

A segurança do sistema não é somente um problema de negócio mas também técnico. Se o sistema não é confiável e os problemas que afetam diretamente os clientes do negócio ou as pessoas do negócio são desviados para outras tarefas para resolver estes problemas, então o sistema tem um baixo valor para o negócio. Devemos considerar que se ele tivesse um alto valor para o negócio esses problemas não teriam algum caminho alternativo para serem resolvidos.

3.6.1.2 Avaliação da qualidade técnica

Devemos avaliar as saídas do sistema para sabermos a importância que elas tem para o sucesso do funcionamento do negócio. Se o negócio depende dessas saídas, então o sistema tem um alto valor para o negócio. Em recíproca, se essas saídas pode ser facilmente geradas de alguma outra maneira ou se o sistema produz saídas que são raramente usadas, então o sistema pode ter um baixo valor para o negócio.

Em (SOMMERVILLE, 2011), o autor afirma que para avaliar os sistemas no ponto de vista técnico devemos considerar tanto a aplicação em si quanto o ambiente no qual o sistema opera. O ambiente inclui todo o hardware e todo o software de suporte associado, como

compilador, sistema operacional, ambientes de desenvolvimento e etc. que são necessários para manter o sistema.

Nesta avaliação do ambiente, se possível devemos considerar também métricas do sistema e dos seus processos de manutenção. Por exemplo, podemos levantar dados como o custo para se manter o sistema, o número de falhas ocorridas num período de tempo e a frequência em que patches e correções precisam ser aplicadas no sistema.

Na Tabela 1, (SOMMERVILLE, 2011) mostra alguns fatores que devem ser levados em consideração na avaliação do ambiente do sistema. Nem todos eles são de cunho técnico do ambiente, também devemos considerar fatores como o suporte existente, disponibilidade do fabricante e etc.:

Fator	Questões
Estabilidade do fornecedor	O fornecedor ainda existe? O fornecedor é estável financeiramente para continuar existindo? Se o fornecedor encerrar seu negócio, alguém pode manter os sistemas?
Nível de falhas	O hardware possui um alto nível de falhas reportadas? O software de suporte falha e força o reinício do sistema?
Idade	Quão velho é o hardware e o software? Quanto mais velho for o hardware e o software de suporte, mais obsoleto ele deverá ser. Ele pode continuar funcionando corretamente, mas podem existir benefícios significantes economicamente e de negócios que podem ser obtidos movendo para um sistema mais moderno.
Desempenho	O desempenho do sistema é adequado? Os problemas de desempenho causam algum efeito significante nos usuários?
Necessidade de suporte	Qual tipo suporte local é requerido pelo hardware e pelo software? Se existir um alto

	custo associado a este suporte, pode ser melhor considerar a substituição do sistema.
Custos de manutenção	Quais são os custos de manutenção do hardware e licenças dos softwares de suporte? Hardware antigo pode ter um custo de manutenção maior do que sistemas modernos. Software de suporte pode ter um alto custo anual de licenças.
Interoperabilidade	Existem problemas de interfaces desse sistema com outros sistemas? Os compiladores, por exemplo, podem ser utilizados com as versões atuais do sistema operacional? É necessária alguma emulação de hardware?

Tabela 1 - Avaliação do ambiente do sistema (SOMMERVILLE, 2011)

Segundo (SOMMERVILLE, 2011), para avaliar a qualidade técnica de uma aplicação devemos analisar uma série de fatores, expostos na Tabela 2, que estão primariamente relacionados as dependências do sistema, as dificuldades para manter o sistema e a sua documentação. Alguns dados adicionais podem ser coletados para ajudar a julgar a qualidade do sistema, por exemplo:

- O número de solicitações de mudança no sistema, pois conforme discutimos anteriormente quanto maior o número de alterações feitas elas tendem a degradar a estrutura do sistema, tornando as mudanças cada vez mais difíceis. Quanto maior for este número de mudanças, menor deverá ser a qualidade do sistema.
- O número de interfaces com o usuário é um fator importante em sistemas baseados em formulários onde cada formulário pode ser considerada uma interface com o usuário separada. Quanto mais interfaces, possivelmente teremos mais inconsistências e redundâncias entre essas interfaces.

- O volume de dados utilizado pelo sistema, pois quanto maior for o volume de dados, incluindo número de arquivos, tamanho do banco de dados e etc.) mais possivelmente poderão existir inconsistências entre esses dados, reduzindo a qualidade do sistema.

Fator	Questões
Entendimento	Quão difícil é entender o código fonte do sistema atual? Quão complexas são as estruturas de controle que ele utiliza? As variáveis possuem nomes claros que refletem a sua função?
Documentação	Qual a documentação do sistema disponível? A documentação está completa, consistente e atualizada?
Dados	Existe um modelo da dados explícito para o sistema? Até que ponto os dados estão duplicados entre os arquivos? Os dados utilizados pelo sistema estão consistentes e atualizados?
Desempenho	O desempenho da aplicação é adequado? Os problemas de desempenho causa algum efeito significante nos usuários?
Linguagem de Programação	Existem compiladores modernos disponíveis para a linguagem de programação utilizada para desenvolver o sistema? A linguagem de programação continua sendo utilizada para novos desenvolvimentos no sistema?
Gerenciamento da Configuração	Todas as versões e partes do sistema são gerenciados por um sistema de gerenciamento da configuração? Existe uma descrição explícita das versões dos

	componentes que são utilizados atualmente pelo sistema?
Dados de Testes	Existem dados de teste para o sistema? Existe algum registro dos testes de regressão realizados quando novos recursos foram adicionados ao sistema?
Habilidades Pessoais	Existem pessoas disponíveis que possuem as habilidades necessárias para manter a aplicação? Existem pessoas disponíveis que possuem experiência com o sistema?

Tabela 2 – Avaliação da qualidade técnica do sistema (SOMMERVILLE, 2011)

3.7 PRIORIZAÇÃO DOS CANDIDATOS À MODERNIZAÇÃO

Após o levantamento do portifolio de sistemas da organização e a avaliação da qualidade dos mesmos devemos decidir o que fazer com eles, e para isto (SEACORD, 2003) e (SOMMERVILLE, 2011) propõem enquadrá-los em um gráfico ilustrado na Figura 2.

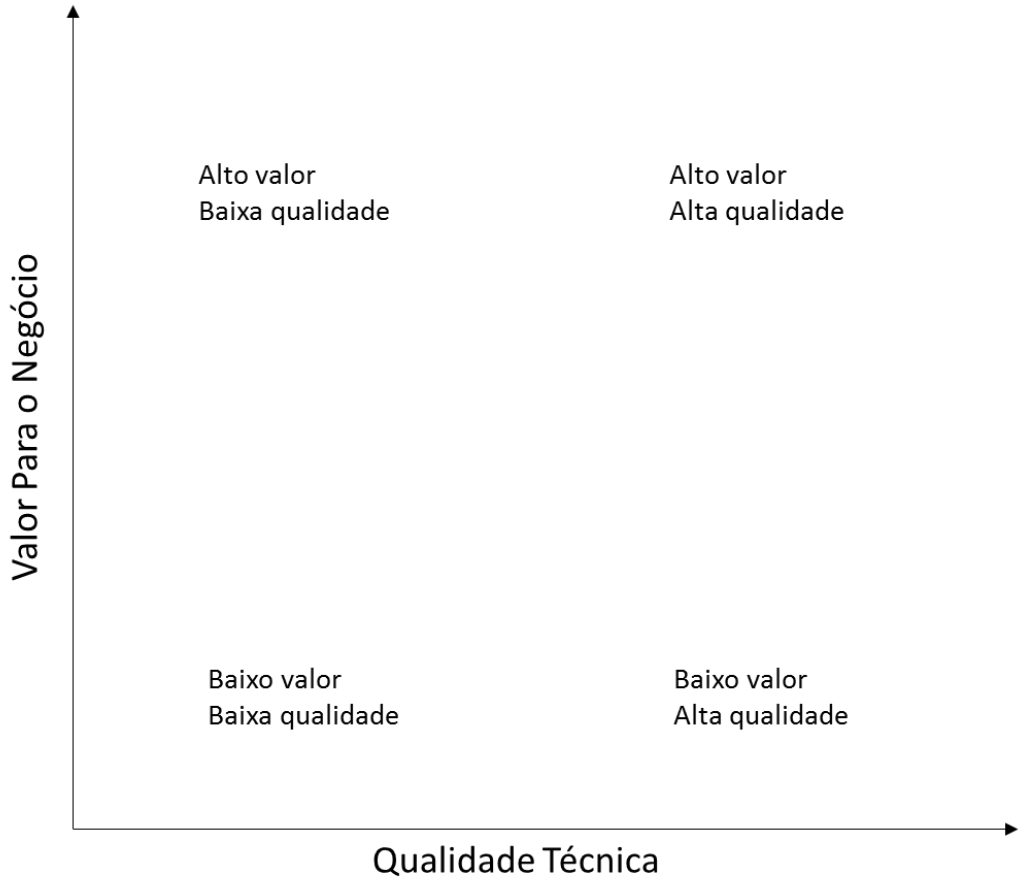

Figura 2: Valor para o negócio X Qualidade técnica em sistemas

Este enquadramento dos sistemas de acordo com a sua qualidade técnica e valor para o negócio é que deve nos auxiliar a tomar a melhor decisão que se aplique a cada sistema dado o momento do seu ciclo de vida. Dependendo do enquadramento do sistema dento da Figura 2, temos quatro ações que podem ser tomadas: manter o sistema como está, substituí-lo por algum componente comercial, reestruturar porém numa prioridade menor ou bons candidatos para a reestruturação com maior prioridade.

Muitas organizações geralmente tem um portifolio dos sistemas legados que ela utiliza com um orçamento limitado para manter e melhorar esses sistemas. Elas devem decidir como obter o melhor retorno no seu investimento fazendo uma avaliação realista dos seus sistemas legados e então decidir pela estratégia mais apropriada para evoluir esses sistemas (SOMMERVILLE, 2011). Efetuada essa avaliação tanto da qualidade técnica quanto do valor para o negócio, teremos quatro opções para modernizar esses sistemas: reestruturar para melhorar a sua manutenibilidade, manter o sistema inalterado e continuar com a manutenção

regular, substituir o sistema todo ou alguma parte dele por um novo sistema, ou mesmo sucatear o sistema.

Quando o sistema estiver com a sua qualidade degradada pelas mudanças que foram feitas durante seu ciclo de vida e novas mudanças ainda se tornam comumente necessárias, ele deve ser um forte candidato a passar por uma reestruturação aumentar a sua qualidade e para reduzir a complexidade e os custos de manutenção atuais. Este processo pode incluir também desenvolvimento de novos componentes de interfaces para que o sistema possa trabalhar junto a outros sistemas mais novos, facilitando o projeto de modernização (SOMMERVILLE, 2011).

Devemos manter um sistema em produção e continuar com as atividades de manutenção convencionais quando o sistema é necessário para o negócio da organização mas é estável, não custa caro para se manter e não são necessárias muitas mudanças por solicitação dos usuários. Esses sistemas funcionam relativamente bem e devem apenas sofrer um grande cuidado para que a sua qualidade seja mantida como está, através de um processo de desenvolvimento adequado que garanta que ele não se degrade com o tempo ou que ao menos sua vida útil seja postergada, evitando que ele se torne um candidato a modernização devido a degradação da qualidade.

Substituir todo o sistema ou partes dele por um novo sistema deve ser considerado quando alguns fatores, por exemplo um novo hardware, significar que o antigo sistema não pode continuar em operação ou aonde ferramentas prontas podem ajudar a desenvolver o novo sistema a um custo mais razoável. Segundo (SOMMERVILLE, 2011), em muitos casos uma estratégia de substituição evolucionária pode ser adotada em um sistema maior onde seus componentes são substituídos por componentes de sistemas prontos, com outros componentes sendo reutilizados se possível.

Segundo (SOMMERVILLE, 2011), sucatear o sistema completamente deve ser a opção considerada quando o sistema não estiver tendo alguma contribuição efetiva aos processos de negócio da organização. Isto ocorre geralmente quando os processos de negócio mudaram desde que o sistema foi entregue, e eles não estão mais confiáveis dentro do sistema legado. Esta opção deve ser considerada quando se tratar de um sistema que já esteja tendo um baixo valor para o negócio, muito provavelmente a função que ele deveria cumprir já esteja atualmente sendo cumprida por agum outro caminho dentro dos processos de negócio da organização, então ele pode ser sucateado.

