Manutenção de Software: Definições e Dificuldades - Artigo Revista SQL Magazine 86

Esse artigo apresenta algumas definições de manutenção de software, os tipos existentes, os impactos de sua aplicação e como é utilizada durante o desenvolvimento de um sistema.

SQL Magazine 86

[Artigo disponível no **Leitor Digital DevMedia**. Clique aqui para acessá-lo]

> Clique aqui para ler todos os artigos da SQL Magazine 86

De que se trata o artigo

Esse artigo apresenta algumas definições de manutenção de software, os tipos existentes, os impactos de sua aplicação e como é utilizada durante o desenvolvimento de um sistema.

Para que serve

A manutenção de software é um processo de melhoria de um software já desenvolvido, ou que está sendo desenvolvido. Com a manutenção também é possível corrigir erros que são encontrados durante a utilização do sistema pelo usuário ou por testes realizados pelos desenvolvedores.

Em que situação o tema é útil

A manutenção de software é hoje um assunto presente em organizações que desenvolvem e mantém software. Isso se deve à necessidade de sempre ajustar e melhorar o produto de software de acordo com as mais diversas necessidades. Diante desse fato, entender o significado e abrangência do termo manutenção de software pode auxiliar organizações e profissionais interessados no tema a melhor conduzir seus esforços quando precisam manter seus produtos.

Sabemos que a vida de um software não termina após a sua implantação. Ele ainda viverá durante muito tempo. Será utilizado por anos, e com certeza, terá muitas atualizações, gerando novas versões do sistema. Nesse sentido, a manutenção é caracterizada pela modificação do software já entregue ao cliente, ou seja, a manutenção é qualquer alteração no software após sua entrada em produção.

Embora a definição de manutenção de software trate genericamente qualquer produto de software, existem diferenças entre a manutenção de softwares com propósitos distintos.

Uma primeira classificação representa aqueles softwares construídos com base em uma especificação rígida e bem definida, cujos resultados esperados são bem conhecidos. Por exemplo, um software construído para realizar operações com matrizes (adição, multiplicação e inversão). Nesse tipo de software, uma vez que tenha sido construído considerando a correta implementação do método, dificilmente haverá a necessidade de manutenções.

Já em uma segunda classificação, são agrupados os softwares que constituem implementações de soluções aproximadas para problemas do mundo real, uma vez que soluções completas somente são conseguidas na teoria nesses casos. Como exemplo, pode-se citar um jogo de xadrez. Embora suas regras sejam bem definidas, não é possível construir um software que calcule a cada passo todos os possíveis movimentos de peças do tabuleiro, de forma a determinar o melhor movimento. Isso porque o número de movimentos possíveis é muito grande para ser calculado em um intervalo de tempo relativamente curto. A técnica utilizada para desenvolver esse tipo de solução baseia-se em descrever o problema de forma abstrata e então definir os requisitos de software a partir dessa abstração.

Percebe-se que esse tipo de sistema já abre espaço para diferentes interpretações por parte do desenvolvedor, o que tende a produzir software com maior necessidade de manutenção do que quando comparado aos da classificação anterior. Por considerar uma abstração para especificação de requisitos, a necessidade de mudança pode aparecer caso a abstração mude, na medida em que um maior entendimento do problema seja alcançado.

Finalmente, uma terceira e última classe de softwares considera mudanças no ambiente onde o software vai ser utilizado, característica não existente nas duas classificações anteriores.

Um software integrante da terceira classificação corresponde àquele criado com base em um modelo dos processos abstratos envolvidos no sistema, e precisará mudar sempre que ocorrer mudanças nesses modelos, sendo, portanto, parte do ambiente que ele modela. Um exemplo desse tipo de software seria aquele que apresenta informações da economia de um país. À medida que a economia passa a ser mais bem compreendida, o modelo muda e com ele a abstração do problema, causando uma necessidade inevitável de manutenção no software. Esse tipo de software, comumente encontrado no dia-a-dia das organizações, tem interesse particular neste trabalho.

O objetivo da manutenção de software é diferente da manutenção de produtos industrializados, pois a manutenção realizada não é feita para reparar danos causados pelo tempo de uso do software. Defeitos não são introduzidos pelo tempo e nem pela carga de utilização. Os defeitos encontrados já existiam, antes do software entrar em produção. Por algum motivo, não foram detectados em fases anteriores. Mas a manutenção não se caracteriza apenas por correções. Existem três tipos principais de manutenções em softwares, Adaptativas, Corretivas e Evolutivas (Perfectivas):

• **Adaptativas:** são alterações que visam adaptar o software a uma nova realidade ou novo ambiente externo, normalmente imposto. Um exemplo claro seriam mudanças de leis ou regras, definidas pelo governo e/ou órgãos reguladores. Assim, manutenções do tipo adaptativas referem-se a adequar o software ao seu ambiente externo. O exemplo apontado por Pfleeger (2001) ilustra bem essa categoria. Suponha um gerenciador de banco de dados, que faz parte um sistema maior de hardware e software. Em uma atualização do gerenciador, os programadores perceberam que as já existentes rotinas de acesso a disco precisavam agora de mais um parâmetro adicional. Essa manutenção corresponde a uma manutenção adaptativa, uma vez que teve por finalidade adequação do software ao seu ambiente e não a correção de um defeito.

