1ºCONTECSI Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação
21-23 de Junho de 2004 USP/São Paulo/SP - Brasil

Sistemas Legados e a Aplicação de Processos de Reengenharia de Software

Autor:

Luciano Lucindo Chaves

e-mail: lucianochaves@uol.com.br, llchaves@ipt.br

Fone: (11) 3685-6511

Instituição:

IPT - Instituto de Pesquisas Tecnológicas do Estado de São Paulo S.A.

Av. Prof. Almeida Prado, 532 – Ed. Adriano Marchini – 1º. Andar
Cidade Universitária – Butantã - São Paulo – SP

Resumo

Os sistemas de informação apóiam os negócios da maioria das organizações sendo que muitos desses sistemas são legados. Este artigo apresenta os aspectos críticos da manutenção de sistemas legados, os possíveis problemas causados por tais sistemas e ainda opções para o seu tratamento. A reengenharia de software é apresentada como opção para modernização ou substituição desses sistemas e são apresentadas recomendações para aplicar esse tipo de processo.

Palavras-Chave: Sistemas Legados, Sistemas de Informação, Manutenção de Sistemas, Reengenharia de Software, Engenharia de Software

1 Introdução

Praticamente toda organização depende direta, ou indiretamente, de sistemas de informação, pois esses sistemas dão suporte aos seus negócios. Grande parte dos sistemas utilizados atualmente foram construídos há muito tempo, em épocas onde eram usadas técnicas, ferramentas e tecnologias que hoje são consideradas superadas. Esse tipo de sistema é chamado de legado.

As organizações, independente das áreas de atuação, enfrentam o desafio de manter seus sistemas legados em funcionamento. Pois, na maioria dos casos, esses sistemas foram mal elaborados, gerando dificuldades no processo de manutenção (OSBORNE & CHIKOFSKY, 1990:11) e aumento de custos (JACOBSON & LINDSTROM, 1991:341).

Sendo assim, as organizações precisam definir estratégias para manter seus sistemas legados, com o objetivo de não serem surpreendidas e sofrerem os efeitos que tais sistemas podem trazer.

O objetivo desse trabalho é apresentar os aspectos críticos da manutenção e riscos no processo de gerência dos sistemas legados. Ainda são apresentados possíveis caminhos e recomendações para lidar com esse tipo de sistema.

O trabalho está organizado da seguinte forma. A seção 2 indica razões para os problemas encontrados em sistemas legados. São relacionados problemas comumente enfrentados na seção 3 e caminhos possíveis na seção 4. A seção 5 apresenta recomendações e na conclusão, seção 6, são sintetizadas as principais idéias deste trabalho.

2 O software em envelhecimento

Há quase 10 anos Pressman (1995:9-10) já fazia o seguinte diagnóstico:

- Sistemas de informação desenvolvidos há 20 anos sofreram dezenas de gerações de mudanças e são quase impossíveis de serem mantidos, havendo risco de falharem;

- Aplicações de engenharia, por causa da idade e estado de reparo não são satisfatoriamente entendidas, a estrutura interna desses sistemas é desconhecida;
- Sistemas de controle de fábricas, usinas de energia, tráfego aéreo e outros, exibem comportamentos inesperados e algumas vezes inexplicáveis, mas não podem sair de operação pois não há nada para substituí-los.

Além dos problemas mencionados há o fato de que os sistemas projetados antigamente eram desenvolvidos com preocupações em relação às limitações do hardware disponível na época, gerando sistemas que priorizavam a eficiência em detrimento da estrutura e clareza com que foram desenvolvidos.

Hoje, há uma preocupação crescente com qualidade de software mas, de qualquer maneira, as empresas têm a herança dos sistemas legados, isso sem mencionar sistemas desenvolvidos atualmente que ainda deixam a desejar em termos de qualidade.

Além dessas considerações, de acordo com Jacobson & Lindstrom (1991:341), o sistema de informação tem um tempo de vida limitado, cada mudança degenera sua estrutura e torna sua manutenção mais difícil e cara.