Segundo (SEACORD, 2003), sistemas que se enquadrem no caso de baixo valor para o negócio e baixa qualidade técnica são candidatos a serem substituídos por algum componente comercial, por duas razões: a primeira é que eles tem uma baixa qualidade técnica, então eles precisam ser melhorados ou substituídos. Segundo porque como eles tem baixo valor para o negócio, eles não fornecem nenhum serviço crítico ou suporta competências chave. Esses sistemas geralmente são sistemas que não desempenham as funções vitais para a empresa e podem ser substituídos por componentes prontos de fornecedores que oferecem todo um suporte e tiram a responsabilidades dos mesmos da área de TI da organização. Esses tipos de sistemas podem incluir sistemas utilizados para pagamentos ou recursos humanos, por exemplo. (SOMMERVILLE, 2011) afirma que manter esses sistemas em operação pode ser caro e o nível de retorno para o negócio é relativamente baixo, então esses sistemas devem ser sucateados.

Em (SEACORD, 2003), o autor afirma que sistemas que se enquadrem no caso de baixo valor para o negócio e alta qualidade técnica não devem exigir algum esforço de reengenharia, modernização ou substituição. (SOMMERVILLE, 2011) afirma que eles podem ser sistemas que não contribuem muito para o negócio mas que não devem ser caros para se manter. Não vale a pena substituir esses sistemas enquanto a manutenção pode ser continuada, se mudanças muito caras não são necessárias no sistema e o seu hardware permanecer em uso. Se mudanças muito caras se tornarem necessárias, o sistema pode ser substituído por algum componente comercial.

Sistemas que se enquadrarem no caso de possuírem um alto valor para o negócio e alta qualidade técnica devem ser mantidos em operação e a manutenção do sistema pode ser continuada, segundo (SOMMERVILLE, 2011). A alta qualidade técnica significa que não é necessário pensar em algum esforço de reengenharia ou modernização no momento. Em (SEACORD, 2003) o autor afirma, porém, que esses sistemas devem ser ativamente evoluídos, utilizando práticas de desenvolvimento evolucionário. Essas práticas devem ser suportadas durante a sua fase de manutenção, para evitar que a qualidade técnica que o sistema possui seja perdida e eles se tornem candidatos a modernização.

Segundo (SEACORD, 2003), sistemas que se enquadrem no caso de possuírem um alto valor para o negócio e baixa qualidade técnica são os melhores candidatos para a modernização ou substituição. (SOMMERVILLE, 2011) afirma que esses sistemas estão executando uma função importante para o negócio então ele não pode ser descontinuado. Entretanto, a sua baixa qualidade técnica indica que eles devem ser caros de manter. Esses

sistemas devem ser modernizados para melhorar a sua qualidade, reduzindo os custos de manutenção e entregando novas funcionalidades para o negócio.

Como vimos, a prioridade dos sistemas a serem modernizados muito provavelmente será encabeçada por aqueles que tiverem um alto valor para o negócio, porém com uma qualidade técnica reduzida. Esse sistemas devem ser importantes pois eles comportam processos de negócios vitais para a empresa e não podem parar, são funcionais e amplamente testados e comportam todo o conhecimento embutido ao longo do seu ciclo de vida em operação, então qualquer esforço de modernização num sistema deste tipo acarreta um grande risco.

Chegar à conclusão de qual tipo esforço de modernização é o mais adequado a ser submetido cada sistema não é uma tarefa fácil, e a sua escolha deve envolver e ser baseada no resultado de uma grande análise de todo o risco e custo que cada estratégia representa.

3.8 MODERNIZAÇÂO COM GERENCIAMENTO DE RISCOS

Em (SEACORD, 2003) o autor apresenta uma abordagem para ajudar a definir qual a estratégia de modernização mais adequada para o sistema fazendo uma constante avaliação de riscos, das estratégias de modernização e até estimativa de recursos, para então chegar num plano de modernização definido para cada sistema. Esta abordagem é chamada de “Risk-Managed Modernization”.

O autor a define o “Risk-Managed Modernization” como uma prática de engenharia de software que continuamente avalia o que pode dar errado (os riscos), determina quais riscos são importantes e implementa estratégias para lidar com esses riscos. O “Risk-Managed Modernization” não é uma ideia nova, ele é um elemento integral do processo de desenvolvimento em espiral proposto por Boehm em (BOEHM, 1988).

Após a análise do portifolio, a estratégia do “Risk-Managed Modernization” possui alguns passos, que consistem em identificar os stakeholders, entender os requisitos dos usuários, os requisitos do sistema, os requisitos de restrições e os requisitos não-funcionais, criar o Business Case, entender o sistema legado e a tecnologia alvo, avaliar as tecnologias existentes, definir a arquitetura alvo – que consiste na arquitetura desejada para o sistema, definir a estratégia de modernização – no início o sistema todo é constituído de código legado, o que vai diminuindo na medida em que novos incrementos vão sendo realizados, reconciliar a

estratégia de modernização com os stakeholders e estimar os recursos para a estratégia de modernização. Este processo deve ser repetido em cada sistema, e no final de cada um, um plano de modernização deve ser gerado. A Figura 3 ilustra num diagrama de atividades a abordagem “Risk-Managed Modernization”.

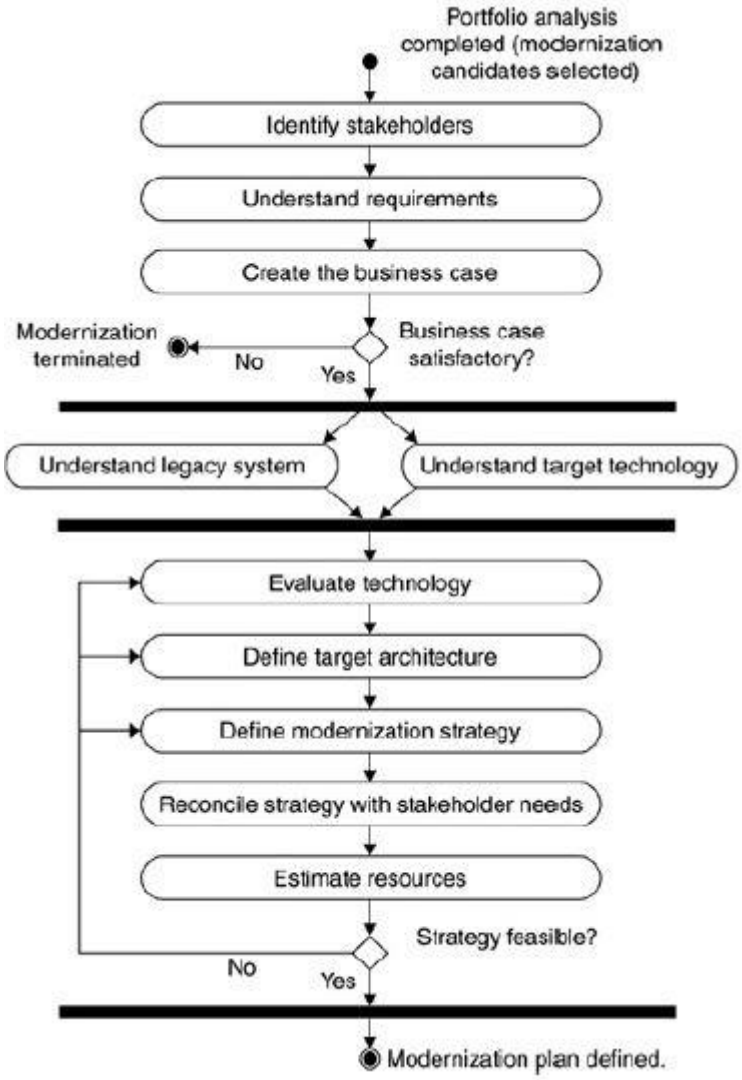

Figura 3 O processo “Risk-Managed Modernization” (SEACORD, 2003)

3.8.1 Identificando os candidatos à modernização

O primeiro passo, a análise do portifolio, é a entrada para todo este processo, ela consiste em levantar quais são os sistemas legados da empresa e fazer uma avaliação criteriosa neles e

classifica-los dado o momento atual do seu ciclo de vida, qualidade técnica e importância para o negócio. Esta análise deve identificar quais são os sistemas candidatos a modernização.

Então é necessário identificar os stakeholders que são relevantes para o sistema. Desenvolvedores, testadores, a equipe de manutenção, administradores do sistema, clientes, fornecedores, patrocinadores, usuários finais, arquitetos e representantes que interagem com o sistema são stakeholders: eles tem interesse em um projeto deste. Essas pessoas irão julgar os impactos e benefícios de um projeto de modernização, e é importante obter o seu aceite e suporte. Porém este é um desafio ao mesmo tempo, pois diferentes stakeholders podem ter diferentes pontos de vista em termos de visualizarem riscos e possuírem interesses de uma maneiras distintas.

3.8.2 Entendendo os requisitos

Entender os requisitos pode ser uma das tarefas mais difíceis no desenvolvimento de software e em um projeto de modernização, e em (SEACORD, 2003) o autor sugere que eles sejam divididos em quatro categorias: requisitos de usuários, requisitos de sistema, requisitos de limite e requisitos não-funcionais.

- Requisitos de usuários são as capacidades que devem ser fornecidas pelo sistema, esses requisitos geralmente estão expressos como tarefas ou atividades que devem ser suportadas pelo sistema. Essas capacidades são os processos de negócio que o usuário executa dentro do sistema para cumprir alguma tarefa, são as funcionalidades que atendem as necessidades dos usuários.
- Requisitos de sistema são requisitos que descrevem capacidades do sistema, e o sistema por si só, em alto nível eles seriam as capacidades que o sistema deve ter para atender os processos de negócio que ele deve suportar.
- Requisitos de limite incluem decisões que foram tomadas e que afetem de alguma maneira o sistema como interações com outros sistemas, padrões de desenvolvimento, ambiente e custo. Esses requisitos de limite irão impactar fortemente no resultado da avaliação do sistema, pois esses limites estabelecidos para o sistema atual podem fortemente influenciar na estratégia de modernização que será adotada, assim como no projeto em si, fazendo com

que a arquitetura do novo sistema modernizado seja altamente influenciada pela arquitetura existente no legado.

- Requisitos não-funcionais incluem capacidades comportamentais como desempenho, usabilidade, segurança e outros requisitos que não se enquadrem no conceito de funcionalidades para o usuário que o sistema deverá ter.

Assim como em qualquer projeto de software os requisitos para um projeto de modernização devem ser completos, consistentes, entendíveis e válidos. Neste momento é possível obter a percepção dos usuários com a qualidade como um todo do sistema atual, uma vez que eles podem desejar muitos requisitos que o sistema possui atualmente justamente pelo fato dele poder estar impactando alguma melhora nos processos de negócio da organização, e isto acaba contribuindo para uma avaliação do sistema em geral.

Após o entendimento dos requisitos, então deve ser criado um Business Case, ou Plano de Negócio. O Business Case é o documento que suporta a tomada de decisões e o planejamento. Como projetos de modernização tomam muitos recursos, este documento pode ajudar a convencer a gestão de que o projeto é financeiramente viável, sendo chave para obter financiamento para o projeto e aprovação.

Um bom Business Case deve considerar diversas abordagens de modernização e contingência diferentes, assim como uma boa justificativa técnica e econômica para a seleção de uma abordagem em particular.

Em geral ele deve fornecer informações sobre o propósito do projeto e seus objetivos, uma descrição do sistema atual e dos processos de negócio atuais que ele suporta, uma descrição do sistema futuro e dos processos de negócio futuros, uma estimativa de custos, uma análise de custo-benefício, uma avaliação de riscos, uma análise de mudanças e medidas de desempenho. A análise de mudanças inclui mudanças em termos de pessoal, equipamentos, software, hardware e suporte (SEACORD, 2003).

3.8.3 Validando a solução proposta

Na abordagem “Risk-Managed Modernization” após a criação do Business Case deve ser feita uma avaliação para verificar se ele é satisfatório, ou seja, se a proposta ou as propostas de abordagem de modernização para o sistema é válida em termos de custo, risco e esforço em

geral, devem ser avaliadas quais são as alternativas para modernizar o sistema atual e, caso ele seja válido, o projeto de modernização continua para os próximos passos. Se com a avaliação do Business Case constatarmos que alguma modernização é inviável, o projeto de modernização deste sistema para por aqui, passando a repetir todos esses passos para o próximo sistema da lista no portifolio de sistemas da empresa.

Se o Business Case for satisfatório e a solução proposta por ele for viável, de acordo com o "Risk-Managed Modernization" devemos então ter duas atividades em paralelo: entender o sistema legado e entender a tecnologia alvo. Essas atividades são predecessoras dos próximos passos que tratam do desenho e avaliação da solução em si pois são fundamentais neste processo de modernização, uma vez que a arquitetura e o desenho do novo sistema deverá ser fortemente influenciada pelo desenho e pela arquitetura do legado.