• **Corretivas:** como o próprio nome diz, servem para eliminar as falhas encontradas em produção. É bastante comum, principalmente quando o processo de desenvolvimento não se preocupou de maneira adequada com a qualidade do software. Assim, manutenções do tipo corretivas visam corrigir defeitos de funcionalidade, o que inclui acertos emergenciais de programa. Pfleeger (2001) expõe um exemplo desse tipo de manutenção, que consiste em um usuário apresentando um problema de impressão em um relatório. O número de linhas impresso por folha é muito grande, o que causa sobreposição de informações. O problema foi identificado como uma falha no driver da impressora, provocando a necessidade de se alterar o menu do relatório para aceitar um parâmetro adicional que determina o número máximo de linhas impressas por folha.

• **Evolutivas:** são alterações que visam agregar novas funcionalidades e melhorias para os usuários que as solicitaram. Não se deve confundir esse tipo de manutenção com as entregas programadas de um processo de desenvolvimento interativo. A integração com outros sistemas também é considerada um tipo de evolução.

As migrações para novas tecnologias, sejam de hardware ou software, podem causar dúvidas quanto à classificação do tipo de manutenção. Se a tecnologia foi imposta, a manutenção é classificada como adaptativa. Se a tecnologia é uma solução de melhoria do sistema, é classificada como evolutiva.

Segundo [LIENTZ], a proporção do esforço de manutenção é de 20% para corretivas, 25% adaptativas, e para 50% evolutivas.

Neste contexto, este artigo irá apresentar algumas das definições básicas sobre manutenção de software, destacando também algumas dificuldades encontradas ao praticá-la.

Processos de Manutenção de Software

A atividade de manutenção requer o cumprimento de algumas etapas consideradas adequadas para se obter um resultado positivo. Deve-se primeiramente avaliar a documentação e código existentes, juntamente com a arquitetura, estrutura de dados e interface. Posteriormente, identificar as modificações necessárias e avaliar seus impactos. Em seguida, realizar as modificações e testá-las através da aplicação de testes.

Alguns problemas podem ocorrer durante a realização de algum processo de manutenção. Eles devem ser na medida do possível previstos para poder ser evitados, tais como:

• Ausência ou deficiência na documentação;

• Dificuldade na identificação de manutenções realizadas anteriormente;

• Falta de controle de versão;

• Baixa manutenibilidade do software.

Além disso, efeitos colaterais podem ocorrer com a realização desses processos:

• Modificação da estrutura do código;

• Inclusão de códigos com erros;

• Desatualização da documentação.

Através dessas características, deve-se identificar e registrar todas as partes que foram afetadas pela modificação.

Para a manutenção de software, os sistemas devem sofrer mudanças de acordo com o contexto em que estão inseridos. Para melhorar o processo de manutenção devem ser levadas em conta as mudanças ocorridas no ambiente em que está inserido o sistema.

Na manutenção de software existem quatro fases que são introdução, crescimento, maturidade e declínio. Destaca-se que durante a fase de introdução existe um grande suporte ao usuário, período em que são concebidas as primeiras idéias sobre o sistema. As fases de maturidade e de crescimento do software são de ajustes, onde o sistema já está concretizado e muitas vezes correções aparecem naturalmente. Ainda na fase de maturidade, podem ser realizadas melhorias no programa. Durante essas fases é necessário que sejam realizados testes e atualização da documentação para avaliar as partes modificadas e acrescentadas na aplicação. Na fase de declínio o sistema chega ao final da vida útil e deve ser avaliado se o sistema deve ser retirado de operação. A avaliação deve ser feita com base nos aspectos econômicos como, por exemplo, substituição de tecnologias, ou até mesmo para conseguir independência de fabricante.

Efeitos colaterais

Quando se altera o software, seja para correção, evolução ou adaptação, existe a possibilidade de se introduzir novos defeitos ou reintroduzir erros que ocorriam anteriormente no mesmo. É o chamado efeito colateral. O ideal seria que todos os módulos ligados à parte alterada ou o sistema todo fossem retestados. Mas devido aos curtos prazos, principalmente em manutenções corretivas emergenciais, quase nunca isso é possível. A falha causada através do efeito colateral pode aparecer em qualquer ponto do software, podendo ficar fora da cobertura dos testes daquela manutenção.