3 O desafio das organizações

Sistemas fundamentais para a operacionalização dos negócios estão cada vez mais difíceis de serem mantidos. Pois são remendados freqüentemente, resultando em falhas e funcionamento ineficiente, não correspondendo às necessidades dos usuários e tornando sua manutenção dispendiosa (PRESSMAN, 1995:163).

Há motivo para preocupação devido às características peculiares que esses sistemas apresentam. Considerando sistemas legados, apesar de que isso muitas vezes pode ser aplicado a sistemas desenvolvidos atualmente, podemos indicar algumas de suas características e os possíveis efeitos:

a. Falta de documentação

A documentação de software é necessária porque auxilia no processo de entendimento do mesmo. Esse entendimento é essencial para o processo de manutenção. Possíveis efeitos:

- dependência de pessoas específicas da organização, normalmente desenvolvedores de software, pois muito do negócio está embutido no sistema;
- dificuldade em medir o impacto em processos de manutenção

devido à falta de entendimento;

b. Tecnologias ultrapassadas

O termo tecnologia inclui hardware, software e técnicas utilizadas para a construção de tais sistemas. Possíveis efeitos:

- dificuldade em encontrar profissionais que trabalham com tal tecnologia;
- maior custo com profissionais por serem mais difíceis de encontrar;
- empresas fornecedoras de tecnologias descontinuam os produtos obrigando a organização a fazer atualizações;
- dificuldades na integração com outros sistemas;

c. Sistemas grandes e complexos

Sistemas crescem ao longo do tempo e geram estruturas complexas, difíceis de entender e manter. Possíveis efeitos:

- prazos maiores para realização de alterações;
- maior dificuldade na realização de alterações;
- maior custo de manutenção;

d. Elevado número de erros e falhas

A estrutura do sistema é corrompida por causa das sucessivas manutenções. Possíveis efeitos:

- indisponibilidade parcial ou total do sistema;
- prejuízos decorrentes dos erros e falhas do sistema;
- impacto na imagem da organização;
- custos não previstos;

Conclui-se, de acordo com o exposto, que a preocupação em relação a tais sistemas não deve ser puramente uma questão de atualização tecnológica. Mais que isso, a preocupação deve ser evitar que a organização, que como foi dito depende desses sistemas, tenha prejuízos e seu desempenho seja prejudicado pelos efeitos negativos de tais sistemas.

Cabe então, à organização, gerenciar os riscos em relação a tais sistemas e definir estratégias para resolução e ou minimização dos problemas.

4 Caminhos possíveis

Sobre o que fazer com sistemas legados, Jacobson & Lindstrom (1991:341) sugerem uma matriz de decisão baseada na relação entre a manutenibilidade do sistema e seu valor para negócio, conforme figura 1.

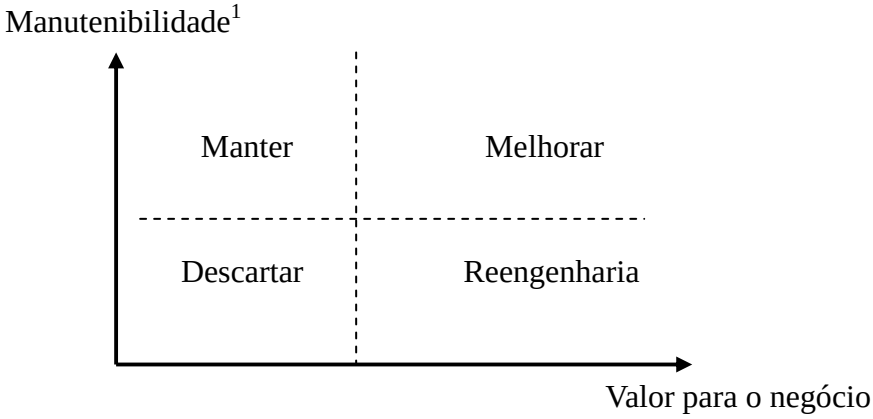

Figura 1 – Matriz de decisão

A decisão do que fazer com o sistema legado deve tomar por base o valor desse sistema para o negócio. Se o sistema tem baixo valor para o negócio e é difícil de manter a opção seria descartá-lo, por outro lado, se o sistema é difícil de manter mas possui grande valor para o negócio a opção seria aplicar a reengenharia de software.