3.8.4 Entendendo o sistema legado

Entender o sistema legado e o seu contexto é essencial para o esforço de modernização. O desafio aqui é conseguir isto de uma maneira eficiente, barata e rápida. Algumas técnicas que podem ser utilizadas neste entendimento incluem a engenharia reversa, entendimento do programa e reconstrução da arquitetura. Adiante, discutiremos algumas técnicas para auxiliar o entendimento de legado e criar abstrações e representações do sistema em mais alto nível.

Entender a tecnologia alvo é um passo extremamente importante, pois devemos evitar que de tecnologias estado da arte ou práticas que são disconectas da realidade não sejam definidas sem antes se fazer uma avaliação da tecnologia que o legado foi desenvolvido, pois um projeto de modernização pode falhar se escolhermos apenas a tecnologia e não pensarmos nos custos, riscos ou o prazo para implantação do sistema que esta decisão pode acarretar. É importante entender tanto a tecnologia utilizada no legado quanto aquelas que podem ser utilizadas no esforço de modernização.

Segundo (SEACORD, 2003) existem três classes de tecnologia de sistemas que são de interesse numa modernização de um sistema legado e podem ser levantadas e avaliadas:

- As tecnologias utilizadas para construir o sistema legado, incluindo as tecnologias de linguagem de programação e sistemas de banco de dados.

- As tecnologias mais modernas, que representam um nirvana naquelas que foram utilizadas na criação do sistema legado e que trazem uma promessa de sistemas mais potentes, efetivos e fáceis de manter.
- As tecnologias oferecidas pelos fornecedores de sistemas legados. Essas tecnologias fornecem um caminho um pouco mais simples para uma melhora e embora possam facilitar a implementação de um novo sistema, chegam a um resultado que pode ser aceitável, porém que não é o ideal.

Após o entendimento da tecnologia do legado e das tecnologias atuais disponíveis que podem ser utilizadas no projeto de modernização e as suas capacidades, podemos compará-las e contrastá-las. Se as suas capacidades se sobrepõem, é preciso verificar qual dessas tecnologias resolvem o mesmo tipo de problema com melhor qualidade de serviço (SEACORD, 2003).

3.8.5 Definindo a arquitetura-alvo

Segundo (SEACORD, 2003), a arquitetura alvo deve representar a arquitetura desejada para o sistema, fornecendo uma visão técnica para o esforço de modernização. Ela geralmente necessita de descrições em diferentes formas e visualização e nível de granularidade e detalhe, pois deve suportar a comunicacão entre diversos tipos de stakeholders.

Em um esforço de modernização incremental, a arquitetura alvo provavelmente irá evoluir conforme os limites, restrições e funcionalidades são melhor entendidos e a tecnologia alvo utilizada para desenvolver o sistema vai ficando maturada. Entretanto, é sempre importante reavaliar e atualizar a arquitetura alvo durante o projeto de modernização.

Após a definição da arquitetura alvo, o próximo passo proposto é o de definir a estratégia de modernização. Este é um passo extremamente importante, pois é nele que toda a abordagem para a substituição do legado para o novo sistema deve ser estudada e estruturada de uma forma que minimize os riscos e atenda a organização num tempo aceitável, de forma que o legado continua a ser utilizado em paralelo ao novo sistema.

3.8.6 Definindo a estratégia de modernização

Segundo (SEACORD, 2003), a modernização de um sistema legado geralmente exige um grande esforço num projeto que pode durar anos. Como esses sistemas são cruciais para a operação das empresas, entregar o sistema modernizado de uma só vez introduz um nível inaceitável de riscos operacionais. Sendo assim, na maioria dos casos os sistemas legados são modernizados incrementalmente, conforme proposto neste trabalho, com o intuito de minimizar os riscos.

Inicialmente, nas primeiras entregas o novo sistema é composto praticamente apenas de código legado. Na medida em que cada novo incremento é completado, a porcentagem de código legado vai diminuindo. Em um momento, conforme as entregas vão acontecendo, o sistema estará todo modernizado.

Uma estratégia de modernização deve garantir que o sistema permanece totalmente funcional durante o esforço de modernização. Conforme vimos, isto deve ser atingido através da adoção de um processo em espiral, conforme proposto por Boehm em (BOEHM, 1988) com constante análise de riscos.

Par a estratégia de modernização ser efetiva ela deve definir como será toda a transformação da arquitetura do sistema legado para a arquitetura do sistema modernizado, definida nos passos anteriores.

Como um projeto de modernização pode levar certo tempo as tecnologias podem mudar, conhecimentos adicionais a respeito do sistema legado podem ser adquiridos e os requisitos dos usuários podem mudar. Uma estratégia de modernização bem feita deve acomodar todas essas possibilidades de mudanças.

Além de prever as mudanças acima, uma estratégia de modernização efetiva deve tentar na medida do possível minimizar os custos de desenvolvimento e de entrega, suportar uma previsível agenda mais agressiva, manter a qualidade dos produtos provisórios e finais, minimizar os riscos, atender as expectativas de desempenho e manter a complexidade do sistema num nível gerenciável.

A estratégia de transformação pode incluir migração do código fonte para alguma versão mais atual, migração dos bancos de dados e uma abordagem para a entrega dos incrementos. Podemos pensar na transformação do legado no novo sistema como uma componentização, uma vez que a ideia é gerar um novo sistema moderno contendo componentes independentes com uma única responsabilidade, a partir do sistema legado não estruturado.

Em (SEACORD, 2003), o autor apresenta mais um passo que pode ser opcional, que é a preparação do sistema. A preparação do sistema deve ser feita antes da transformação arquitetural e pode ser benéfica, mas apresenta alguns riscos. Na preparação do sistema o legado deve ser evoluído para um ponto que fique mais fácil de cumprir com a arquitetura desejada. O benefício é que os custos da modernização tendem a diminuir, mas como riscos temos o fato de que a preparação do sistema pode não ir como o planejado, e como um segundo risco temos o fato de que que a equipe de desenvolvimento pode se atolar e ficar presa ao código do legado.

3.8.7 Avaliando a estratégia de modernização

Na abordagem do "Risk Managed Modernization", o último passo do processo é estimar os recursos para o esforço de modernização. Uma vez que este passo estiver completo, após isto devemos ter o conhecimento do sistema legado e das suas tecnologias, da arquitetura alvo, da estratégia de modernização, da estimativa de custos e da agenda.

Segundo o autor, baseado nessas informações é que vamos avaliar se a proposta de modernização é viável ou não, e se não for satisfatória, devemos descobrir qual foi o motivo. Descobrindo o motivo, devemos então apresentar uma nova proposta de melhoria do legado atualizando o ponto que fez com que a abordagem não ficasse satisfatória e submetê-la novamente para validação da gestão da organização. Se a estratégia puder ser adotada, o plano de modernização é finalizado e executado.

Se caso a estratégia não for viável, a pergunta "por que não?" deve ser respondida. Dependendo da resposta desta pergunta, pode ser necessário repetir alguns passos do "Risk Managed Modernization". Por exemplo, se a migração do código não resultar em alguma funcionalidade sendo entregue dentro da agenda num prazo curto que seria aceitável, pode ser necessário revisar o plano até que a estratégia seja viável com alguma outra alternativa. Pode ser que a tecnologia selecionada não seja a melhor alternativa ou seja muito ambiciosa, neste caso devemos voltar ao passo de avaliar as tecnologias e fazer uma melhor escolha. Se o motivo for de que a arquitetura proposta não poder ser implementada num tempo e custo aceitáveis, uma nova opção de arquitetura alvo deve ser proposta.

3.9 OS RISCOS EM UM PROJETO DE MODERNIZAÇÃO

Segundo (ANITHA, 2012), o processo de reengenharia pode se deparar com vários tipos de riscos como engenharia de software, definindo a identificação dos riscos como uma arte. A identificação dos riscos é mais importante para uma avaliação efetiva desses riscos, análise dos riscos e o seu gerenciamento. O autor propõe um modelo de trabalho para analisar e categorizar riscos em potencial.

A proposta principal deste método é monitorar a reengenharia do software para identificar os riscos quando eles ocorrerem e categorizá-los, e então mitigá-los com a solução apropriada. Este processo irá guiar a reestruturação do software legado para reduzir os riscos e fazer ele ter um menor custo.

Quando monitoramos o projeto de uma maneira contínua, a identificação dos riscos pode ser feita num estágio inicial. Uma vez identificado um risco, o processo de mitigação é fácil pois conta com a expertise do time e eles podem evitar o problemas em operações funcionais.

Em (ANITHA, 2012), o autor afirma que a análise dos riscos tem sido um requisito padrão para o processo de transformação do legado em diversas organizações. Basicamente ele deve identificar, analisar, classificar e categorizar os riscos envolvidos em cada situação. Quando os riscos são sistematicamente listados e categorizados, o processo de mitigação fica muito mais simples e economicamente mais viável.

São apresentados alguns tipos de riscos que podem qualquer projeto de reestruturação de um sistema pode encontrar. Alguns desses riscos seguem abaixo:

- Missão e objetivos da organização
- Gerenciamento da organização
- Os parâmetros do projeto
- O conteúdo do produto
- A entrega do novo sistema
- O ambiente de desenvolvimento
- O time do projeto
- As tecnologias
- A manutenção

3.9.1 Processo de mitigação de riscos

A abordagem para a mitigação dos riscos proposta pelo autor em [ClassificationRisks] consiste tem quatro estágios, conforme ilustrado na Figura 4: Classificação, Monitoramento, Identificação, Avaliação e Mitigação.

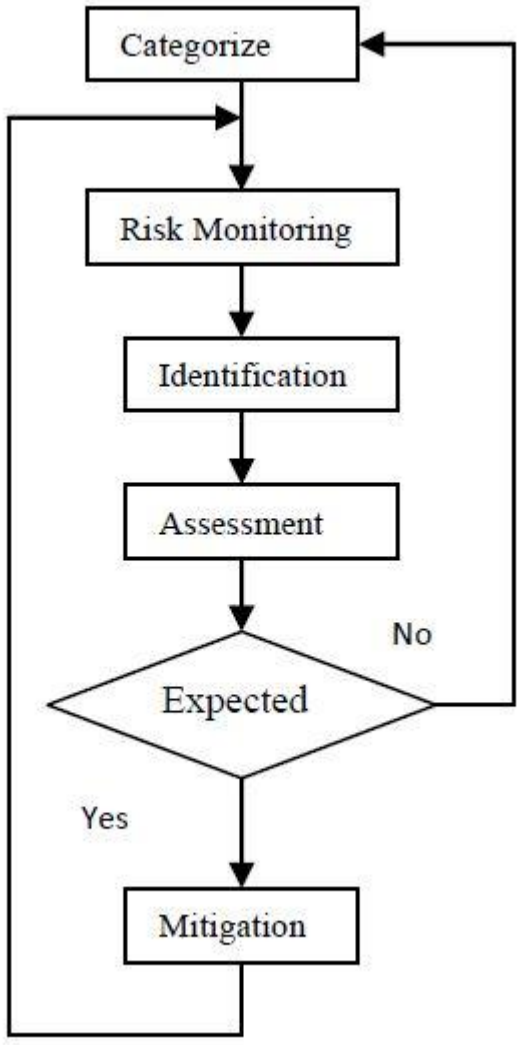

Figura 4: Abordagem para mitigação dos riscos (ANITHA, 2012)

Neste processo, a constante monitoração dos riscos identificando um novo risco para o projeto deve então avaliá-lo e então caso ele já seja um risco esperado, ou seja, caso já tenha sido identificado, avaliado e corretamente categorizado, o processo de mitigação definido para ele é acionado. Caso seja um risco que ainda não era esperado, deve ser iniciada a categorização

do mesmo, que vai ajudar a definir o processo que será acionado para mitiga-lo caso ele volte à tona no futuro.

