Engenharia de Software

Anhanguera

AVALIE SUA PROFISSÃO

QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:

NOTAS **9 OU 10** SIGNIFICA QUE VOCÊ INDICA

NOTAS **7 OU 8** SIGNIFICA QUE VOCÊ NÃO INDICA

Anhanguera

Anhanguera

MANUTENÇÃO E EVOLUÇÃO DE SOFTWARE

Após certo tempo de uso, os sistemas operacionais deixam de receber suporte das empresas e isso significa que não serão mais disponibilizadas correções e atualizações para eles. Já os sistemas operacionais livres e de código aberto não são descontinuados, pois anualmente a comunidade evolui o sistema, efetua correções e insere novas funcionalidades.

Percebeu como existem tratativas diferentes para o envelhecimento do software? Os processos evolutivos e de manutenção acompanham o ciclo de vida do sistema, e, em alguns casos, ao atingir o seu limite, ele é descontinuado. Com base no exposto, nesta seção de aprendizagem, você conhecerá a necessidade de evolução e de manutenção de um software, a classificação e os tipos de atividades de manutenção normalmente executadas. Ainda será possível entender como são utilizadas as ferramentas dentro dos processos de manutenção, e ao final como a engenharia reversa e a reengenharia podem contribuir com a manutenção e os processos evolutivos dos softwares.

Antes de iniciarmos tais discussões, compreendamos qual a motivação para estudar esses tópicos. Imagine que seja desenvolvido um aplicativo para rastreamento de PETs com chip de localização. No seu lançamento, existiam poucos clientes, somente método de pagamento via cartão (crédito e débito). Como passar do tempo, o número de clientes aumentou consideravelmente, surgiram novos métodos de pagamento, como o PIX, e novas tecnologias de rastreamentos.

Percebeu como a evolução de outras tecnologias impactou diretamente naaplicação? Para compreender melhor os processos de evolução e de manutenção,entendamos, inicialmente, como ocorre o processo de envelhecimento de umsoftware.

Segundo Vetorazzo (2018), os softwares são sequências lógicas de algoritmos cujo intuito é atender aos objetivos estabelecidos, os quais estão suscetíveis a mudanças de requisitos e ao ambiente que está sendo operacionalizado. Esse processo de envelhecimento é inevitável e exige uma análise de causas, de forma afazer sua evolução e/ou manutenção, garantindo, assim, sua continuidade.

Vetorazzo (2018) define ainda que existem dois tipos de envelhecimento de software, conforme pode ser observado a seguir:

1. **Falha de adequação:** ocorre quando a equipe responsável pela evolução do software comete erros e falhas na adequação ou na implementação dos requisitos, ocasionando muitas vezes a perda da integridade e da confiabilidade da aplicação.

Um exemplo prático: um software para conversão de formatos de vídeo, que funcionava em uma versão do sistema operacional, não tem compatibilidade com a versão mais recente desse mesmo sistema operacional.

Caso o software seja utilizado por um usuário comum, certamente ele procurará outra solução. Mas, se uma empresa utiliza esse software em suas operações e em determinado momento os computadores são trocados e vêm com um sistema operacional não compatível com o software de conversão de vídeo, isso será um grande problema.

2. Falha na mudança: é quando existe alguma atualização, manutenção ou implementação que impacta negativamente outras funcionalidades que já estavam em pleno funcionamento.

Um exemplo prático: um e-commerce possui apenas um gerador de boleto em funcionamento, mas, devido a solicitações dos usuários, precisará apresentar outras formas de pagamento. Para isso, foi implementado (de forma incorreta) uma API de módulo de pagamentos. O impacto desse erro foi o sistema gerador de boletos e o módulo de pagamentos diversos não funcionarem. Isso ocorreu devido ao desconhecimento da estrutura do gerador de boletos.

Caro aluno, percebeu a necessidade de evolução e manutenção dos softwares? São muitas variáveis a serem observadas: evolução dos sistemas operacionais, modos de consumo, novos meios e métodos de pagamento, segurança e diversos outros pontos. Mas, como prever a expectativa de envelhecimento do software? Isso só ocorreria se tivéssemos informações de todas as empresas de tecnologia da informação, as quais poderiam, por meio de uma alteração, impactar no funcionamento do sistema de alguma forma, o que é impossível.