Parecida com a matriz de decisão de Jacobson & Lindstrom (1991:341), Comella-Dorda et al. (2000:3-4) apresentam três opções possíveis quanto ao que fazer com o sistema legado:

Manter

O processo de manutenção é incremental e iterativo. Pequenas mudanças são feitas para corrigir erros ou incluir novas funcionalidades, normalmente sem exigir maiores mudanças estruturais no sistema. A manutenção é necessária para qualquer sistema mas limita possíveis vantagens competitivas em função da não

[1] Propriedade em relação à facilidade de manutenção do sistema

adoção de novas tecnologias. Custos com manutenção de sistemas legados tendem a crescer com o tempo e pequenas manutenções sucessivas podem afetar a integridade do sistema.

Modernizar

A modernização envolve mais mudanças que os processos normais de manutenção mas, ainda assim, conserva grandes porções do sistema existente. Essas mudanças podem incluir reestruturação, melhoria de funcionalidades ou novas atribuições ao sistema. A modernização é aplicável quando a organização tem conhecimento sobre a estrutura do sistema.

Substituir

A substituição de sistemas legados é uma opção quando estes já não atendem às necessidades e quando os custos de modernização não são compensadores.

Ainda segundo Comella-Dorda et al. (2000:2), podemos representar a situação de acordo com a figura 2.

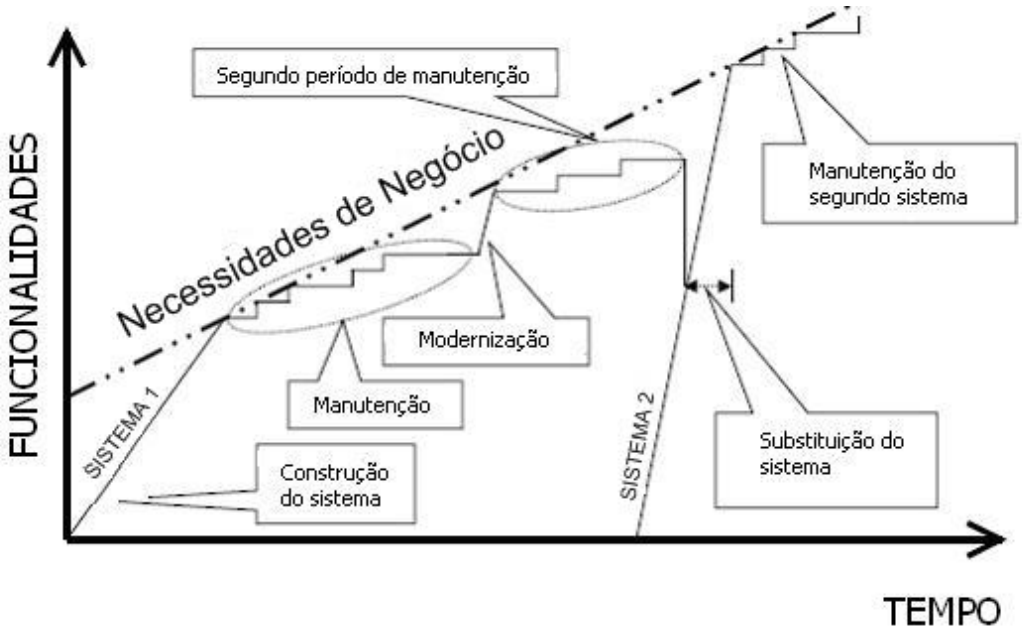

Figura 2 – Ciclo de vida de sistemas de informação

A gerência do ciclo de vida de sistemas é essencial, pois pode evitar que os

sistemas cheguem a uma situação crítica, de forma que, quando um sistema estiver em vias de ser desativado haja um novo sistema para substituí-lo, ou que os sistemas sejam modernizados de forma a aumentar sua vida útil.