3.9.2 Classificação dos riscos

A Figura 5 ilustra a categorização desses riscos que podem aparecer em um projeto de modernização de um sistema legado, proposto por (ANITHA, 2012):

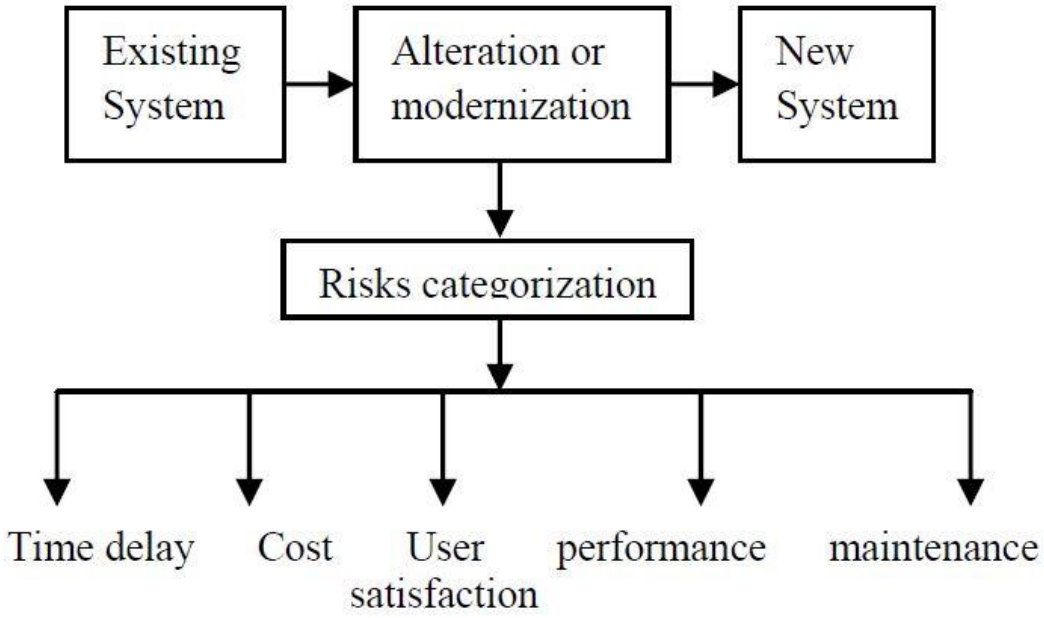

Figura 5: Categorização de riscos (ANITHA, 2012)

Podemos ver na Figura 5 que esses riscos podem ser classificados em cinco tipos, que discutiremos a seguir, conforme apresentado em (ANITHA, 2012): tempo de atraso, custo, satisfação do usuário, desempenho e manutenção.

- Tempo de atraso: o tempo é o fator de risco principal em qualquer projeto. O tempo é necessário para antecipar e prevenir os problemas. Ao administrar os riscos o tempo deve ser usado para uma vantagem, ao invés de ser desperdiçado. Os riscos relacionados ao tempo são mostrados abaixo:

- Falta de conhecimento sobre os sistemas legados

- Falta de compreensão dos conceitos do sistema legado
- Razões políticas
- Entregar o projeto no prazo
- Estouro de custos previstos no orçamento
- A migração de pessoas da equipe

Custo-benefício: qualquer projeto de reestruturação é feito para que as necessidades do negócio sejam atendidas de uma maneira melhor do que atualmente pelo legado, fazendo uso de novas tendências e tecnologias para torná-lo mais rentável. Abaixo seguem alguns riscos pertinentes ao custo do projeto:

- Baixo benefício/receita
- Alto custo de manutenção
- Backup caro
- Baixa qualidade e inconsistência dos planos do negócio
- Perca dos investimentos no legado

- Desempenho: os riscos de desempenho é o grau de incerteza se no processo de desenvolvimento e de entrega do sistema modernizado ele vai continuar sendo adequado para uso, atendendo as suas especificações técnicas, ou se o seu uso pretendido vai ser impactado pelo esforço de reengenharia. Os projetos de reengenharia depender fortemente do desempenho do novo sistema. O desempenho do novo sistema deve ser definitivamente melhor do que o desempenho do sistema legado. Abaixo seguem alguns riscos pertinentes quanto ao desenvolvimento do sistema modernizado:

- Não portabilidade
- Falta de acurácia nos resultados
- Incompatibilidade de resultados
- Seleção de uma abordagem de reengenharia inadequada
- Reestruturação dos dados inapropriada

Satisfação do cliente/usuários: em um mercado de trabalho competitivo onde o negócio compete por novos clientes, a satisfação dos usuários é vista como um diferencial e aumentá-la torna-se cada vez mais o ponto chave para o sucesso de qualquer estratégia de negócio. Para isto, as organizações necessitam de medidas confiáveis e representativas da satisfação dos usuários. Abaixo seguem alguns riscos que podem afetar a satisfação do cliente/usuário num projeto de reengenharia:

- Estouro de orçamento
- Resultados não esperados
- Desamparo dos usuários

- Manutenção: a manutenção é definitivamente um desenvolvimento evolutivo e as decisões da manutenção são auxiliadas por entender o que acontece com todos os sistemas de software com o tempo. A manutenção é quem guiará todos os novos desenvolvimentos que serão feitos no novo sistema reestruturado, e os riscos de como essas atividades deverão ser feitas deve ser bem avaliados. A chave para os problemas da manutenção podem ser tanto gerenciais quanto técnicos.

Alguns problemas gerenciais chave da manutenção são:

- Alinhamento com as prioridades dos usuários
- Gerenciamento das expectativas do negócio
- Pessoal
- Qual organização faz a manutenção
- Estimar custos

Alguns problemas técnicos que são chaves da manutenção são:

- Entendimento limitado do sistema
- Análise de impactos falha
- Testes
- Métricas de manutenibilidade

As tarefas de manutenção são classificadas em quatro classes diferentes:

- Adaptativa: a manutenção adaptativa lida com mudanças para adaptar o sistema ao seu ambiente
- Perfectiva: é responsável por acomodar novos requisitos, ou entregar requisitos alterados para o negócio, concentrada em trazer benefícios funcionais ao software legado
- Corretiva: lida com os erros encontrados e as correções deles
- Preventiva: concentra as atividades que visam melhorar a manutenibilidade do software, tornando-o mais amigável a mudanças futuras e prevenir problemas futuros

Os fatores de riscos associados a qualquer dessas atividades de manutenção são:

- Backup
- Recuperação de sistemas legados
- Reestruturação de dados imprópria
- Redocumentação imprópria
- Solução de um sistema alternativo

Os riscos apresentados acima que podem ocorrer em um projeto de reengenharia são tabulados pelo autor em (ANITHA, 2012) de acordo com a frente que eles podem afetar durante o ciclo de vida do desenvolvimento do software: as pessoas, o projeto, o processo e o produto. Esses pilares e os riscos associados a eles estão apresentados na Tabela 3, que segue abaixo.

Riscos Identificados	**Pessoas**	**Projeto**	**Processo**	**Produto**
Satisfação dos usuários	Comunicação	Tecnologia	Degradar	Sistema alternativo
Custo-benefício	Alto salário	Estouro de orçamento	Baixa qualidade	Perca do produto

Desempenho	Arriscar	Entrega	Incerteza	Backup
Tempo de atraso	Incapaz	Missão e objetivos	Tempo de envio	Tempo de entrega
Manutenção	Migração	Melhoria	Incompatibilidade	Confiança

Tabela 3 – Riscos em um projeto de engenharia para a frente (ANITHA, 2012)

A avaliação dos riscos deve ajudar a guiar o plano de modernização de cada sistema legado da organização, e dentro doo projeto de modernização de cada um deles este processo constante de identificação, avaliação, categorização e mitigação deve ser repetido a cada iteração quando uma nova parte do sistema modernizado estiver sendo planejada para ser entregue.

Com a estratégia de modernização dos sistemas em mãos, podemos iniciar o projeto de modernização de acordo com a ordem de priorização definida na análise do portifolio. No próximo capítulo serão apresentadas algumas abordagens para um desenvolvimento incremental do sistema que podem ser utilizadas para modernizar os sistemas da organização de uma maneira estruturada.

4 TÉCNICAS DE MODERNIZAÇÃO

No capítulo anterior discutimos como o software durante o seu ciclo de vida atendendo as necessidades do negócio chega a um determinado estágio onde a manutenção não é mais viável e ele necessita ser reestruturado. Após uma avaliação criteriosa do portifolio de sistemas legados da organização, dado o valor para o negócio desses sistemas serem altos devido os processos que suportam e a sua estrutura estar ruim pois foi degradada com o tempo, a ação a ser tomada com eles deveria ser uma reestruturação para melhorar a manutenibilidade e entregar novas funcionalidades ao negócio.

Neste capítulo estudaremos como esses sistemas podem ser modernizados utilizando a um esforço de reengenharia e engenharia reversa para encontrar o estado da arquitetura atual do legado e desenhar uma representação em alto nível dele que servirá como base para a arquitetura do novo sistema que será desenvolvido, além de apresentar algumas técnicas para a modernização desses sistemas dentro de um processo em espiral, conforme proposto por Boehm em (BOEHM, 1988) com entregas incrementais com constante gerenciamento de riscos.

4.1 REENGENHARIA

Em (SEACORD, 2003), o autor define reengenharia como uma forma de modernização que melhora as capacidades e/ou manutenibilidade de um sistema legado introduzindo tecnologias e práticas mais modernas, sendo a reengenharia uma abordagem disciplinada para a migração de um sistema legado para um sistema evolutivo. Reengenharia é a transformação sistemática de um sistema existente num novo sistema de forma realizar algumas melhorias na qualidade como na sua operação, nas suas capacidades, funcionalidades, desempenho ou facilidade de manutenção a um custo, agenda e risco menor para o cliente.

Segundo (SOMMERVILLE, 2003), para tornar os sistemas legados mais fáceis de se manter, podemos reestruturar esses sistemas para melhorar sua estrutura e simplificar o entendimento. Esta reestruturação pode envolver esforços para redocumentar o sistema, refatorar a arquitetura do sistema, traduzir programas para uma linguagem de programação mais moderna, e modificar e atualizar a estrutura dos dados do sistema. A funcionalidade do software não é alterada, e normalmente devemos tentar evitar fazer grandes mudanças na sua arquitetura.

Em um projeto de reestruturação, a arquitetura original do sistema legado deve tentar ser preservada pois mudanças muito drásticas podem acarretar em um risco e um custo muito maior. A arquitetura do sistema legado pode ser otimizada com a modularização, separando partes comuns em componentes. Estes componentes podem expor as interfaces extraídas do legado num primeiro momento, facilitando a sua substituição no futuro.

Novas funcionalidades para atender o negócio devem ser desenvolvidas somente após e sistema estar reestruturado pois uma alteração maior implicaria um grande esforço em testes, por se tratar de um sistema crítico para os processos da organização. Esta reestruturação deve visar a gestão dos riscos ao se modificar o sistema legado. Como o sistema legado está em produção atendendo as áreas da empresa e está estável e amplamente testado, isto deve tentar ser mantido.

A reestruturação, segundo (SOMMERVILLE, 2003), resulta em dois benefícios importantes como alternativa à substituição: o seu custo e risco fica reduzido.

A reengenharia de um sistema é mais cara do que a manutenção convencional e menos atrativa no ponto de vista técnico do que a substituição completa de um sistema, segundo (SEACORD, 2003). Como existem diversos sistemas legados numa organização, uma substituição completa de todos os sistemas seria inviável financeiramente, então a reengenharia oferece uma maneira viável em termos de custos para estender o tempo útil do ciclo de vida desses sistemas.

Em (SEACORD, 2003), o autor apresenta seis possibilidades para a reengenharia de um sistema legado: redirecionamento, renovação, uso de componentes comerciais, tradução do código-fonte, redução de código e transformação funcional.

As transformações realizadas no código fonte geralmente podem ser mais associadas às tarefas de manutenção em si do que a algum esforço de reengenharia. Ela pode ser pequena demais para conseguir melhorar a estrutura do código legado e pode ocorrer do código final gerado ser mais difícil ainda de se manter.

- O redirecionamento consiste em migrar um sistema legado para uma nova plataforma de hardware. Migrar um sistema para uma nova plataforma de hardware geralmente diminui os gastos operacionais e com manutenção, e fornece uma plataforma evolutiva para outros esforços de modernização.

