UNIP EaD

Projeto Integrado Multidisciplinar

Cursos Superiores de Análise e Desenvolvimento de Sistemas

CONCEITOS DENTRO E FORA DE UM SOFTWARE

UNIP EaD Polo Guaratinguetá-SP

2020

UNIP EaD

Projeto Integrado Multidisciplinar

Cursos Superiores de Análise e Desenvolvimento de Sistemas

CONCEITOS DENTRO E FORA DE UM SOFTWARE

Vinicius Da Silva Nogueira RA-1945946

2º Bimestre

UNIP EaD Polo Guaratinguetá-SP

2020

Resumo

Abstract

SUMÁRIO

Introdução

**1.Apresentação do problema**

Devemos desenvolver um software para um Colégio Vencer Sempre. O objetivo desse software é organizar e administrar esses aparelhos eletrônicos.

**2. Agentes econômicos que vão atuar na empresa de software**

Os agentes econômicos que efetua desse software é uma empresa (Colégio), que precisa desse aplicativo,

Junto com a empresa tem os professores que atuam também nesse software, considerado famílias

**3.Projetando o Software**

**3.1 Engenharia de software em Programação**

Minha microempresa decidiu definir algumas funções da engenharia de software, porque sem ela bem estruturada para atender seus usuários, não conseguimos ter uma boa qualidade de software que realmente almejamos.

Para fazer um sistema operacional, por exemplo, não é só código, tem muita coisa envolvida, como ciclo de vida, documentação, versões e etc.

Lidando só com o código o programador deve pensar que é mais produtivo, cada caso é um caso, se for um projeto mais complexo é preciso usar a engenharia de software a seu favor. Num projeto mais fácil, uma documentação e um backup é sempre aconselhável usar.

Documentação é algo para clarear o que deve ser feito em um software de editor de texto, tipo o Word, ou até mesmo em um papel. Manutenção também é preciso de documentos para observar o que já foi feito, para consertar ou para evoluir

o seu software.

Um repositório é indiscutível em seus projetos, ás vezes o programador faz um monte de mudanças em seu software, e quando o cliente fala que está no caminho certo, cria um zip e deixa guardado. Mas isso não é bom, e se formatar o computador? Em qualquer componente que seja hardware, pode-se perder, por exemplo: um pen-drive. Então qual é o lugar mais adequado de se guardar um software, em um controle versões distribuído e repositório local como o git. e o repositório remoto que é o GitHub.

**3.2. Condições Funcionais, Não Funcionais e Regras de Negocio:**

**3.2.1 Funcional**

Um resumo das condições funcionais é, entrada, o processo e a saída de um programa.

Entrada de um programa para o problema é:

- Número de aparelhos;

- Tipos de aparelho;

- Quais serão os dias da semana que vão ser utilizado os aparelhos;

- Dias da semana;

- Disciplinas;

- Horários das disciplinas.

Processos:

Vamos analisar:

- Quais aparelhos são utilizáveis;

- Está sendo consertado;

- Não tem mais conserto.

O aparelho que está pronto para o uso é passado pela segundo processo.

Os professores vão observar qual o horário da disciplina deles e vão utilizar o aparelho. Um professor de utilizar um aparelho por vez.

Quando terminar de utilizar o aparelho, o professor que está usando vai guarda-o e no software o professor vai tirar o aparelho da sua disciplina e colocar onde os aparelhos prontos estão.

Saída:

Como demonstramos acima quando software desligar, os aparelhos que não foram “alugados”, no dia seguinte podem ser utilizados normalmente e os que foram alugados ficam agendados até que algum dos professores os retire.

**3.2.2 Não Funcional**

Condições não funcionais são um padrão da ISO 9126, que são seis características e suas sub características, vamos falar algumas qualidades que tem nesse software:

1. Funcionalidade: Conjunto de funções que satisfazem as necessidades do software. Colégio pediu um software que resolva seus problemas, então importante uma especificação bem detalhada do que foi pedido para que os programadores façam um melhor trabalho. Maior erro aqui é no entendimento dos requisitos.

