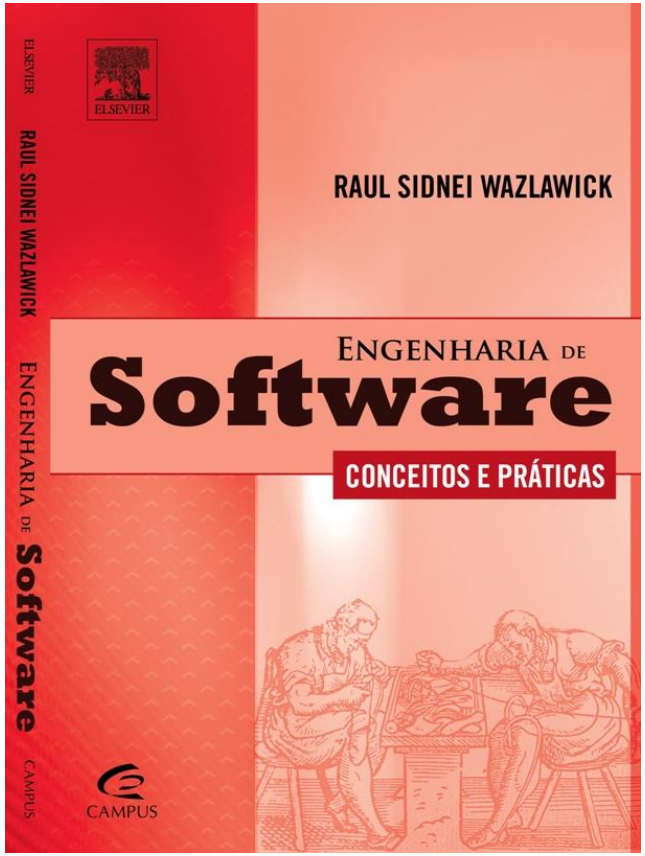

Manutenção de Software

Engenharia de Software: Conceitos e Práticas

Prof. Raul Sidnei Wazlawick
UFSC-CTC-INE

Manutenção

- Manutenção de software é como se denomina, usualmente, o processo de adaptação e otimização de um software já desenvolvido, bem como a correção de defeitos que ele eventualmente tenha.

- A manutenção é necessária para que um produto de software mantenha sua qualidade ao longo do tempo, já que, se isso não for feito, haverá uma deterioração do valor percebido deste software e, portanto, de sua qualidade.

Evolução

- Modernamente o termo “manutenção de software” vem sendo substituído ou utilizado em conjunto com “evolução de software”.

- *Evolução* talvez seja um termo mais adequado porque usualmente as atividades de modificação do software na fase de Produção não visam mantê-lo como está, mas fazê-lo evoluir de forma a adaptar-se a novos requisitos ou ainda a corrigir defeitos que possivelmente tenha.

Necessidade da Manutenção

- Considera-se que um software uma vez desenvolvido terá um valor necessariamente decrescente com o passar do tempo. Isso ocorre por que:
  - Falhas são descobertas.
  - Requisitos mudam.
  - Produtos menos complexos, mais eficientes ou tecnologicamente mais avançados são disponibilizados.

- Desta forma, torna-se imperativo que, simetricamente, para manter o valor percebido de um sistema:
  - Falhas sejam corrigidas.
  - Novos requisitos sejam acomodados.
  - Seja buscada simplicidade, eficiência e atualização tecnológica.

Classificação das Atividades de Manutenção

- *Corretiva*:
  - é toda atividade de manutenção que visa corrigir erros ou defeitos que o software tenha.
- *Adaptativa*:
  - é toda atividade que visa adaptar as características do software a requisitos que mudaram, seja novas funções, sejam questões tecnológicas.
- *Perfectiva*:
  - é toda atividade que visa melhorar o desempenho ou outras qualidades do software sem alterar necessariamente sua funcionalidade.
- *Preventiva*:
  - é toda atividade que visa melhorar as qualidades do software de forma que erros potenciais sejam descobertos e mais facilmente resolvidos.

Manutenção Corretiva

- Visa corrigir os defeitos (que provocam erros) que o software possa ter.

- Ela ainda pode ser subdividida em dois subtipos:
  - Manutenção para correção de erros conhecidos.
  - Manutenção para detecção e correção de novos erros.

- Os erros conhecidos de um software são usualmente registrados em um documento de considerações operacionais, ou em notas de versão.

Manutenção Adaptativa

- Conforme visto nas Leis de Lehman, a *manutenção adaptativa* é inevitável quando se trata de sistemas de software. Isso por que:
  - Requisitos de cliente e usuário mudam com o passar do tempo.
  - Novos requisitos surgem.
  - Leis e normas mudam.
  - Tecnologias novas entram em uso.
  - Etc.