- A renovação consiste em substituir toda a interface com o usuário do sistema, uma vez que ela é a parte mais visível do mesmo. Geralmente a opção por esta abordagem é feita quando a organização pensa em melhorar a usabilidade de uma aplicação web. Numa perspectiva do sistema legado em si o sistema todo continua sendo praticamente o mesmo, uma vez que ele continuará sendo inflexível e difícil de manter pois essas novas telas ainda dependem fortemente do legado. Porém para os usuários a modernização terá sempre sido um sucesso, pois agora o sistema oferece uma interface nova e moderna.
- O uso de componentes comerciais consiste em substituir partes do sistema legado por sistemas prontos de fornecedores que podem dar suporte ao software adquirido para cumprir a função que era feita pelo sistema legado. Esta abordagem acaba por reduzir a quantidade de código que precisa ser mantido, pois essas funções que antigamente eram cumpridas pelo legado são substituídas por novos sistemas comerciais prontos.
- A tradução do código fonte consiste em converter o sistema escrito numa linguagem de programação muito antiga em uma linguagem mais moderna, para usufruir de capacidades de hardware e software que a adoção de uma plataforma mais moderna pode oferecer. A tradução do código fonte pode ser feita de uma forma automática, através do uso de ferramentas específicas a este fim.
- A redução de código é conceitualmente simples, mas a sua execução pode ser um pouco mais complexa. Ela consistem basicamente em eliminar o código fonte desnecessário antes de migrar o sistema para uma outra plataforma ou linguagem mais moderna. Ela pode se tornar difícil pois para jogarmos código fora com segurança deve ser feito um grande trabalho de mapear todas as dependências e se certificar que nenhuma outra parte do sistema irá parar de funcionar.
- A transformação funcional inclui melhorias na estrutura do legado, modularização do sistema e reengenharia de dados. A estrutura do legado pode ser melhorada identificando pontos que existam códigos duplicados dentro do sistema e com isto fazer um trabalho de refatoração, por exemplo. A modularização do sistema deve visar consolidar partes relacionadas do sistema

> em módulos comuns que podem ser reutilizados, tornando-se um passo preliminar para uma modernização incremental, uma vez que módulos bem definidos facilitam a substituição por novos componentes. A reengenharia dos dados envolve modificar o armazenamento, organização e o formato dos dados que são processados pelo sistema legado, sendo geralmente necessária quando a estrutura dos dados está degradada e eles são armazenados com inconsistências em múltiplos locais, e precisam ser consolidados. Para uma transformação funcional se dar com sucesso é necessário entender o sistema ao menos no nível de suas interfaces, quando elas e o funcionamento interno do sistema não tiverem sido entendidos uma alternativa pode ser envolver esse sistema numa nova camada para usá-las em um novo contexto que fornecerá as interfaces desejadas de uma maneira mais moderna.

Apesar da reengenharia de software ajudar a melhorar um sistema em diversos pontos ela tem alguns limites práticos, segundo (SOMMERVILLE, 2011). Mudanças maiores na estrutura e na arquitetura do sistema devem ser evitadas pois acarretaria em um esforço muito grande, tornando-se caras. Apesar da reengenharia conseguir melhorar o entendimento do sistema e a sua manutenibilidade, o sistema reestruturado provavelmente não irá ser de tão fácil manutenção quanto um novo sistema que seria desenvolvido utilizando métodos modernos de engenharia de software.

Segundo (DEMEYER, 2008) praticamente qualquer atividade de reengenharia deve começar com alguma engenharia reversa, desde que não seja possível confiar na documentação existente do sistema, por exemplo, caso ela esteja desatualizada – e se houver alguma. Basicamente devemos analisar o código-fonte do legado, rodar o sistema e entrevistar seus usuários e desenvolvedores para entender como eles utilizam o sistema e então construir um modelo do sistema legado. Com isto, devemos determinar quais são os obstáculos para um progresso maior, avaliar os riscos, e consertá-los.

Essa é a essência da reengenharia, que busca transformar um sistema legado no novo sistema que você teria construído se pudesse ter a oportunidade de voltar no tempo e conhecer todos os novos requisitos e problemas encontrados que foram sendo conhecidos com o tempo, modificando o sistema durante seu ciclo de vida até chegar no estado atual. Porém, como não

podemos nos dar ao luxo de reconstruir tudo a não ser assumindo um alto custo e risco, devemos cortar esses custos reestruturando apenas as partes mais críticas.

A reengenharia, segundo (DEMEYER, 2008), está concentrada em reestruturar o sistema geralmente para corrigir problemas reais ou conhecidos, mas mais especificamente em preparar o sistema para um desenvolvimento maior futuro e extensão. Ela deve visar resolver os problemas mais críticos do sistema legado, reduzindo custos de manutenção, para dar a oportunidade da organização poder pensar no futuro, considerando um esforço maior para a sua substituição.

Um objetivo a ser alcançado num projeto de reengenharia deve ser tentar descobrir como e por que o sistema legado evoluir durante o tempo para o seu estado atual. Em particular, podemos tentar entender mais especificamente quais partes do sistema tiveram que ser fortemente refatoradas, quais partes sofreram muitas mudanças ou correções de erros durante as atividades de manutenção e quais partes se manteram estáveis durante o tempo (DEMEYER, 2008). Essas partes mais críticas do sistema, então, são as que devem ser reestruturadas.

Num projeto de reengenharia, as partes do software que sempre se manteram estáveis com o tempo e foram pouco modificadas podem permanecer como estão, sem nenhuma necessidade de reestruturação, não importando como está a qualidade do código dentro dela. O alvo da reestruturação deve ser resolver os problemas das partes mais críticas do sistema legado, não devemos alterar muito as outras partes neste primeiro momento, se alguma mudança for necessária pode ser por motivos apenas de modularização.

Segundo (DEMEYER, 2008), o software que as organizações utilizam para suportar seus processos de negócio devem ser adaptáveis, para facilitar a implementação das novas necessidades do negócio que mudam constantemente. Muitos sistemas legados, principalmente os de baixo valor para o negócio, podem ser eventualmente jogados fora quando não servirem mais para o propósito inicial no qual eles visavam atender quando foram criados ou então podem ser substituídos por sistemas completamente novos de fornecedores externos, não necessitando mais ficar nas mãos da organização. Mas um sistema legado de alto valor para o negócio não pode ser substituído ou melhorado exceto através de um alto custo.

O alvo da reengenharia desses sistemas legados deve ser reduzir a sua complexidade suficientemente até um nível que ele possa continuar a ser utilizado por um tempo, e fazer com que ele possa ser continuar sendo melhorado e adaptado a um custo mais aceitável. Este é o ponto chave que vai fazer a organização poder pensar para a frente, numa constante cultura de

reengenharia. A cultura de reengenharia contínua é pré-requisito para atingir um sistema orientado objetos flexível e de fácil manutenção. Esta cultura de reengenharia possui um ciclo de vida que é composto também da engenharia reversa e a engenharia para a frente.

4.2 ENGENHARIA REVERSA

Todo projeto de reengenharia deve ser iniciado por um trabalho de engenharia reversa, pois ela é quem será a responsável por extrair uma abstração do sistema legado com base nos levantamentos do portifolio existente e a recriação da documentação e do desenho da arquitetura desses sistemas, servindo como base para as melhorias futuras.

Segundo (PRESSMAN, 2010), o processo de engenharia reversa deve ser capaz de derivar representações do sistema legado em diversos níveis de abstração, dependendo do propósito da informação que se deseja extrais. Ela deve geralmente obter desenhos processuais da arquitetura geral da aplicação (em mais baixo nível), informações sobre a estrutura do programa e dos dados em si (num nível de abstração um pouco maior), modelos de objetos, dados e/ou modelos de controle de fluxo (num nível de abstração relativamente alto) e modelos de relacionamento entre as entidades (num alto nível de abstração). Conforme o nível de abstração aumenta, são fornecidas informações que vão facilitando o entendimento do sistema.

A engenharia reversa de um sistema legado é o processo de analisar os programadas num esforço para criar uma representação do programa a um nível de abstração mais alto do que o código fonte. A engenharia reversa é um processo para a recuperação do desenho da arquitetura de uma aplicação. As ferramentas para engenharia reversa são capazes de extrair dados, desenho da arquitetura e informação dos processos de negócio que o sistema legado existente suporta.

As atividades de extrair essas abstrações do sistema legado são o grande alvo da engenharia reversa, segundo (PRESSMAN, 2010). Devemos avaliar todo o programa legado e a partir do seu código-fonte (geralmente não documentado) e se houver alguma documentação existente, desenvolver uma especificação significativa do processamento que é desempenhado por esse sistema, as interfaces com o usuário que são aplicadas e a estrutura dos dados ou do banco de dados que é utilizado.

A engenharia reversa está concentrada em entender as estruturas do sistema e como o ele funciona, entender suas interfaces, entidades e dados, e entender como eles se relacionam

para processarem as informações e executarem as ações que são responsáveis pelos processos de negócio da organização.

Segundo (DEMEYER, 2008), nós recorremos à engenharia reversa quando precisamos entender a maneira como alguma coisa realmente funciona. Normalmente, será necessário aplicar a engenharia reversa somente na parte do software que pretendemos consertar, substituir ou estender. Algumas vezes pode ser necessário aplicar a engenharia reversa em alguma parte do software para inclusive entender como utilizá-la, mas isto é um sinal de que esta parte talvez necessite também de alguma reengenharia.

Como consequência, os esforços da engenharia reversa de um sistema legado tipicamente focam em redocumentar o software e identificar potenciais problemas, no intuito de prepará-lo para a reengenharia. Em (DEMEYER, 2008) o autor sugere o uso de diferentes fontes de informação enquanto a engenharia reversa estiver sendo feita, como:

- Ler e entender a documentação existente
- Ler e entender o código-fonte
- Rodar o software, de preferência em modo debug
- Entrevistar usuários e desenvolvedores
- Codificar e executar casos de teste
- Gerar e analisar rotas
- Utilizar várias ferramentas para gerar visões em alto nível do código-fonte e das rotas
- Analisar o histórico de versões

Conforme essas atividades vão sendo desempenhadas, os modelos do software legado construídos vão sendo progressivamente refinados, mantendo o controle de várias perguntas e respostas que forem surgindo, cada vez mais limpando as incertezas numa documentação mais técnica.

Em (DEMEYER, 2008), o autor propõe uma abordagem para a engenharia reversa dos sistemas legados de uma organização que consiste em levantar quais são os sistemas que necessitam de uma reengenharia e estabelecer uma prioridade entre eles, conforme discutido no capítulo anterior, técnicas para o primeiro contato com o sistema e para o entendimento inicial,

assim como uma proposta para ajudar a capturar um modelo detalhado do legado, que serão apresentados a seguir.

Entrevistar a equipe de manutenção: as entrevistas com a equipe de manutenção do sistema podem ser úteis para aprendermos sobre todo o contexto histórico o político sobre o projeto de modernização. Algumas perguntas podem ser feitas para ajudar a entender o contexto e o sistema em geral, como:

- Qual foi o bug mais difícil que foi corrigido no último mês? E qual foi o mais fácil? Quanto tempo foi gasto para corrigir eles? O que faz um ser mais difícil do que o outro?
- Como o time de manutenção coleta relatórios de bugs e solicitações de melhorias? Quem decide a prioridade entre eles? Existe algum software que gerencia as versões dos códigos?
- O quão bom está o código? O quão confiável é a documentação?

Essas perguntas podem ajudar a ganhar a confiança da equipe de manutenção e obter uma sensação dos motivos que fizeram a necessidade de reengenharia deste sistema vir à tona, que serão úteis para se entender o contexto geral do projeto.

Ler o código-fonte em uma hora: uma leitura rápida do código-fonte do sistema pode ser útil para avaliar de maneira breve o estado atual do sistema. Após esta leitura do código-fonte, um pequeno relatório poderá ser produzido, contendo:

- Uma avaliação geral se a reengenharia neste sistema parece ser viável ou não, e porque
- Entidades que parecem ser importantes, por exemplo, classes ou pacotes
- Locais onde o código-fonte não estiver bem escrito
- Partes que devem ser melhor investigadas mais tarde

Este primeiro contato com o código-fonte do sistema deve ser breve, mas é ele quem começará a dar a visão de como o sistema em si está construído, algumas dificuldades que podem atrapalhar o projeto de reengenharia em determinadas partes, além de ajudar a obter o conhecimento dos padrões de codificação existentes no sistema.

Verificar a documentação: se houver alguma, avaliar a documentação existente de uma maneira rápida, como feito com o código-fonte. Deve ser avaliada a relevância da documentação para este projeto, se a mesma está confiável e atualizada, e se reflete o estado atual do sistema. Esta avaliação da documentação pode ajudar a criarmos um primeiro desenho do sistema, num nível mais alto de abstração.

Entrevista durante uma demonstração: para ajudar a entender como os usuários utilizam o sistema e as suas particularidades, uma entrevista com alguns deles enquanto mostram como utilizam o sistema pode ser necessária. Esta entrevista deve ajudar a entender alguns pontos como alguns cenários típicos de uso, as funcionalidades principais que os usuários apreciam e aquelas que não são tão utilizadas, assim como alguns componentes do sistema e suas responsabilidades. Essas entrevistas podem ser feitas com diferentes pessoas da organização a fim de obter a sensação de como eles utilizam o sistema através de diversos pontos de vista, por exemplo:

- Um usuário final ajudará a entender como o sistema é utilizado para executar as tarefas do dia-a-dia
- Um gerente deverá informar como o sistema se encaixa dentro do domínio do negócio da organização
- Uma pessoa da equipe de help-desk pode ajudar a mostrar quais recursos do sistema causam mais problemas
- Uma pessoa da equipe de manutenção pode mostrar alguns subsistemas, e como eles se comunicam com outros subsistemas.