Porém, a compreensão da classificação e dos tipos de atividades de manutenção de softwares trará uma vantagem de recuperabilidade nas atividades de ajustes, manutenções, evoluções e adequações do sistema em casos de inconformidades.

Com isso, Pressman e Maxim (2016) defendem que há diferentes intenções para evoluir um sistema que não apenas correção de falhas, bugs, erros e inconformidades. Para compreender como são classificadas as atividades de manutenção, observe o Quadro 4.6.

CLASSIFICAÇÃO	CONCEITO	APLICAÇÃO
Adaptativa	São modificações necessárias para estar de acordo com novos requisitos, os quais podem ser provenientes de leis, regras, ameaças, meios ou métodos novos.	Recentemente (no ano de 2020) o Banco Central autorizou as instituições financeiras a utilizarem o PIX como método de pagamento. Isso exigiu uma adaptação do sistema de internet banking para que essa nova funcionalidade estivesse disponível aos clientes.

Ver anotações

CLASSIFICAÇÃO	CONCEITO	APLICAÇÃO
Corretiva	Sua função é corrigir falhas ou qualquer outro aspecto que seja motivo de degradação dos serviços do software. Além disso, a manutenção corretiva pode ocorrer antes, nas fases de desenvolvimento, ou depois, com o software já em funcionamento.	Nas eleições municipais de 2020, para que as pessoas que não foram votar pudessem justificar o voto, o governo federal disponibilizou um aplicativo para mobile conhecido como e-título. No primeiro turno, ele apresentou problemas desde a sua instalação até o processo de realizar a justificativa. A única funcionalidade em conformidade era a consulta da situação do eleitor quanto à Justiça Eleitoral. No segundo turno, o aplicativo teve de passar por uma manutenção corretiva para que fossem efetuados os ajustes necessários.

CLASSIFICAÇÃO	CONCEITO	APLICAÇÃO
Evolutiva	A manutenção evolutiva tem o objetivo de inserir novas funcionalidades no sistema.	Os chats eram um recurso bastante presente no e-commerce para atendimento aos clientes. Uma evolução desse tipo de atendimento é, em vez de ter um colaborador de plantão para atendimento no chat, utilizar atendimento virtual. Para isso, foram desenvolvidos algoritmos que utilizam inteligência artificial, os quais, com a evolução tecnológica, conseguem responder grande parte das dúvidas dos clientes.

O conhecimento da classificação dos tipos de manutenção, em termos profissionais, permite que tanto um desenvolvedor quanto um gestor se posicione quanto às reais necessidades dentro da estrutura do sistema, facilitando o direcionamento de recursos dentro do ciclo de vida de desenvolvimento de software. Para tal, é necessário conhecer os processos e as ferramentas utilizados na manutenção de software, os quais são recursos de extrema importância para que, de fato, os processos sejam otimizados.

TÉCNICAS E FERRAMENTAS UTILIZADAS NA MANUTENÇÃO DE SOFTWARES

Segundo Pressman e Maxim (2016), os processos e ferramentas utilizados na manutenção de softwares têm como objetivo obter métodos que sejam utilizados como boas práticas e garantir conformidade aos requisitos.

Para entender melhor esse tópico, observe algumas das técnicas e ferramentas mais utilizadas a seguir.

CODIFICAÇÃO

É tida como uma parte muito importante na manutenção. A qualidade do código de programação deve possuir legibilidade, ou seja, deve ser fácil e legível. Ainda que as técnicas de indentação e os comentários de código devam estar presentes nas principais linhas da escrita do sistema e as suas funcionalidades. Observe na figura a seguir uma forma de utilizar a codificação com boas práticas.

```
194    <script type="text/javascript">
195      function limpa_formulario_cep() {
196              //Limpa valores do formulário de cep (rua, bairro, cidade e estado).
197              document.getElementById('rua').value=("");
198              document.getElementById('bairro').value=("");
199              document.getElementById('cidade').value=("");
200              document.getElementById('estado').value=("");
201          }
```