- O sistema desenvolvido poderá estar ou não preparado para acomodar tais modificações de contexto.

Requisitos permanentes e transitórios

- Com os requisitos permanentes acontece o seguinte:
  - É mais barato e rápido incorporá-los ao software durante o desenvolvimento.
  - É mais caro e demorado mudá-los depois que o software está em operação.

- Com os requisitos transitórios acontece o inverso:
  - É mais caro e demorado incorporá-los ao software durante o desenvolvimento.
  - É mais barato e rápido mudá-los depois que o software está em operação.

Manutenção Perfectiva

- Consiste em mudanças que afetam mais as características de desempenho do que de funcionalidade do software.

- Usualmente tais melhorias são buscadas em função de pressão de mercado, visto que produtos mais eficientes, ou com melhor usabilidade, com mesma funcionalidade são usualmente preferidos em relação aos menos eficientes, especialmente em áreas onde o processamento é crítico, como jogos e sistemas de controle em tempo real.

- A melhoria de características vai estar quase sempre ligada às qualidades externas do software, mas especialmente àquelas qualidades ligadas a funcionalidade, confiabilidade, usabilidade e eficiência.

Manutenção Preventiva

- Pode ser realizada através de atividades de reengenharia, nas quais o software é modificado para resolver problemas potenciais.

- Por exemplo, um sistema que suporta até 50 acessos simultâneos e que já conta com picos 20 a 30 acessos, pode sofrer um processo de manutenção preventiva, através de reengenharia ou refatoração de sua arquitetura, de forma que passe a suportar 500 acessos, desta forma afastando a possibilidade de colapso por um período de tempo razoável.

- Outro uso da manutenção preventiva consiste em aplicar técnicas de engenharia reversa ou como refatoração ou redocumentação para melhorar a manutenibilidade do software.

Processo de Manutenção

- Análise de esforço para a tarefa de manutenção.
- Análise de risco para a tarefa de manutenção (verificando possíveis riscos, sua probabilidade e impacto, bem como elaborando e executando possíveis planos de mitigação).
- Planejamento da tarefa de manutenção (estabelecendo prazos, responsáveis, recursos e entregas).
- Execução da tarefa de manutenção.

Norma ISO 1219-98

- *Classificação* e *identificação* da requisição de mudança.
  - Essa atividade vai avaliar, entre outras coisas, se a manutenção necessária será corretiva, adaptativa ou perfectiva e, em função de sua urgência, ela receberá um lugar na fila de prioridades das atividades de manutenção.
  - Opcionalmente também a solicitação de modificação poderá ser rejeitada se for impossível ou indesejável implementá-la.
- *Análise.*
  - Aqui as atividades tradicionais de análise entram em cena, com a identificação ou modificação de requisitos, modelo conceitual, casos de uso e outros artefatos, conforme a necessidade.
- *Design.*
  - Aqui entram as atividades usuais de *design*, com a definição da tecnologia e das camadas de interface, persistência, comunicação, etc.
- *Implementação.*
  - Onde é gerado novo código que atende à modificação solicitada, bem como são feitos os testes de unidade e integração.
- *Teste de sistema.*
  - Onde são feitos os testes finais das novas características do sistema do ponto de vista do usuário.
- *Teste de aceitação.*
  - Onde o cliente é envolvido para aprovar ou não as modificações feitas.
- *Entrega.*
  - Onde o produto é entregue ao cliente para nova instalação.

Suporte a Usuários

- O *suporte a usuários* fará a interface entre o cliente do software e a empresa que presta manutenção ao software. O suporte a usuários usualmente recebe as reclamações, faz uma triagem delas e, ou encaminha uma solução previamente conhecida ao cliente, ou encaminha o problema ao setor de manutenção.

- O tamanho da equipe de suporte dependerá de vários fatores, dentre os quais, os mais importantes são a quantidade esperada de defeitos e a quantidade de clientes.

- Estima-se que para um software típico (que não apresenta grandes problemas de qualidade logo de partida), um atendente consiga tratar as chamadas de cerca de 150 clientes por mês, caso o meio de contato seja o telefone. Por outro lado, se o meio de contato for email ou chat, esse número pode subir para 1000 usuários por atendente por mês.

Migração entre Plataformas

- A migração de um produto para outra plataforma, quando se trata de software personalizado, é feita por demanda do cliente. Quando se trata de software de prateleira, é feita com a intenção de aumentar o mercado.

- Normalmente migrações são projetos por si só, embora possam ser consideradas atividades de evolução de software. Assume-se que sistemas desenvolvidos de acordo com boas práticas e com boa documentação possam ser migrados a uma taxa de 50 pontos de função por desenvolvedor-mês. Porém, se os sistema forem mal documentados e com organização obscura, essa taxa pode baixar para até 5 pontos de função por desenvolvedor-mês.