Simular uma instalação: após as entrevistas serem feitas e um entendimento inicial do contexto geral da aplicação ser identificado, podemos obter o código-fonte e instalá-lo, compilando-o e executando o código. A ideia principal neste ponto é verificar como é o processo de construção e instalação do sistema, assim como identificar possíveis problemas e dúvidas a respeito de como o mesmo é instalado que serão respondidas no futuro.

Após todas as entrevistas e um entendimento geral do sistema tiver sido obtido, podemos ter outras atividades para conseguir aprofundar os entendimentos e conseguir chegar a um nível de abstração suficiente para conseguirmos iniciar alguns desenhos do estado atual

do sistema legado. Uma dessas atividades pode ser a de estudar os dados que são persisidos pelo sistema.

Analisar os dados que são persistidos no banco de dados pode ajudar a entender quais objetos ou tipos de dados são de valor para o sistema. Essa avaliação deverá ajudar a identificar quais são as estruturas principais que representam informações de alto valor para o sistema e identificar as suas relações e responsabilidades, ajudando a gerar um primeiro desenho em mais alto nível das estruturas do sistema legado.

Com base nas informações obtidas nesta fase podemos apontar algumas oportunidades de melhorias e os riscos que foram encontrados, por exemplo, você pode identificar a presença de documentação atualizada como uma oportunidade e a falta de testes como um risco, então podemos planejar uma atividade de gerar os testes com base na documentação durante o projeto de reengenharia.

Para a montagem do plano de projeto e identificação dos problemas é necessário primeiramente conversar com o time de manutenção para aprender sobre o contexto histórico e político do projeto. Eles devem ser tratados como "irmãos em armas" pois é importante uma boa relação e suas informações serão valiosas para o andamento do projeto. Após isto, com o código-fonte em mãos podemos fazer uma leitura rápida em apenas uma hora, para avaliar o estado atual do software de uma maneira breve. Assim, estaremos aptos a produzir um pequeno relatório com as suas impressões, incluindo uma avaliação geral de se a reengenharia parece viável ou não, entidades importantes ou partes que devem ser melhor investigadas. Se o código-fonte for muito extenso (maior que 10 kloc) podemos estabelecer um critério para dividí-lo de uma maneira que essa leitura seja feita em uma série de sessões. Depois desta leitura e avaliação do código-fonte podemos novamente conversar com o time de manutenção.

Após a leitura do código-fonte, deve ser avaliada a relevância da documentação existente lendo-a em um pequeno intervalo de tempo. Esta atividade é útil para fazer uma avaliação se a documentação será útil ou não no projeto de reengenharia, então deve-se listar as partes da documentação que serão úteis e a primeira impressão do quanto a sua descrição pode estar atualizada.

Para se ter uma ideia da utilização do sistema e dos cenários, pode ser feita uma entrevista com os usuários enquanto ele faz uma demonstração do seu trabalho. Esta técnica pode ser útil para gerar um relatório sobre alguns cenários de uso e algumas funcionalidades

principais fornecidas pelo sistema, além de ser possível notar as impressões dos usuários com relação as funcionalidades do sistema atual.

Um próximo passo para entender o sistema é fazer uma instalação para simulação, onde deve ser compilado o código-fonte, pois isto deve dar um entendimento maior da instalação e da compilação do sistema.

Essas atividades de engenharia reversa propostas em (DEMEYER, 2008) devem ajudar a extrair as informações referentes ao desenho e arquitetura das aplicações a fim de ajudar a desenhar em um mais alto nível de abstração os primeiros desenhos que servirão como base para a reengenharia.

Para a engenharia reversa do sistema ter sido feita com sucesso, ela deve ter nos ajudado a (ANITHA, 2012):

- Identificar os componentes do sistema e como eles se relacionam
- Criar representações do sistema numa outra forma ou num nível maior de abstração.

4.3 ENGENHARIA PARA A FRENTE

Segundo (PRESSMAN, 2010) a reestruturação de um software modifica seu código-fonte eu/ou os seus dados num esforço para torná-lo mais amigável a mudanças futuras. Em geral, esta reestruturação não modifica a arquitetura geral dos programas. Ela tenta se concentrar nos detalhes de design de módulos individuais, e nas estruturas de dados locais definidas dentro dos módulos. Se o esforço de reestruturação se estender além das fronteiras dos módulos e abranger a arquitetura do software, a reestruturação se torna engenharia para a frente.

O processo de engenharia para a frente aplica princípios, conceitos e métodos da engenharia de software para recriar uma aplicação existente. Em muitos casos, a engenharia para a frente simplesmente não cria somente um programa mais moderno que o programa antigo equivalente. Ao invés disto, novos requisitos de usuários e novas tecnologias são integrados aos esforços de reengenharia. O programa redesenvolvido estende as capacidades da aplicação antiga existente (PRESSMAN, 2010).

Em (DEMEYER, 2008), o autor define a engenharia para a frente como o processo tradicional de se mover abstrações lógicas em alto nível para desenhos de implementação independentes da implementação física de um sistema.

A engenharia para a frente deve ser aplicada após um esforço de engenharia reversa e reengenharia, pois ela é quem estabelece a cultura de melhoria contínua que deve ser aplicada nos sistemas da organização sempre tentando melhorar a qualidade dos sistemas existentes. Com ela o software é constantemente melhorado, aplicando as técnicas e princípios da engenharia de software para guiar a sua evolução.

Em (ANITHA, 2012), o autor o processo de engenharia para a frente iniciando com a especificação do sistema, seguidos o desenho e a implementação, até chegar no novo sistema conforme Figura 6.

Figura 6: Processo da Engenharia para a frente (ANITHA, 2012)

A especificação do sistema deverá ser o resultado obtido das atividades de reengenharia e engenharia reversa que foram realizadas nos sistemas para se obter as abstrações e gerar a documentação atualizada da arquitetura, das funcionalidades e dos requisitos dos usuários que servirão como base para o processo de engenharia para a frente, que cuidará então de implementar o novo sistema.

Os esforços da engenharia para a frente deverão se concentrar no desenho e na implementação do novo sistema, aplicando novas tecnologias num processo moderno utilizando técnicas e práticas da engenharia de software para manter o sistema sempre moderno e de fácil manutenção.

Segundo (DEMEYER, 2008), como exatamente este processo de engenharia para a frente pode ou deve trabalhar é, naturalmente, um assunto de grande debate, embora a maioria das pessoas aceite que este é um processo iterativo, conforme o modelo de desenvolvimento de software em espiral de Barry Boehm (BOEHM, 1988). Neste modelo, sucessivas versões de

um software vão sendo desenvolvidas repetidamente coletando requisitos, avaliando os riscos, estruturando a nova versão do sistema e avaliando os resultados.

Essas novas versões do sistema vão sendo entregues até que todo o sistema legado tenha sido substituído por completo, após isto as novas versões conterão somente partes de código do novo sistema modernizado.

Na Figura 7, (DEMEYER, 2008) ilustra como a engenharia reversa e a reengenharia trabalham em relação a engenharia para a frente.

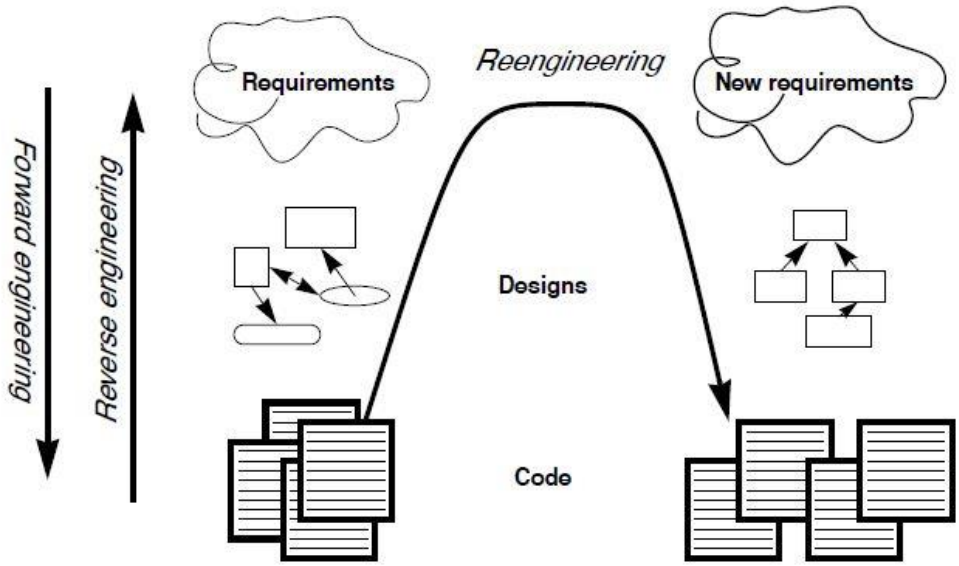

Figura 7: Engenharia reversa, Reengenharia e Engenharia para a Frente (DEMEYER, 2008)

Dentro do processo de engenharia para a frente, a engenharia reversa fica responsável por a partir do código do sistema legado, podendo fazer uso das técnicas apresentadas anteriormente, extrair o desenho da arquitetura, das funcionalidades e de como os processos de negócio são implementados dentro do sistema, construindo modelos em alto nível de abstração e artefatos a partir do código, gerando uma documentação atualizada do seu estado atual.

O processo de reengenharia completo consiste em iniciar com uma engenharia reversa para obter esta documentação e entregar um sistema modernizado, sendo assim, ela pode ser vista como o processo que transforma uma representação de baixo nível em outra, enquanto recria artefatos de maior nível de abstração durante este caminho (DEMEYER, 2008).

Segundo (DEMEYER, 2008), se a engenharia para a frente consiste em mover as visões de alto nível de requisitos e modelos para realizações concretas, então a engenharia reversa é sobre voltar atrás de uma realização concreta para modelos mais abstratos, e a reengenharia consiste em transformar implementações concretas em outras implementações concretas.

4.4 O PROCESSO DE REENGENHARIA

Em (PRESSMAN, 2010), o autor apresenta um modelo de processo para a reengenharia de um sistema legado que é composto pelas fases de análise do inventário, reestruturação da documentação, engenharia reversa, restruturação do código e reestruturação dos dados, seguido da engenharia para a frente.

Segundo (PRESSMAN, 2010) a reengenharia leva tempo, custa uma quantia significante de dinheiro e acaba absorvendo recursos que poderiam estar alocados de uma outra maneira, concentrado em outros esforços. Por todas essas razões, a reengenharia não é conseguida em poucos meses ou eventualmente em poucos anos. A reengenharia de sistemas de informação é uma atividade que deverá absorver recursos da organização por alguns anos. Por este motivo, toda organização precisa de uma estratégia pragmática para a reengenharia dos seus sistemas se dar com sucesso.

A reengenharia é um processo de reconstruir. Podemos tomar como exemplo a reconstrução de uma casa, conforme proposto por (PRESSMAN, 2010), para analisar as situações que podemos nos deparar durante esse esforço:

- Primeiramente, antes de iniciar a reconstrução devemos avaliar se realmente ela necessita ser reconstruída, avaliando o seu estado atual e possíveis alternativas. Para determinar a necessidade de reconstrução devemos contar com a ajuda de um especialista, a fim de criar uma lista com critérios sistemáticos que ajudarão a chegar no resultado final desta avaliação com base nos pontos pré-definidos que informarão qual o estado atual da mesma.
- Antes de derrubar a casa toda e começar a reconstruí-la, devemos estar cientes do quão a estrutura é fraca. Se a estrutura da casa estiver relativamente boa pode ser possível apenas remodelar, sem a necessidade de reconstruir por completo, o que resulta em um custo e um prazo muito menor do que reconstruir.

- Antes de iniciar a reconstrução, devemos tentar entender como a original foi construída. Entender as estruturas internas, olhar por trás das paredes, verificas as instalações. Mesmo que tudo seja jogado fora, o conhecimento obtido irá ajudar quando iniciar a reconstrução.
- Se a decisão for a de começar a reconstruir, utilizar somente os materiais mais modernos e mais duráveis na construção. Isto pode custar um pouco mais caro agora, mas ajudará a evitar um grande consumo de dinheiro e tempo nas manutenções futuras.
- Se a decisão for mesmo a de reconstruir, ser disciplinado e fazer uso somente de práticas modernas que irão resultar em uma alta qualidade no produto, hoje e no futuro.