Isto é um desafio para os testadores, pois eles precisam garantir o correto funcionamento do sistema como um todo. Uma forma de prevenir este efeito seria realização dos testes de regressão, que são realizados para garantir que os novos defeitos introduzidos ou desmascarados sejam encontrados e removidos.

Para atender aos curtos prazos, os testes de regressão devem ser automatizados. Realizá-los manualmente não seria viável. Entretanto, automatizar é uma tarefa complexa e custosa, que exige grande esforço. Mas, uma vez automatizado, o teste de regressão torna-se muito mais econômico, correspondendo apenas a uma fração do tempo gasto caso fosse realizado manualmente. Uma sugestão para diluir o alto custo da automação dos testes de regressão, seria a automação de partes do sistema a cada nova demanda de manutenção, alcançando a situação ideal no médio ou longo prazo.

A automação é particularmente importante para obter um nível razoável de garantia em um ciclo de desenvolvimento muito rápido, como por exemplo, o ciclo das manutenções corretivas. Uma vez atingido esse cenário ideal, os testes serão rápidos e eficazes, mitigando os riscos das falhas causadas pelo efeito colateral.

Custos e Desafios

A manutenção de software é uma operação importante, pois consome a maior parte dos custos envolvidos no ciclo de vida de um software, e a falta de habilidade em mudar um software rapidamente, e de maneira confiável, pode causar a perda de oportunidades de negócio.

Embora não exista um consenso sobre o valor exato do custo atrelado à atividade de manutenção, as pesquisas na área apontam, na totalidade dos casos, sempre mais de 50% dos investimentos realizados no software. A razão do custo elevado deve-se, em parte, à própria natureza da atividade de manutenção, caracterizada principalmente pela imprevisibilidade. Além dos altos custos financeiros, essa é também a atividade que exige maior esforço dentre as atividades de engenharia de software. Pressman (2005) ainda completa que o grande esforço necessário na manutenção se justifica pela abrangência do significado desse termo no contexto d software.

A importância financeira atrelada à manutenção de software é ainda agravada quando se leva em consideração o risco para as oportunidades de negócio, que podem ser causadas pela falta de gerenciamento e compreensão total da dinâmica da atividade. Esse gerenciamento deve considerar três fatores: (i) ferramentas, (ii pessoas, (iii) processos, revelando-se, pois, uma atividade gerencial complexa.

Se por um lado a atividade de manutenção é dispendiosa, por outro, ela é um desafio para as organizações que precisam considerá-la em seu dia-a-dia. Não é de se esperar que uma empresa de grande porte troque todos seus sistemas somente pelo fato de que a tecnologia neles empregada está ultrapassada. Esses sistemas representam ativos importantes da organização e ela estará disposta a investir de maneira a manter seus valores.

Prever quando uma manutenção precisará ser conduzida é uma tarefa geralmente muito difícil de ser realizada. Essa dificuldade de previsão, e também controle sobre a manutenção, é explicada pelo fato de muitas vezes ficar a cargo de empresas terceirizadas a manutenção, revelando-se um problema já que normalmente essas organizações não possuem nenhum contato com o projeto inicial do software. O aumento na complexidade dos softwares produzidos (tanto em termos de funcionalidades, como de técnicas) torna a previsão de esforços de manutenção muito vaga. Essa dificuldade em estimar esforços torna-se mais evidente quando se trata de sistemas legados.

Além disso, a atividade de manutenção de software sofre de uma série de dificuldades exemplificadas na **Tabela 1**. Os problemas, como pode ser observado, foram classificados de acordo com sua natureza em problemas gerenciais e técnicos.

Problemas Gerenciais	Ausência de um processo de manutenção de software
	Grande expectativa dos usuários
	Elevada rotatividade de membros e funções dentro da equipe
	Sobrecarga de tarefas
	Estimativa de prazo não condizente com a complexidade do software
	Baixa motivação entre profissionais de manutenção
	Ausência de manutenção preventiva
	Falhas de comunicação com o usuário
	Atrasos na entrega
Problemas Técnicos	Registro inexistente ou superficial de manutenções anteriores
	Ausência de um ambiente computacional específico para manutenção
	Validação insuficiente de manutenções efetuadas
	Documentação insuficiente ou superficial
	Falta de compreensão do software e suas estruturas

Tabela 1. Problemas de manutenção de software

Ferramentas de apoio à manutenção de software

Hoje no mercado existem ferramentas que foram desenvolvidas para auxiliar os desenvolvedores a realizarem a manutenção de software. Não será demonstrado nesse artigo uma ferramenta específica para gerenciar a manutenção de software. Ao invés disso, iremos explicar de forma geral como podem auxiliar a equipe de desenvolvimento e gerência durante a tomada de decisões.