Conversão de Arquitetura

- Uma conversão de arquitetura de sistema usualmente é feita por pressão tecnológica. É o caso, por exemplo, de mudar de arquivos simples para bancos de dados relacionais, ou mudar uma interface orientada a linha de comando para uma interface gráfica.

- No caso da migração entre plataformas, se o software for personalizado, a conversão possivelmente será uma demanda do cliente, enquanto que no caso de software de prateleira a conversão será uma estratégia para buscar novos mercados.

- A conversão de arquitetura também pode ser uma estratégia para melhorar a manutenibilidade de um sistema.

- A produtividade de um projeto de conversão de arquitetura dependerá basicamente da qualidade das especificações do sistema. Quanto mais obscuras as especificações, mais difícil será a conversão.

- Usualmente sistemas mal documentados ou obscuros precisarão passar por processos de engenharia reversa antes de serem convertidos para uma nova arquitetura.

Adaptações Obrigatórias

- Talvez o pior tipo de manutenção de software sejam as adaptações obrigatórias, devidas a mudanças em leis, formas de cálculo de impostos, etc.

- O problema é que essas mudanças são completamente imprevisíveis pela equipe de desenvolvimento ou manutenção e mesmo pelo cliente.

- Além disso, elas normalmente têm um prazo curto e estrito para serem aplicadas e as penalidades por não adaptação costumam ser altas.

Otimização de Performance

- Atividades de otimização de performance implicam em analisar e resolver gargalos da aplicação, usualmente relacionados com acesso a dados, processamento e número de usuários simultâneos.

- Tais atividades variam muito em relação ao tipo e a carga de trabalho e, assim, é muito difícil estabelecer um padrão para estimação de custos.

Melhorias

- São um tipo de manutenção adaptativa e perfectiva que são iniciadas, normalmente por requisição dos clientes, que são os que normalmente acabam arcando com os custos relacionados.

- Melhorias muitas vezes implicam na introdução de novas funcionalidades, de forma que as técnicas usuais de estimação de esforço por CII, pontos de função ou pontos de caso de uso podem ser aplicadas.

- Pode-se considerar que existam dois tipos de melhoria:
  - *Pequenas melhorias*,
    - consistindo de aproximadamente 5 pontos de função, ou seja, a introdução de um novo relatório, consulta ou tela.
  - *Grandes melhorias*,
    - consistindo de um número significativamente maior de pontos de função, tipicamente mais de 20 pontos de função, que devem ser tratadas como pequenos projetos de desenvolvimento.

Engenharia Reversa e Reengenharia

- Em algumas situações o processo de manutenção ou evolução de um sistema exige uma atividade mais drástica do que simplesmente consertar partes do código.

- Sistemas antiquados, mal documentados e mal mantidos poderão requerer um processo completo de reengenharia para que possam voltar a evoluir de forma mais saudável.

- A reengenharia de um sistema é, basicamente, o processo de descobrir como um sistema funciona para que se possa refatorá-lo ou mesmo criar um novo sistema tecnologicamente atualizado que cumpra suas tarefas.

Taxionomia de Chikofsky e Cross II

- *Engenharia direta* (*forward engineering*).
  - É o processo tradicional de produção de software que vai das abstrações de mais alto nível até o código executável.
- *Engenharia reversa* (*reverse engineering*).
  - É o processo de analisar um sistema ou seus modelos de forma a conseguir produzir especificações de nível mais alto. É um processo de exame e explicação.
- *Redocumentação* (*redocumentation*).
  - É uma subárea da engenharia reversa. Usualmente trata-se de obter formas alternativas de uma especificação no mesmo nível do artefato examinado.
- *Recuperação de projeto* (*design recovery*).
  - É outra subárea da engenharia reversa. Ao contrário da anterior, a recuperação de projeto vai realizar abstrações a partir dos elementos examinados a fim de produzir artefatos em níveis mais altos do que os examinados.
- *Reestruturação* (*reestructuring*).
  - É uma das formas de refatoração, consistindo em transformar um artefato internamente, mas mantendo sua funcionalidade aparente. Normalmente a reestruturação é realizada para simplificar a arquitetura de sistemas de forma a minimizar futuros problemas de manutenção.
- *Reengenharia* (*reengineering*).
  - É o exame e alteração de um sistema para reconstruí-lo de uma forma diferente. Geralmente inclui alguma forma de engenharia reversa seguida de engenharia direta ou reestruturação.