A falta de gerência do ciclo de vida do sistema pode levar a uma situação onde a manutenção do sistema é inviável. Onde todos os recursos são direcionados para manutenção do sistema e desta forma restringindo recursos que poderiam ser aplicados na modernização ou reconstrução do mesmo.

As opções de modernizar e substituir o sistema levam à reengenharia de software.

4.1 Reengenharia de Software

A reengenharia de software recupera as informações de projeto de um software existente e usa essas informações para alterar ou reconstruir o sistema existente, num esforço para melhorar a qualidade global (PRESSMAN, 1995:900)

Desta forma, vemos que a reengenharia de software pode atender necessidades de substituir ou modernizar um sistema, atentando para o fato de que a substituição pode ser feita através de um sistema pronto, adquirido no mercado, e dessa forma eliminando a necessidade de reconstrução. Quando temos uma substituição, o sistema é reconstruído por completo. Na modernização, partes do sistema podem ser adaptadas e reaproveitadas.

Cabe dizer que a reengenharia de software não é aplicada apenas para sistemas legados, mas para qualquer sistema em que haja necessidade de reconstrução. Pode ocorrer que o sistema a ser reconstruído seja um sistema desenvolvido recentemente, com tecnologias atuais, mas por problemas de projeto há a necessidade de reconstrução.

O processo de reengenharia de software pode ser definido como (JACOBSON & LINDSTROM, 1991:341):

Reengenharia = Engenharia Reversa + Mudança + Engenharia Progressiva

Onde “Mudança” pode ser de dois tipos:

Mudança de funcionalidade: são acrescentadas novas funcionalidades para satisfazer necessidades de negócio ou para melhorar o desempenho do sistema.

Mudança de implementação: são alterados aspectos do sistema computacional tais como: sistema operacional, técnica de programação e linguagem de programação.

Nada impede que as mudanças de funcionalidade e de implementação sejam aplicadas de forma conjunta.

A engenharia progressiva é o processo normal de desenvolvimento de sistemas, onde, partindo das necessidades do cliente, os passos são executados na direção da construção do sistema.

4.2 Engenharia Reversa

Considerando sistemas legados, na maioria das vezes, há a necessidade de recuperar informações de projeto do sistema, devido à falta de documentação do sistema ou, ainda, por causa da documentação estar desatualizada. Para recuperar essas informações aplica-se um processo de engenharia reversa.

Engenharia reversa de software é o processo de analisar um programa, num esforço para criar uma representação do programa em um nível de abstração[2] maior do que o código fonte (PRESSMAN, 1995:900).

Ainda segundo Waters & Chikofsky (1994:24) a engenharia reversa promove a recuperação de informações, em alto nível, da arquitetura do sistema, para entendimento de detalhes e regras de negócio embutidas no sistema.

Dessa forma o objetivo da engenharia reversa é o de gerar visões do sistema, que podem assumir a forma de documentação descritiva, diagramas ou outros tipos de representação, que promoverão o entendimento da estrutura e funcionalidades do sistema e auxiliarão no processo de reconstrução.

[2] Habilidade de considerar um ou mais aspectos de uma totalidade complexa ignorando detalhes ou aspectos não relevantes

Técnicas para aplicação do processo de engenharia reversa podem ser encontrados em Demeyer, Ducasse & Nierstrasz (2000) e Penteado, Germano & Masiero (1996). Essas técnicas, também chamadas de padrões, podem auxiliar de forma significativa, apresentando uma espécie de receita de como aplicar a engenharia reversa, aumentando a eficiência e qualidade do processo.

5 Recomendações

5.1 Análise de Custo/Benefício

Quando da aplicação de um processo de reengenharia é necessário avaliar o custo/benefício de tal empreendimento, Sneed (1995:32-33) propõe um modelo para tal análise, onde são considerados os seguintes parâmetros:

P1 => custo anual de manutenção da aplicação.

P2 => custo anual de operação da aplicação.

P3 => valor comercial anual da aplicação.

P4 => previsão do custo anual de manutenção após Reengenharia.

P5 => previsão do custo anual de operação após Reengenharia.

P6 => valor comercial anual após Reengenharia.

P7 => custo estimado para Reengenharia.