Apesar dos princípios acima estarem contextualizados nos esforços de reconstruir uma casa, eles podem ser aplicados para a reengenharia de qualquer sistema. Para implementar esses princípios em um projeto de reengenharia podemos fazer o uso de um processo evolutivo em espiral, conforme proposto por Boehm em (BOEHM, 1988), onde Pressman em (PRESSMAN, 2010) baseia o seu modelo, conforme Figura 8.

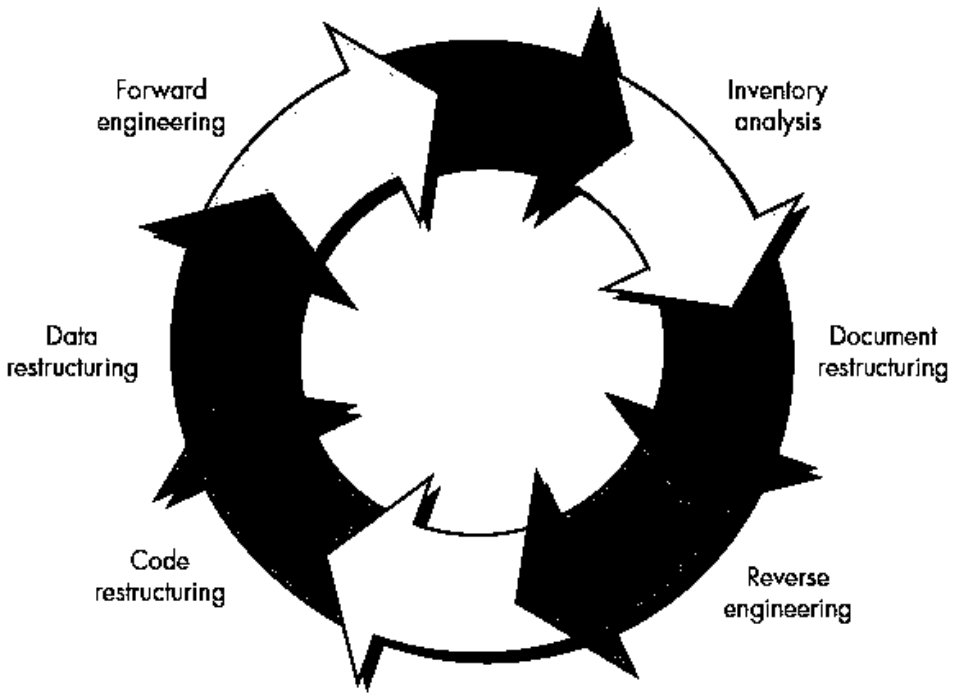

Figura 8: Processo de Reengenharia (PRESSMAN, 2010)

Abaixo descreveremos as atividades que fazem parte deste processo, lembrando que este é um processo cíclico, e em determinados ciclos pode ser que o processo termine antes de uma dessas atividades:

Análise do inventário: toda organização deve possuir um levantamento do inventário dos softwares que suportam o seu negócio, que pode ser uma simples planilha fornecendo descrições detalhadas sobre o tamanho, idade e impacto para o negócio de cada aplicação. Essas aplicações devem ser avaliadas e com base nas prioridades que serão estabelecidas o processo de reengenharia desses sistemas pode então ser iniciado. Para esta avaliação do portifolio e a classificação dos sistemas podem ser utilizadas as técnicas que foram apresentadas no capítulo anterior

Reestruturação da documentação: É aqui que a engenharia reversa será utilizada para entender como o sistema funciona internamente, gerando desenhos e visões em mais alto nível de abstração que servirão como base para as atividades de engenharia subsequentes. Documentação fraca é a marca registrada de muitos sistemas legados, quando eles possuírem alguma. Para lidar com isto, em (PRESSMAN, 2010) o autor apresenta algumas opções que podem ser avaliadas:

- Criar toda a documentação leva muito tempo. Se o sistema funciona, você pode optar por conviver com o que você tem. Em muitos casos, esta é a abordagem correta. Não é possível recriar a documentação para centenas de programas de computador. Se um determinado sistema é relativamente estável e está chegando ao fim de sua vida útil, podemos manter como está.
- A documentação necessita ser atualizada, mas a organização possui recursos limitados. Neste caso podemos utilizar uma abordagem de documentar alguma parte do sistema somente quando tocar, quando alguma atividade de reengenharia for necessária nela. Pode não ser necessário redocumentar toda uma aplicação existente. Ao invés disto, aquelas partes do sistema que estão sofrendo alguma mudança pela reengenharia devem ser completamente documentadas. Com o tempo, a coleção de documentos atualizados e úteis irá evoluindo.

- O sistema é crítico para o negócio e precisa ser completamente documentado. Mesmo neste caso, uma abordagem inteligente para a documentação é fazer o mínimo que for essencial.

Engenharia reversa: com a documentação tendo sido gerada ou atualizada, a engenharia reversa será responsável por extrair as informações de desenho do sistema que servirão como base para a reestruturação de código que será seguida. A engenharia reversa deverá ajudar a gerar os desenhos do novo modelo do sistema que será implementado neste processo de reengenharia.

Reestruturação do código: a reestruturação de código é uma tarefa comum da reengenharia, que mais especificamente se refere à refatoração de código e eventualmente de arquitetura, simplificando as estruturas internas do sistema. Alguns sistemas legados podem possuir uma arquitetura geral dos programas que o compõem relativamente sólida, mas alguns módulos podem eventualmente terem sido codificados de uma maneira que os torna dicíceis de manter, testar ou mesmo entender. Nestes casos, o código nesses nódulos deve ser reestruturado.

Reestruturação dos dados: um programa com uma arquitetura fraca nos seus dados pode se tornar difícil de adaptar e de melhorar. De fato, para muitas aplicações, a arquitetura dos seus dados tem muito mais a ver com a viabilidade desses programas no longo prazo do que o código em si que eles estão escritos. O contrário da reestruturação de código, que é feita num nível de abstração relativamente menor, a reestruturação de dados é uma atividade de reengenharia em alta escala. A arquitetura dos dados atual é dissecada, e os modelos de dados necessários são definidos. Os objetos de dados e atributos são identificados, e as estruturas de dados existentes são revistas para melhorar a qualidade. Como a arquitetura dos dados geralmente implica em uma forte influência na arquitetura dos sistemas, eventualmente pode ser que essas mudanças na arquitetura dos dados se estenda a mudanças de código.

Engenharia para a frente: Com a documentação atualizada do sistema gerada nos passos anteriores, as mudanças no código feitas para melhorar a qualidade serem entregues e seguidas pela mudança nas estruturas de dados, a engenharia para a frente cuidará de entregar as novas implementações utilizando as técnicas e métodos modernos da engenharia de software para constantemente entregar versões que irão recriar a aplicação existente contemplando novas tecnologias e funcionalidades.

4.5 TIPOS DE MODERNIZAÇÃO

Segundo (COMELLA-DORDA, 2000), um sistema legado pode ser modernizado a nível funcional, dados ou de interface com o usuário. Esta técnica de modernização a ser adotada vai depender da análise efetuada no sistema legado candidato a evolução. A seguir vamos apresentar esses tipos de modernização e discutir algumas aplicações típicas.

4.5.1 Modernização da interface com o usuário

Em um projeto de modernização da interface com o usuário a usabilidade do sistema é melhorada e ela é bem apreciada pelos usuários. Porém no ponto de vista técnico o sistema continua tendo a qualidade que sempre teve, ou seja, ele pode continuar sendo difícil de manter e eventualmente escrito numa linguagem antiga ou com técnicas obsoletas.

Neste tipo de modernização apenas a interface com o usuário é modernizada, as telas do sistema recebem um novo layout – no caso de um sistema web – que pode fornecer mais facilidades e ser mais atraente para os usuários, através da utilização de técnicas modernas para a web. As demais partes do sistema permanecem como estão, a nível lógico e estruturalmente o sistema legado permanece intacto.

Este tipo de modernização deve ser aplicado em sistemas que tenham uma boa qualidade técnica, mas que não são tão bem avaliados pelos usuários. A modernização das interfaces com o usuário do sistema dá a eles a sensação de ter um sistema totalmente novo, uma vez que as telas não são mais as mesmas e oferecem mais recursos que antigamente.

Durante o projeto de modernização, as interfaces com o usuário também podem ser entregues incrementalmente até que todas as telas do sistema tenham sido modernizadas, caso o sistema legado seja muito grande. Desta maneira eles podem ver uma rápida evolução e fornecer feedback constante, o que pode ajudar a ganhar a confiança e facilitar a aceitação do sistema modernizado.

Desta maneira podemos inclusive pensar em gerar protótipos das novas telas e disponibilizá-las para uso dos usuários, tendo as novas telas funcionando em paralelo com as telas antigas, pois elas consomem os mesmos dados e a mesma lógica do sistema existente, e

simplesmente descontinuar a tela antiga quando a nova estiver completamente estável e testada pelos usuários.

Na modernização dos dados do sistema, o sistema legado é analisado a nível dos dados que são importantes para ele e a sua estrutura de como eles são armazenados é modernizada. Este tipo de modernização deve ser adotada num sistema que já tenha uma qualidade relativamente boa, mas que a análise tenha apontado que a nível de dados o mesmo precisa ser reestruturado.

Neste tipo de modernização eventualmente pode ser necessária alguma mudança a nível de código, pois eventualmente pode ser que o sistema acesse dados de uma maneira obsoleta ou que o código que faz isto tenha sido escrito de uma maneira antiga. Ela melhora a qualidade técnica do sistema, mas do ponto de vista dos usuários ele continua sendo o mesmo.

4.5.2 Modernização dos dados do sistema

Na modernização dos dados de um sistema a sua estrutura deve ser analisada a fim de identificar os dados que são vitais para o sistema e melhorar a maneira como eles estão estruturados no banco de dados. Pode ser, por exemplo, que no sistema legado sejam encontrados diversos dados repetidos em locais diferentes, ou mesmo que a maneira que eles estão organizados não esteja mais adequada a nível lógico, ou que o acesso a dados no sistema precisa ter o seu desempenho melhorado.

Dependendo do estado atual do sistema esta abordagem de modernização pode melhorar a qualidade técnica fornecendo um melhor desempenho no acesso a dados. Eventualmente pode ser que uma modernização no código das *procedures* do banco de dados forneça este ganho de desempenho, ou mesmo a falta de utilização das mesmas pode ser um bom motivo para a adoção desta técnica de modernização.

Um outro caso de utilização desta técnica é quando os dados que são importantes para o sistema estão granulados dentro do banco de dados, e encontrar eles se torna uma tarefa custosa por ser necessária uma pesquisa em diversos locais. Neste caso, esses dados podem ser melhor acessados se uma nova estrutura de tabelas for adotada, envolvendo alguma mudança no código das classes que acessam as mesmas.

4.5.3 Modernização lógica do sistema

Na modernização lógica do sistema uma porção maior do sistema legado é modernizado, pois isto pode envolver mudanças tanto em interfaces com o usuário quanto a nível de dados. Este é o tipo de modernização que causará maiores impactos tanto no ponto de vista técnico quanto para os usuários, pois o sistema geralmente terá mudanças em sua estrutura para melhorar a sua qualidade.

Em projetos de reengenharia de sistemas legados este é o tipo de modernização que pode servir como a preparação do sistema para uma modernização maior, ou para facilitar a sua manutenção no futuro pois as suas estruturas internas devem ser melhoradas. Ela pode servir como a entrada para a cultura de engenharia para a frente que deve ser adotada.

Um sistema legado pode ser modernizado logicamente utilizando técnicas que as linguagens orientadas a objetos nos oferecem. Uma alternativa para a modernização lógica de um sistema legado é envolver o seu código numa camada separada e extrair as suas interfaces a fim de fazer uso dos dados e da lógica do legado, abstraindo a complexidade dele numa camada separada.

4.5.3.1 Encapsulamento Orientado a Objetos

Segundo (COMELLA-DORDA, 2000), o modelo conceitual do encapsulamento orientado a objetos é enganosamente simples: aplicações individuais são representadas como objetos, serviços comuns são representados como objetos e dados de negócio são representados como objetos. Na realidade, o encapsulamento orientado a objetos está longe de ser simples e envolve diversas tarefas que incluem análise do código legado, decomposição e abstração do modelo orientado a objetos.

O uso desta técnica pode implicar em um custo e um esforço menor, mas não livrará a equipe de desenvolvimento a ter contato e conhecimento do código legado. Justamente pelo motivo de necessitar de um esforço no código legado é que está técnica deve ser bem avaliada, para limitar até que ponto os esforços serão direcionados ao código legado, pois pode ser que a equipe gaste esforços demais adaptando código legado onde não teriam resultados tão positivos no momento.