1.1. Adequação: Funções especificadas estão presentes no software.

1.2. Precisão: Produto gera resultados esperados.

2. Confiabilidade: Desempenho adequado nas condições estabelecidas.

2.1. Maturidade: Evitar falhas decorrentes no software.

2.2. Tolerância a falhas: Software deve manter um nível de desempenho em caso de defeitos no software.

3. Usabilidade: Capacidade do produto de ser compreendido, operado e atraente ao usuário.

3.1. Capacidade do produto de software de possibilitar que o usuário aprenda sobre a aplicação.

3.2. Software deve ser atraente para o usuário, mas não é o principal.

4. Eficiência:

4.2. Comportamento em relação a recursos: Quantidade de recursos utilizados.

**3.2.3. Regras de Negócio**

Oque é regras de negócio:

Regras do negócio respondem às necessidades da aplicação, modelam a reação dos eventos que ocorrem no mundo real, com efeitos tangíveis no conteúdo da base de dados, assim como encapsula o comportamento reativo da aplicação para tais eventos. CERI & FRATEMALE (1997) apud LEITE & LEONARDI (1998),

Regra de negócios destaca como os clientes, no caso são os professores estão sendo tratados com essa nova tecnologia que é o software. Como recursos são gerenciados, no caso dos aparelhos: Pronto para o uso, em manutenção ou não tem mais conserto. Se não tem mais concerto, qual seria a melhor escolha a se fazer numa questão dessa? Isso é muito importante!

**3.3. MPS.BR na Empresa**

**3.3.1 Metodologias**

Uma estrutura do modelo se divide em 4 componentes, 7 níveis de maturidade e 19 processos distribuídos nos níveis definidos.

Os modelos são: Referência de software, referências para serviços, método de avaliação, modelo de negócio.

Os níveis de maturidade são de G (parcialmente gerenciado) até a letra A(otimizado).

**3.3.2 Por que MPS.BR**

Esta concorrência seria entre CMMI, ISO e MPS.br, que é reconhecida nacionalmente e internacionalmente.

Vamos juntar os fatos que a empresa é pequena, não tem um grande mercado ainda e a microempresa está no Brasil.

MPS.BR é uma ótima opção porque o objetivo mesmo da MPS.BR é de incentivar as pequenas e médias empresas brasileiras de produção de software a implantar um modelo de qualidade de melhoria de processo com custos mais acessíveis à realidade brasileira.

Mas o seu reconhecimento como selo de qualidade de software está limitado ao território brasileiro.

**3.4. Roteiro de Testes**

Iremos fazer os testes nas condições funcionais e regras de negocio.

|  |  |  |
| --- | --- | --- |
|  | FUNCIONAL |  |
| Passo para Execução | **Dados de entrada** | **Resultados Esperados** |
| Quantos aparelhos existem na escola | Numero | N**°** de aparelhos |
| Qual os nomes dos aparelhos | Nome | Nomes dos aparelhos |
| Vê se o N° de aparelhos é igual ao N° de nomes | **-** | o 1° na tabela é igual ao 2° em questão de números |
| Dispositivos prontos para o uso | Numero de aparelhos prontos | Exibe na tela os aparelhos que estão prontos |
| Dispositivos que estão em manutenção | Numero de aparelhos em manutenção | Exibe na tela os aparelhos que estão em manutenção |
| Numero de aparelhos que podem ser consertados | - | Numero de aparelhos que podem ir para manutenção |

**4.Orientação a objetos**

Orientação a objeto se aplica o conceito da modelagem em linguagem de programação. Vamos falar sobre alguns desses conceitos sobre modelar.

Primeiramente vamos falar sobre Objetos:

Objetos podem ser considerados uma imitação do comportamento intrínseco de entidades reais.

Classe:

Iremos abordar Abstração para entendermos classe. Abstração é algo que se possa imaginar a partir de algo concreto. Um exemplo, Um objeto da vida real para abstrair e elaborar um software.

O resultado dessas abstrações é desenvolvido pelas classes.

Os objetos que vimos acima com estrutura semelhante são definidos por meio de classes.

Herança:

Tem uma classe(pai), ou seja, a sua base, e uma classe(filha) que é direcionada para sua base, sendo adicionados a essa nova classe apenas atributos e métodos que não existam na classe original.

Polimorfismo:

O polimorfismo está relacionado com conceito de herança. Por exemplo: Tem uma classe chamada pessoa(pai). E dois (filhos), funcionário e cliente. O software consegue identificar por causa de seus atributos.

CONCLUSÃO

REFERÊNCIAS