O trecho de código foi escrito em JavaScript (não sendo necessário saber programar nessa linguagem). Mas repare em algumas boas práticas, que podem ser observadas:

Nomes: os nomes de variáveis devem sempre remeter ao que será usado, exemplo: rua, bairro, cidade e estado (em amarelo nas linhas 197, 198, 199 e200); outra boa prática é nomear as funções a serem realizadas. Por exemplo: na linha 195 existe uma função (function) para deixar os campos sem nenhum valor dentro, a qual é chamada de limpa_formulario_cep.

Comentário: essa é uma técnica muito importante, mas deixada de lado por alguns desenvolvedores quando as práticas do desenvolvimento não impõem mais dificuldades, o que acaba complicando a execução da manutenção efetuada por outros desenvolvedores. Um exemplo pode ser observado na Figura 4.11, pois, na linha 196, é explicado o que a função irá executar naquele trecho de código.

Indentação: são espaços utilizados para demonstrar o nível hierárquico das funcionalidades dentro do algoritmo. Ainda utilizando como base a Figura 4.11,repare que na linha 195 existe uma estrutura de função em que se abre uma chave, a qual é fechada na linha 201. Para a indentação, as linhas de 169 a 200 sofreram um recuo por meio de um TAB no teclado. Isso organiza o código-fonte e facilita o processo de manutenção.

VERSIONAMENTO

São documentações que determinam as modificações e as atualizações dos softwares. Elas podem ser feitas por dois meios:

Numeração: trata-se de um sistema que por meio de uma numeração demonstra a sua versão. O objetivo é especificar as suas características por meio desse sistema numérico. Para um exemplo vamos supor que um determinado software esteja na versão 10.2.4.3 e que cada um desses números tenha um significado, conforme o que se explica adiante:

10: indica que houve dez mudanças significativas no sistema. Essa numeração muda cada vez que o software faz uma evolução que promova mudanças de grande escala.

2: esse número indica que foram adicionadas novas funcionalidades no sistema. Por exemplo, um software possui, na nova versão, a funcionalidade de impressão.

4: esse número indica a quantidade de correções de bugs e falhas. Por exemplo, existia uma falha de envio de confirmação de inserção de produtos no carrinho de compras em um e-commerce na versão 10.2.3.3,que foi corrigido na versão 10.2.4.3.

3: são correções graves relacionadas a incidentes de segurança. Por exemplo, determinado aplicativo expunha os dados dos usuários na versão 10.2.4.2, e a falha foi corrigida na versão 10.4.2.3.

Comentário: uma prática muito comum é adicionar, nas primeiras linhas do arquivo no qual foram feitas as manutenções, informações como: data, objetivo da manutenção, modificações efetuadas e demais comentários úteis para manutenções futuras.

ESTRUTURAR O CÓDIGO PARA EVOLUÇÃO

A atividade de planejamento requer uma reflexão dos limites do software. Dessa forma, permite projetar o código para permitir sua evolução, seja ela por meio de adaptações, mudanças, ou por qualquer outro meio, o importante é que sua estrutura permita a evolução.

Caro aluno, as discussões e os exemplos acerca dos processos e das ferramentas de manutenção de software em aplicações profissionais estão no cotidiano das práticas de desenvolvimento de sistemas e são essenciais para que se possam fazer manutenções corretamente e para que seja realizado um trabalho a ser continuado por outros profissionais com habilidades técnicas.

Além disso, em termos profissionais, as discussões acerca da classificação e dos tipos de atividades de manutenção podem levá-lo a compreender os momentos nos quais as manutenções adaptativas (ajustar a novos requisitos), corretivas(correções de bugs e falhas) e evolutivas (adicionar novas funcionalidades) são operacionalizadas, em ambiente de desenvolvimento de sistemas, bem como assuas finalidades.