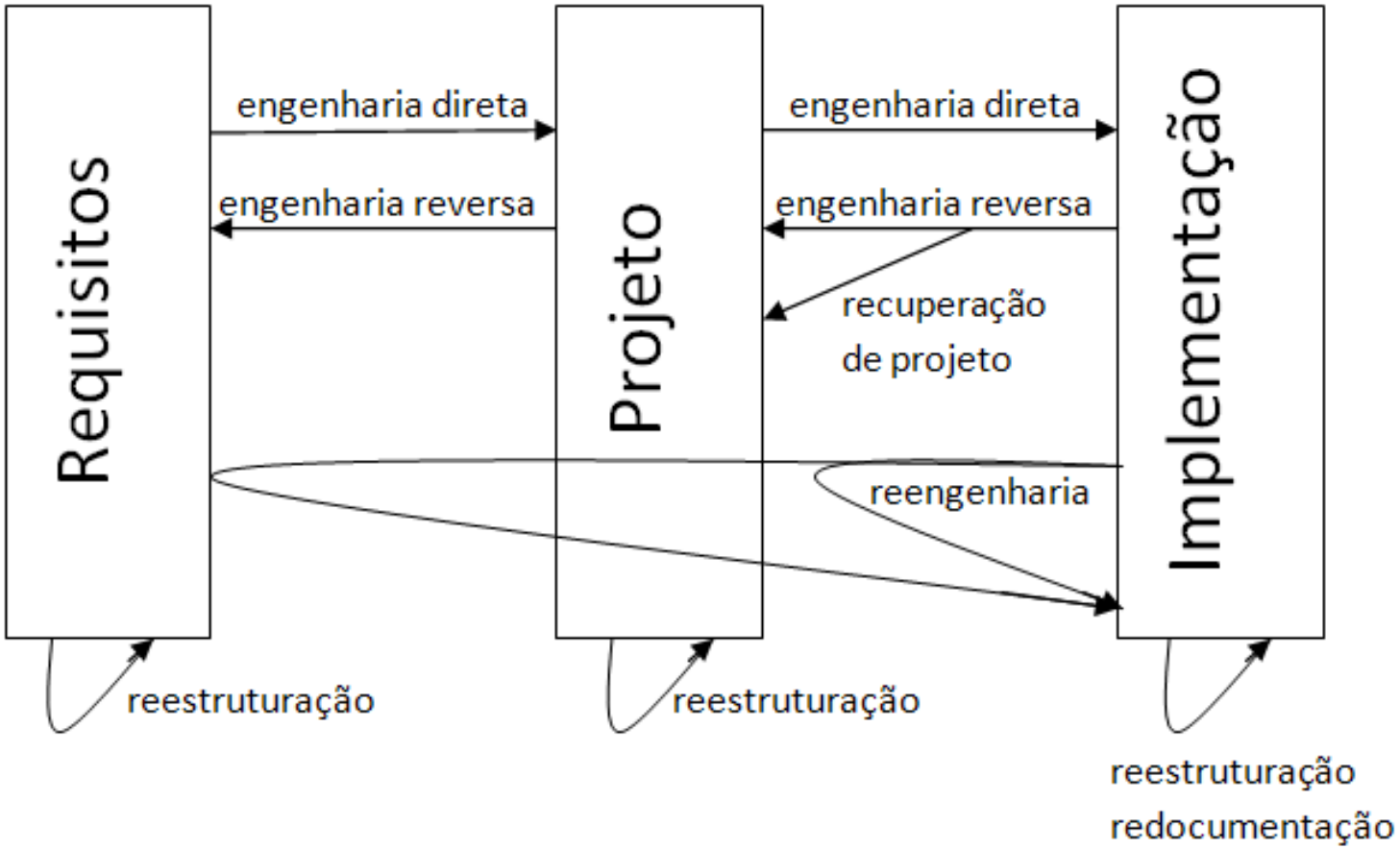

Figura 14-1: Relação esquemática entre os diferentes termos relacionados à engenharia reversa[201].

Engenharia reversa

- De código
  - Fonte
  - Objeto
- De dados

Engenharia Reversa de Dados

- Pode ser considerada como um caso especial da engenharia reversa onde o foco está na localização, organização e reinterpretação do significado dos dados de um sistema.
- Uma das atividades relacionadas à engenharia reversa de dados é a *analise de dados*.
- Esta atividade consiste em recuperar um modelo de dados atualizado (a partir de um sistema em operação), estruturalmente completo e semanticamente anotado.
- Esta atividade é particularmente difícil de ser automatizada.
- A recuperação dos modelos de bancos de dados (quando existem) é relativamente simples, mas estas estruturas frequentemente não contêm informações semânticas e estruturais completas sobre os dados, que acabam sendo diluídas em código executável e documentação.