Universidade Paranaense Unipar

Engenharia de Software

Engenharia de Software de Reuso

Alan Henrique Ribeiro – 00233006
Pedro Henrique Pinheiro Rodrigues - 00236649
Eduardo Henrique Monteiro - 00233842
Ricardo Marchesan Felipin - 00238795
Rafael da Silva Lima- 239884
Renan Alves Peregrino- 00237721
Adrian Lucas Toderke - 00230994
Lucas Braga de Lima - 00243921
Allan Matheus Pereira - 00239726
Gustavo Henrique Nava - 00243826
Renan Augusto Zeferino Geraldo - 00237604
João Vitor Florisvaldo Vieira - 00241738

CASCAVEL

2022

Sumário

Por que usar?

- Facilidade
- Agilidade
- Rapidez no desenvolvimento
- Confiabilidade
- Custo benefício

Por que não usar?

- Maior custo de Manutenção
- Falta de ferramentas de suporte
- Sindrome do "Não fui eu"
- Falta de compatibilidade dos códigos reusados
- Adaptar e atualizar os compunentes utilizados
- Dificuldade na gerência das versões
- Se um software tem requisitos muito rigorosos é imposivel usar estratégias de Reuso com eficiência

Conceitos e técnicas utilizadas para o reuso de software

- Consistem basicamente no reaproveitamento de partes previamente desenvolvidas qualificadas e armazenadas.
- Para reusar um software, não é necessario usar o código do mesmo obrigatoriamente.
- Também pode ser utilizada a ideia ou a arquitetura do mesmo.

Plataformas comumente utilizadas

- Várias plataformas de controle de versão de softwares open source são utilizadas para encontrar códigos que podem servir para o seu projeto.
- Exemplos como Github, Gitlab e Bitbucket, são muito utilizados para importação de bibliotecas open source práticas e funcionais.

Benefícios

I. **Confiança Aumentada**

- Softwares reusados que foram testados e aprovados em sistemas em funcionamento são mais confiaveis do que um software novo.

II. **Risco de Processo Reduzido**
- Uma vez que boas partes do software são reusadas, é mais facil calcular custos de processo, já que o custo dos softwares é conhecido, o que diminui a margem de erro.

III. **Uso Eficaz de Especialistas**
- Ao invés de repetir o mesmo trabalho, desenvolvedores experientes podem desenvolver codigos reusaveis que englobem o seu conhecimento.

IV. **Conformidade com Padrões**
- Alguns padrões como UI podem ser desenvolvidos como módulos reutilizaveis. O uso de Interfaces padrão nas aplicações melhora a confiança dos usuários, já que diminui a quantidade de erros que o mesmo comete por ser apresentado a interfaces familiares.

V. **Desenvolvimento Acelerado**
- O reuso de um software acelera o tempo de desenvolvimento e entrega de uma aplicação, pois pode reduzir o tempo de desenvolvimento e validação.

Problemas

I. **Maiores Custos de Manutenção**
- Caso o código fonte de um sistema reusavel não esteja disponível, o custo da manutenção ser maior pelo fato dos elementos reusados do sistema não são compatveis com atualizações

II. **Falta de Ferramentas de Suporte**
- Algumas ferramentas de desenvolvimento podem não suportar o desenvolvimento com reuso. O processo assumido por essas ferramentas pode não considerar o reuso, dificultando a implementação.

III. **Sindrome do "Nao-inventaram-aqui"**
- Alguns engenheiros de softare preferem reescrever componentes, por acreditar que é possivel melhorá-los. Isso tem a ver, parcialmente, com a falta de confiança em softwares que não são originais do proprio.

IV. **Criação e Manutenção de uma Biblioteca**
- Criar uma biblioteca de componentes reutilizaveis é um processo caro e trabalhoso, já que é necessário um processo de desenvolvimento personalizado para que seja possivel utilizar a biblioteca.

V. **Encontrar, Entender e Adaptar Componentes Reusaveis**
- Muitos componentes precisam ser descobertos, entendidos, e muitas vezes, adaptados para poderem ser utilizados em projetos variados.

Perguntas

1. **Cascata**
   - **a)** Como é feita a Documentação?
   - **b)** Como é usado em Sistemas Críticos?
   - **c)** Por que eu usaria em Projetos Grandes?
   - **d)** Como é feita a divisão dos estados distintos do projeto?
   - **e)** O que é feito quando aparecem erros no sistema?
   - **f)** O que o congelamento do sistema acarreta no desenvolvimento final?
   - **g)** Como é feito o Feedback?

2. **Incremental**
   - **a)** O que é feito quando aparece danos críticos no sistema?
   - **b)** Com várias incrementações feitas o que começa acontecer ?
   - **c)** Por ser feito várias versões do sistema como é feita a documentação ?
   - **d)** Como os gerentes do projetos ficam sabendo das atualizações ?
   - **e)** Como é usado em sistemas críticos ?
   - **f)** Como é feito o controle dos feedback exaustivos ?

3. **Reuso**
   - **a)** Como é feito a analise de componentes ?
   - **b)** Qual a diferença entre desenvolvimento e modificação de requisitos ?
   - **c)** Como é o framework?
   - **d)** que é feito quando um software não pode ser adquiro ?
   - **e)** Como o sistema de COTS é integrado ?

4. **Rupi**
   - **a)** Como é feito a documentação ?
   - **b)** que é feito quando tem correlação assuntos técnicos
   - **c)** que é feito nas 4(quatro) fazes distintas ?
   - **d)** Quais são as boas praticas na especificação e projeto ?
   - **e)** que seria a prototipação ?

Bibliografia

- Ian Sommerville. Engenharia de Software, 9º Edição. Pearson Education, 2011.
  - Seção 2.1 Modelos de processo de software
  - Seção 2.1.1 O modelo em Cascata
  - Seção 2.1.2 Desenvolvimento Incremental
  - Seção 2.1.3 Reuso
  - Seção 2.4 Rational Unified Process (RUP)