REENGENHARIA DE SOFTWARE

Segundo Pádua (2019), o processo de reengenharia de software é uma forma de reorganizar e/ou modificar o sistema a fim de fazê-lo apresentar um desempenho aceitável. Alguns fatores como falta de evolução de software ou excesso de contínuas mudanças (pior ainda quando promovidas por equipes diferentes)acabam degradando os serviços do sistema.

Nesse momento, tentar encontrar novas soluções em um sistema comprometido pode não gerar o resultado desejado e ainda comprometer muitos recursos. Dessa forma, fazer uma reconstrução com a correção de erros e falhas, além de adequaras evoluções necessárias é uma ótima saída em busca de soluções.

Ainda de acordo com Pádua (2019), a reengenharia de software tem o objetivo de reimplementar sistemas legados, em que se busca:

Melhorar a manutenção: ao se reestruturar o sistema, as futuras manutenções serão mais fáceis para efetuar a manutenção, visto que os antigos erros devem ser corrigidos na nova versão.

Redocumentar o software: quando o software é reestruturado, adocumentação é refeita, permitindo, assim, que novas informações sejamcolocadas nos scripts.

Reestruturar o sistema: as estruturas que funcionavam à base de reparos ecorreções podem ser repensadas, o que permitirá a construção de um sistemacom otimizações e atendimento aos requisitos.

Quando o software é produzido, existem funcionalidades e módulos que requerem grandes esforços para serem desenvolvidos, e, embora o sistema passe por testes, ao longo do tempo ele pode apresentar falhas. Porém, no processo de reengenharia de software, se o entendermos como uma releitura dos algoritmos, podemos dizer que a chance de erro será menor. Com base nisso, Pádua (2019)defende que os processos de reengenharia possuem riscos reduzidos, pois alguns problemas já haviam sido tratados.

Esses processos são apoiados em metodologias de operacionalização. Para tal, Pressman e Maxim (2016) defende o modelo apresentado na Figura 4.12.

Anhanguera

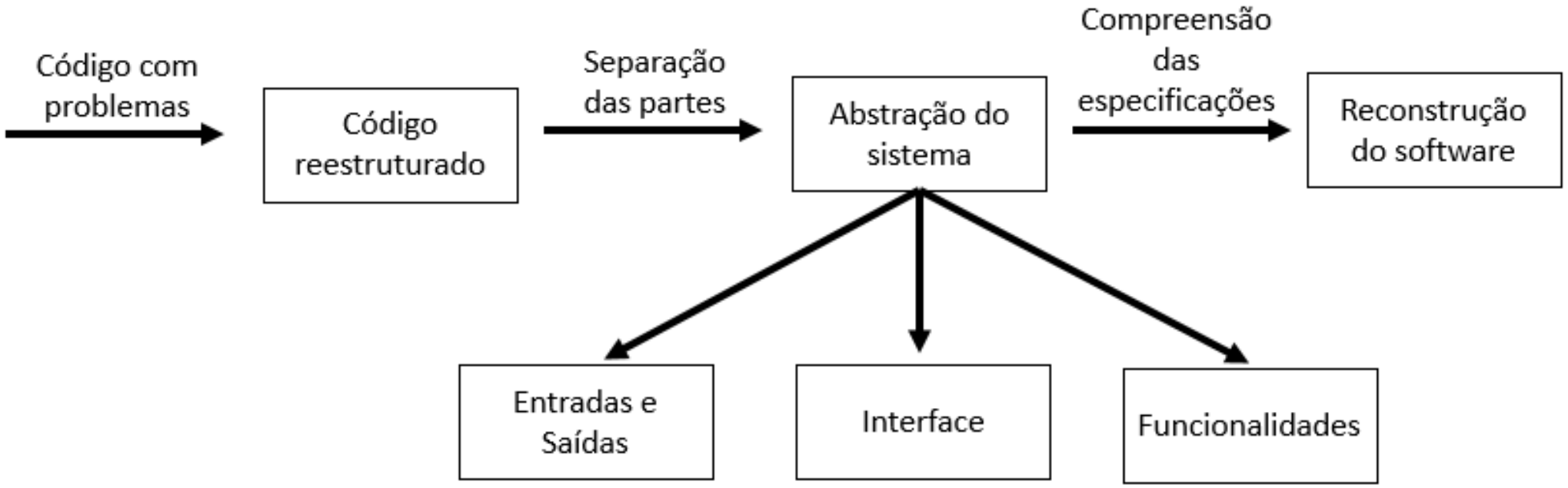

Análise de documentação
Reconstrução de documentação
Engenharia reversa
Reestruturação do código
Reestruturação dos dados

Observe que, ao longo dessa discussão, esses processos foram explorados e exemplificados. Porém, um novo termo muito importante quanto à manutenção e à evolução dos softwares foi citado no modelo representado na Figura 4.12:

engenharia reversa.