Um outro ponto a ser ponderado é que fazendo o uso desta técnica não estaremos livres do código legado, ele continuará existindo dentro de uma camada separada do sistema modernizado. Como ele acessará sua lógica e os seus dados, pode ser que qualquer necessidade de mudança que implique alguma funcionalidade que essa camada fornece deverá ser realizada na camada legada. Desta maneira e equipe não estará livre de ter que realizar modificações no código legado.

4.5.3.2 Encapsulamento por componentes

O encapsulamento por componentes, segundo (COMELLA-DORDA, 2000), é muito similar ao encapsulamento orientado a objetos mas os componentes, em contraste aos objetos, devem obedecer a um modelo de componentes. Esta restrição habilita ao framework de componentes a fornecer componentes com um alto nível de qualidade.

O primeiro passo para a modularização do sistema entre esses componentes é identificar os objetos que representam o domínio da aplicação e fornecem serviços específicos ao negócio, e agrupá-los em entidades que representarão esses serviços. Nesta fase pode ser que essas entidades identificadas tenham sua lógica e seus dados espalhados em diversos pontos do sistema legado, então o segundo passo deve ser estabelecer um ponto comum de acesso a esta lógica e esses dados em uma camada que fornece esses serviços.

Neste momento pode ser necessária alguma reestruturação no código do legado, a fim de acomodar a lógica existente de uma maneira que facilite a adaptação a esta nova estrutura que será criada. Este esforço visa tentar já adequar as estruturas do sistema legado no modelo de serviços mas ainda não será o código final, visto que aqui ainda teremos somente código legado sendo mexido.

O terceiro passo é criar esses serviços em si já como parte do sistema modernizado, que acessarão esses serviços que foram reestruturados no sistema legado. Par a criação desses serviços diversas tecnologias podem ser utilizadas hoje em dia como EJB, Web Services ou REST. Esses serviços encapsularão toda a lógica e os dados que serão fornecidos pelo sistema legado.

Neste momento já devemos ter partes do sistema moduralizado, então já compreenderá código legado trabalhando em conjunto com código modernizado. A ideia é fazer os dois

sistemas funcionarem em paralelo, então o uso de alguma modernização nas interfaces com o usuário que fornecem esses serviços pode ser necessária.

Dentro do processo incremental esses serviços são disponibilizados e a cada entrega modernizando uma parte do sistema legado, até que em algum momento todo o sistema será compreendido somente de código modernizado. Esta entrega incremental ajuda a os usuários manterem a familiaridade com o sistema e facilita os testes, uma vez que existe um sistema de comparação rodando em paralelo. Quando o novo módulo e as interfaces estiverem amplamente testadas e estáveis, a parte legada é descontinuada. Isto ajuda a reduzir o risco de uma grande entrega de um sistema novo modernizado.

4.6 ENTREGA DO SISTEMA MODERNIZADO

Segundo (DEMEYER, 2008), devemos introduzir o novo sistema gradativamente, para que o novo sistema entre em uso enquanto o anterior continua sendo utilizado. Para isto é necessário envolver os usuários constantemente em cada etapa, desta maneira eles vão se sentindo envolvidos com o novo sistema e o impacto da mudança diminui.

Em (DEMEYER, 2008), o autor afirma que demonstrando resultados em incrementos regulares ajuda a ganhar a confiança e remover o ceticismo que alguns podem ter com o projeto de reengenharia. Devemos mostrar resultados pequenos mas de valor. Assim a migração para o novo sistema vai acontecendo incrementalmente, evitando a complexidade e o risco que um "big-bang" pode gerar.

Quanto mais tarde entregar o sistema mais tarde virão os feedbacks dos usuários, então devemos entregar uma primeira atualização do sistema legado o quanto antes, e ir migrando incrementalmente ao novo sistema. O processo proposto por (DEMEYER, 2008) é decompor o sistema legado em partes, que são os serviços e módulos que estamos abordando, selecionar uma dessas partes - a que implicará em um impacto maior para o negócio no momento -, efetuar a reengenharia nela, testar e entregar esta parte modernizada aos usuários. A cada iteração, se não houverem os testes pré-definidos, estes podem ser desenvolvidos. Este processo deve ser repetido iterativamente, até que todo os sistema esteja modernizado.

Se o projeto de modernização envolver uma migração de dados, este é um bom momento para ela ser desenvolvida. Com a divisão do sistema legado em módulos os dados que

eles armazenam de certa forma podem ser melhor estruturados, a fim de atender de melhor maneira o desenho desses serviços.

4.7 MIGRAÇÂO DOS DADOS DO LEGADO

Para a migração dos dados a esta nova estrutura podemos fazer o uso de uma ponte, que cuidará de atualizar os dados da base legada no novo banco de dados com o seu novo formato e que cuidará de manter esses dados rodando de forma paralela. Esta ponte está ilustrada na Figura 9.

Segundo (DEMEYER, 2008), esta ponte ajuda a transferir os dados para o novo sistema pois ela vai incrementalmente mover os dados do sistema legado para o novo, de uma forma que conforme ela lê os dados na base legada vai inserindo-o no novo formato da base modernizada caso ela ainda não exista nela. O sistema legado deve ser adaptado para efetuar as operações de escrita na forma modernizada do novo banco de dados, e quando todos os dados deste módulo estiverem migrados, e ponte pode ser removida.

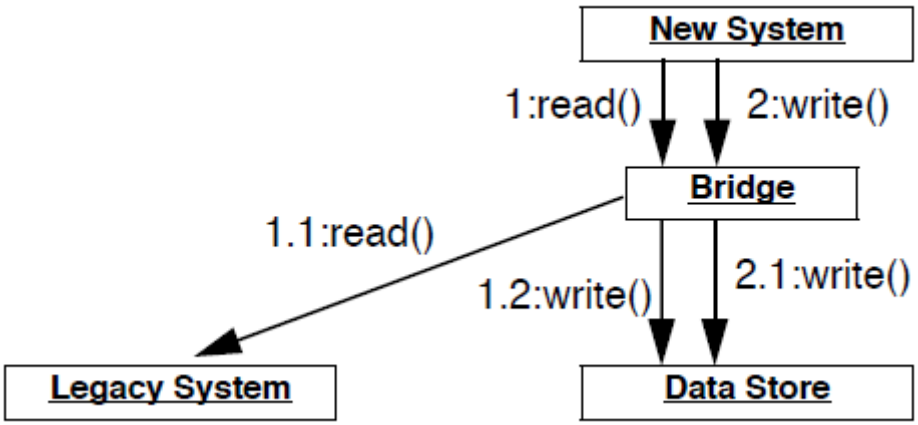

Figura 9: Modelo de migração de dados (DEMEYER, 2008)

A ponte deverá ser responsável por todas as conversões de dados necessárias, e cuidará dos processos de leitura da base legada e escrita na nova base, caso o dado ainda não esteja lá. O benefício desta técnica é que podemos começar a utilizar o novo sistema sem migrar todo os dados do legado. Conforme os módulos do sistema vão sendo desenvolvidos, as pontos que cuidarão de migrar os dados que eles acessão são desenvolvidas paralelamente. Desta maneia

facilita o trabalho de termos os dois sistema rodando em paralelo, porém devemos levar em consideração que o uso das mesmas poderá implicar em alguma perda no desempenho para fazer as transformações necessárias.

Com as técnicas vistas neste capítulo podemos modernizar um sistema legado utilizando abordagens que reduzem o risco a um custo mais aceitável. A modernização com entregas incrementais faz com que se ganhe confiança na reengenharia dos sistemas e permite o progresso ser medido a cada iteração. Como no início temos um sistema legado rodando em paralelo com o sistema modernizado podemos manter a familiaridade dos usuários e facilitar a aceitação do novo sistema, além de fornecer uma ampla base de testes por termos os dois sistema rodando em paralelo e somente quando a parte modernizada está completamente estável e testada é que a antiga é descontinuada.

Em toda modernização algum esforço no código legado deverá ser aplicado, mas devemos estabelecer limites a esses esforços e fazê-los até o ponto em que beneficiem o sistema modernizado. Com a criação dos serviços que abstraem a lógica monolítica do legado, conforme o tempo este código que fornece esses serviços vai sendo modernizado até o ponto em que todo o código do sistema tenha sido desenvolvido utilizando técnicas e boas práticas atuais de desenvolvimento que implicarão em uma maior qualidade e longevidade do sistema, com a cultura de melhoria contínua que deve ser implantada após todo esse processo.

5 CONCLUSÃO

Este trabalho se propôs a apresentar uma revisão bibliográfica sobre os assuntos que envolvem os sistemas legados e seu ciclo de vida, abrangendo os problemas que as empresas enfrentam com o legado assim como um estudo da abordagem de evolução mais adequada dado o momento atual em seu ciclo de vida, e como podemos identificar a estratégia de evolução mais adequada para os sistemas candidatos à modernização após um processo de levantamento do portifolio de sistemas e avaliação da sua qualidade técnica e valor para o negócio.

O trabalho inclui um amplo processo para o estudo da viabilidade de modernização desses sistemas numa forma que busca minimizar os riscos. Foram apresentadas algumas estratégias de modernização para esses sistemas e um processo de reengenharia que visa aplicar uma cultura de melhoria contínua através da utilização da reengenharia, engenharia reversa e engenharia para a frente, finalizando com a apresentação de como o sistema modernizado convive em paralelo ao sistema legado enquanto partes modernizadas dele vão sendo entregues e como os dados podem ser migrados para o novo sistema com a utilização de uma ponte para a conversão no novo formato.

Concluímos que para um projeto de modernização ser efetivo devemos utilizar uma abordagem que permita uma boa análise da qualidade técnica e valor para o negócio dos sistemas a fim de identificarmos quais sistemas devem continuar recebendo as atividades de manutenção normalmente e quais podem ser descontinuados e substituídos por um software pronto de algum fornecedor, até chegarmos aos candidatos à modernização.

Antes de partir para o projeto de modernização do sistema em si devemos avaliar uma série de fatores a fim de identificar e minimizarmos os riscos, que inclui entender os requisitos e o sistema legado, e então definir e validar a solução e a estratégia de modernização propostas a fim de se chegar na definição de uma estratégia de modernização que seja viável para a organização.

Com a estratégia de modernização do sistema em mãos podemos iniciar o projeto fazendo uso de um processo de reengenharia em espiral que envolve a reestruturação da documentação desse sistema e uma engenharia reversa para entender e criar as abstrações em alto nível que servirão como base para se chegar ao desenho do sistema modernizado, e então partir para a reestruturação do código e dos dados. Por fim, a engenharia para a frente deve cuidar que as abstrações em alto nível se transformem em código no novo sistema fazendo uso de tecnologias e técnicas modernas de engenharia de software para manter o sistema

modernizado com uma boa qualidade e em utilização por mais tempo, recriando a aplicação existente a cada iteração entregando novas tecnologias e funcionalidades.

Dentro do escopo deste mesmo trabalho, trabalhos futuros poderiam abordar a aplicação das técnicas apresentadas aqui em um estudo de caso envolvendo alguns sistemas reais de uma empresa, a fim de avaliar a aplicabilidade e os resultados desta abordagem em termos de esforço e custo.

REFERÊNCIAS BIBLIOGRÁFICAS

1 SOMMERVILLE, IAN Software Engineering, Ninth Edition. Boston, Addison Wesley, 2011.

2 PRESSMAN, ROGER S. Software Engineering: A Practitioner's Approach, Seventh Edition. New York, McGraw-Hill, 2010.

3 DEMEYER, SERGE; DUCASSE, STÉPHANE; NIERSTRASZ, OSCAR Object-Oriented Reengineering Patterns. Switzerland, Square Bracket Associates, 2008.

4 SEACORD, ROBERT C.; PLAKOSH, DANIEL; LEWIS, GRACE A. Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. Boston, Addison Wesley, 2003.

5 LEHMAN, M.; BELADY, L., Program Evolution: Processes of Software Change, Academic Press, 1997.

6 COMELLA-DORDA, SANTIAGO; WALLNAU, KURT; SEACORD, ROBERT C.; ROBERT, JOHN A Survey of Black-Box Modernization Approaches for Information Systems. San Jose, Software Maintenance, Proceedings. International Conference on, 2000.

7 ANITHA, JOSHPHIN; KARTHIKA, M.; ALAGARSAMY, K. The Classification of Risks in Reengineering Systems. New York, International Journal of Computer Applications 39 (18):57-61, February 2012. Published by Foundation of Computer Science, New York.

8 CHIKOFSKY, ELLIOT J.; CROSS, JAMES H. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 7 (January 1990): 13-17.

9 BOEHM, B A Spiral Model of Software Development and Enhancement. Computer, Vol. 21, no. 5, pp. 61-72, May 1988.