P8 => previsão de tempo para realizar Reengenharia (em anos)

P9 => fator de risco da Reengenharia (fator multiplicador, ex. 1.25 indicará que os custos com reengenharia podem crescer em 25%)

L=> expectativa de vida do sistema (em anos)

O custo de manutenção contínua do sistema dá-se por :

$$CMC = [P3 – (P1 + P2)] \times L$$

O custo da reengenharia é :

$$CR = [P6 – (P4 + P5) \times (L – P8) – (P7 \times P9)]$$

Assim, a diferença entre o custo de manutenção contínua e o da reengenharia pode ser considerada para a tomada de decisão. De qualquer forma esta análise é

puramente em relação a custos e outros aspectos devem ser considerados.

5.2 Avaliando Riscos

A aplicação de um processo de reengenharia de software não é tarefa fácil, Bergey et al. (1999:4) fizeram um estudo patrocinado pelo Departamento de Defesa dos Estados Unidos (*DoD – U.S. Department of Defense*), apresentando as que seriam as dez principais razões para projetos de reengenharia de software falharem:

- escolha de uma estratégia de reengenharia ineficiente ou incompleta;
- organização faz uso inapropriado de ajuda externa (consultores, etc);
- força de trabalho esta “amarrada” com as velhas tecnologias, sem um programa de treinamento adequado;
- organização não tem o sistema legado sob controle;
- existe pouco levantamento e validação de requisitos;
- a arquitetura de software não é a consideração primária para reengenharia;
- não há noção da separação e distinção de processos de reengenharia;
- planejamento inadequado ou falta de resolução para seguir o planejamento;
- falta de gerência;
- gerência pré-determina decisões técnicas;

Complementando a lista de riscos, pode-se mencionar um outro grande risco. Trata-se da grande dificuldade no processo de desativação do sistema legado e ativação do novo sistema, principalmente se o sistema legado for grande.

Normalmente há um grande risco em simplesmente desativar o sistema legado e ativar o novo sistema. O recomendável é desativar o sistema legado gradualmente, conforme partes do sistema novo são construídas e ativadas. Para isso, deve-se definir uma estratégia para segmentar o desenvolvimento do novo sistema, de forma a facilitar e tornar mais seguro o processo de desativação de partes do sistema legado.

5.3 Utilização de Padrões

Uma atitude que pode diminuir riscos de forma significativa é aplicar um padrão de processo de reengenharia de software que já tenha sido aplicado com sucesso em

outros sistemas. Isso é importante pois pode trazer algum nível de previsibilidade na aplicação do processo e os erros mais comuns podem ser evitados. Padrões a serem aplicados em processos de reengenharia de software podem ser encontrados em Prado (1992) e Rechia & Penteado (2002).

Ainda existem trabalhos específicos onde é proposta a mudança automática da linguagem de programação usando ferramentas de apoio (FONTANETTE et al., 2002). Essa abordagem é aplicada para mudar sistemas de Clipper para Java (PRADO et al., 1998), de DataFlex para Visual DataFlex (NOGUEIRA & PRADO, 2001), de Progress para Java (PRADO & NOVAIS, 2000) e Cobol para C/C++ (LEITE, SANT'ANNA & PRADO, 1997).

A idéia de aplicar padrões na aplicação de processos de reengenharia de software é exatamente a mesma dos padrões aplicados na engenharia de software, ou seja, usar técnicas, metodologias e ferramentas que levem ao sucesso do projeto.

6 Conclusão

As organizações encontram muitos problemas no processo de manutenção de sistemas legados e isso causa aumento de custos. Esse é um ponto crítico, pois o negócio dessas organizações depende de tais sistemas e as conseqüências desses problemas podem prejudicar a organização. Os problemas de manutenção dos sistemas legados decorrem de projetos mal elaborados e das sucessivas manutenções desses sistemas ao longo dos anos.

Para evitar que sistemas cheguem a uma situação crítica deve haver um processo de gerência do ciclo de vida dos sistemas. Essa gerência deve acompanhar o sistema desde a sua construção, passando pelo processo de manutenção e terminando na desativação. A gerência deve ser realizada de forma que a necessidade de desativação seja identificada antecipadamente, para que haja tempo de elaborar um planejamento para modernização ou substituição do sistema.