Claramente, quando a engenharia reversa é utilizada em hardware, a técnica fica visualmente mais fácil de ser compreendida, pois imaginamos os componentes sendo desmontados, o que nos permite compreender a ordem de montagem e, ainda, analisar cada componente em separado a fim de entender o seu objetivo e funcionamento.

Mas como a engenharia reversa é tratada a nível de software? Antes de responder esse questionamento, observe a Figura 4.13.


Código com problemas
Código reestruturado
Separação das partes
Abstração do sistema
Compreensão das especificações
Reconstrução do software
Entradas e Saídas
Interface
Funcionalidades

Observe que gradativamente os métodos de engenharia reversa tratam de reestruturar o código, permitindo, assim, a compreensão das funcionalidades(entradas e saídas, interface e funcionalidades).

Segundo Pádua (2019), ao se utilizar os processos e técnicas de engenharia reversa, é possível visualizar o software de diferentes maneiras, tais como:

Nível de implementação: permite a compreensão das características e especifidades da linguagem de programação utilizada no processo de implementação.

Nível estrutural: permite a compreensão dos diferentes módulos e funcionalidades e das suas respectivas dependências funcionais. Essa abstração se dá por meio da análise da estrutura da linguagem de programação utilizada.

Nível funcional: permite que as partes que compõem os sistemas sejam compreendidas; com ênfase na lógica utilizada no desenvolvimento.

Nível de domínio: permite compreender onde o software é utilizado.

Com isso, em termos profissionais, tanto a reengenharia quanto a engenharia reversa possuem técnicas relativamente simples, mas que não são tão fáceis de operacionalizar. As técnicas são comumente utilizadas em práticas cotidianas para gerar diminuição no tempo de desenvolvimento. Dessa maneira, os resultados tendem a ser melhores visto que boa parte das funcionalidades já foram desenvolvidas.

O versionamento de software é uma técnica que agrega informações importantíssimas, as quais auxiliam na identificação das alterações promovidas pelos desenvolvedores de software. Porém, para que isso ocorra, é necessário conhecer as partes que compõem essa numeração.

Se um sistema teve a sua versão alterada de 2.3.4.5 para 2.4.4.5, ocorreu uma alteração na:

a. Estrutura como um todo, fazendo com que o software fosse totalmente modificado.

b. Implementação de novas funcionalidades dentro do sistema.

c. Correção de falhas e bugs encontrados na sua utilização.

d. Segurança, devido à vulnerabilidades do sistema.

e. Forma de acessar o sistema.

As atividades de manutenção de software estão presentes no ciclo de vida do desenvolvimento de software com o intuito de, no início, evitar que o sistema chegue aos usuários com falhas, e, se o produto de software já estiver em uso, fazer as devidas manutenções a fim de promover sua evolução. Quanto à classificação e tipos de manutenção, observe as afirmativas a seguir:

I. Manutenção adaptativa significa que determinada funcionalidade migrará para outro sistema, de forma a se adaptar em um módulo, por exemplo.

II. Manutenção corretiva diz respeito às atividades que visam resolver falhas, erros e demais motivos que possam estar degradando o software.

III. Manutenção evolutiva é utilizada para melhorias de funcionalidades já implementadas.

Assinale a alternativa CORRETA:

a. Está correta apenas a afirmativa I.

b. Está correta apenas a afirmativa II.

c. Está correta apenas a afirmativa III.

d. Estão corretas apenas as afirmativas I e II.

e. Estão corretas apenas as afirmativas I e III.