Atualmente existem ferramentas que possuem recursos direcionados ao gerenciamento de equipamentos e auxílio à mão-de-obra, por exemplo, tentando maximizar tempo e produtividade. Podem também direcionar recursos para correção e prevenção de falhas nos produtos, além de gerar relatórios, gráficos e indicadores gerenciais. Esses relatórios e indicadores permitem que a gerência tenha um domínio maior sobre os processos de desenvolvimento e sobre os produtos construídos. Muitas dessas ferramentas permitem que o gerente do projeto receba notificações sobre algum ocorrido, como alarmes e anomalias.

Existem ainda outras ferramentas que podem ser usadas para auxiliar a equipe que irá realizar a manutenção do sistema. São ferramentas que realizam controle de versão das aplicações. O controle de versão permite o gerenciamento de diferentes versões do código fonte da aplicação ou da documentação do sistema. Com o versionamento, a manutenção de software pode se beneficiar de algumas funcionalidades que são muito úteis para implantação da manutenção no projeto, como o registro de todas as modificações realizadas ou a recuperação de uma versão específica de um produto.

Outra vantagem é que mais de um desenvolvedor pode realizar a manutenção em determinada parte do programa ao mesmo tempo. Ao alterar alguma parte do código, é possível unir diferentes modificações, e se o código realiza a mesma rotina, pode-se escolher a melhor implementação.

O que pode ser feito no código fonte pode ser feito nos documentos relacionados ao sistema. É primordial que a documentação esteja sincronizada e condizente com o que está sendo feito na aplicação. As ferramentas de controle de versão realizam o gerenciamento do código da aplicação, podendo, também, realizar o gerenciamento da documentação usada no sistema.

Outras ferramentas apóiam o relato de defeitos e solicitações de mudanças, fazendo o controle do ciclo de vida de uma solicitação de mudança.

Considerações Finais

A manutenção de software é uma atividade muito importante na prática das empresas de desenvolvimento de software. Apesar de sua importância, ainda é uma prática mal vista pela maioria dos profissionais, pouco estudada e entendida.

Nesse artigo foi possível conhecer um pouco sobre o que é a manutenção de software e suas principais características e os tipos existentes, além de conhecer algumas práticas e ferramentas de manutenção que auxiliam no desenvolvimento do projeto, e que devem ser consideradas para que as manutenções necessárias em um software sejam realizadas com o maior sucesso possível.

Referências

• PRESSMAN, Roger S. Engenharia de software. 6ª. ed. São Paulo : McGraw-Hill, 2006. xxxi, 720p.

• BENNETT, K. H.; RAJLICH, V. T. (2000) "SOFTWARE MAINTENANCE AND EVOLUTION: A ROADMAP", IN: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING, LIMERICK, IRELAND, JUNE.

• BENNETT, K. H.; RAMAGE, M.; MUNRO, M. (1999) "DECISION MODEL FOR LEGACY SYSTEMS", IEEE PROCEEDINGS ON SOFTWARE (TSE), V.146, N. 3, P. 153-159.

• BHATT, P.; SHROFF, G.; MISRA, A. K. (2004) "DYNAMICS OF SOFTWARE MAINTENANCE", ACM SIGSOFT SOFTWARE ENGINEERING NOTES, V. 29, N. 5, P. 1-5.

• BASILI, V. (1990) "VIEWING MAINTENANCE AS REUSE-ORIENTED SOFTWARE DEVELOPMENT", IEEE SOFTWARE, V. 7, N. 1, P. 19-25.

• CHAPIN, N. (1986) "SOFTWARE MAINTENANCE: A DIFFERENT VIEW", IN: CONFERENCE ON SOFTWARE MAINTENANCE, ORLANDO, FL, USA, NOVEMBER.

• LIENTZ, B.P; Swanson, E.B. (1980) "Software Maintenance Management", Reading, MA, Addison Wesley.

• GHEZZI, C.; MANDRIOLI, D. (2005) "The Challenges of Software Engineering Education", In: 27th International Conference on Software Engineering, Saint-Louis, Missouri, USA, May.

• HAZZAN, O.; DUBINSKY, Y. (2003) "Teaching a Software Development Methodology: The case of Extreme Programming", In: 16th Conference o n Software Engineering Education and Training (CSEE&T 2003), Madrid, S pain, March.

• IEEE (1998) "Std 1219 – IEEE Standard for Software Maintenance", Ins titute of Electrical and Eletronic Engineers, New York, NY, USA.

• ISO/IEC 12207 (1998) "Standard for Information Technology - Softwar e Lifecycle Processes", International Standard Organization, New York, NY, USA.

• ISO/IEC 12207 (2002) "Standard for Information Technology - Softwar e Lifecycle Processes/Amd.1", International Standard Organization, New York, NY, USA.

por Rodrigo Spinola
Revista SQL Magazine lover