A decisão do que fazer com o sistema legado deve ser embasada em critérios bem definidos. Um possível critério é a relação entre o valor do sistema legado para o

negócio e o nível de dificuldade na sua manutenção, uma análise de custo/benefício também pode ser feita para decidir sobre a aplicação de um processo de reengenharia de software.

O processo de reengenharia de software visa recuperar informações do sistema através da engenharia reversa e promover a reconstrução parcial ou total do sistema. Estratégias devem ser definidas e riscos avaliados para execução da reengenharia de software. Podem ser utilizados padrões, que são técnicas e metodologias, que podem mitigar riscos e aumentar as chances de sucesso do projeto de reengenharia.

Referências

BERGEY, J. et al. *Why Reengineering Projects Fail* (CMU/SEI-99-TR-010), Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 1999

COMELLA-DORDA, S. et al. *A survey of Legacy System Modernization Approaches* (CMU/SEI-2000-TN-003). Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 2000

DEMEYER, S.; DUCASSE, S.; NIERSTRASZ, O. *A Pattern Language for Reverse Engineering.* In: Proceedings of the 5th European Conference on Pattern Languages of Programming and Computing (EuroPLOP'2000), 2000.

FONTANETTE et al. *Reengenharia de Software usando Transformações.* In: The Second Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC/2002), Salvador, Brazil, 2002.

JACOBSON, I; LINDSTROM, F. *Re-engineering of old systems to an object-oriented architecture.* SIGPLAN Notices, v.26, n.11, p.340-350, 1991.

LEITE, J. C. S. P.; SANT'ANNA, M. ; PRADO, A. F. *Porting Cobol Programs Using Transformational Aprroach.* Journal of Software Maintenance: Research and Practice, John Wiley & Sons Ltd., vol. 9, n.1, p. 3-31, 1997

NOGUEIRA, A. R., PRADO, A. F. *Transformação de Dataflex Procedural para Visual Dataflex Orientado a Objetos Reutilizando um Framework.* In: Workshop de Teses - XV Simpósio Brasileiro de Engenharia de Software - SBES'2001. Rio de Janeiro-RJ, Brasil, 2001.

OSBORNE, W. M.; CHIKOFSKY, E. J. *Fitting Pieces to the Maintenance Puzzle***.** IEEE Software, v. 7, n. 1, p. 11-12, 1990.

PRADO, A. F. *Estratégia de Reengenharia de Software Orientada a Domínios.* 1992, 333 p., Tese (Doutorado em Informática), PUC-RJ, 1992

PRADO, A. F. et al. *Reengenharia de Programas Clipper para Java.* In: XXIV Conferência Latino Americana de Informática - CLEI 98, Quito-Equador, p.383-394, 1998.

PRADO, A. F., NOVAIS, E. R. A. *Reengenharia Orientada a Objetos de Código Legado Progress 4GL* In: XIV Simpósio Brasileiro de Engenharia de Software - SBES'2000, João Pessoa-PB, Brasil, p. 21-36, 2000.

PENTEADO, R.D.; GERMANO, F.; MASIERO, P.C. *An Overall Process Based on Fusion to Reverse Engineer Legacy Code.* In: III Working Conference on Reverse Engineering, IEEE, Monterey, California, p. 179-188, 1996.

PRESSMAN, R. S. *Engenharia de Software.* São Paulo: Makron Books, 1995.

RECHIA E. L.; PENTEADO, R. *FaPRE/OO: Uma Família de Padrões para Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais.* In: Proceedings of Second Latin American Conference on Pattern Languages of Programming (Sugarloaf Plop'2002), 2002

SNEED, H. *Planning the Reengineering of Legacy Systems.* IEEE Software, v.12, n.1, p. 24-34, 1995.

WATERS, R. C.; CHIKOFSKY, E. J. *Reverse Engineering: Progress Along Many Dimensions.* Communications of the ACM, v. 37, n. 5, p. 23-24, 1994.