Reuso de Componentes

Sua ideia foi proposta a mais de 40 anos, porém só começou a ser mais praticada a partir dos anos 2000.

* Menores custos, entregas mais rápidas, maior qualidade.

Níveis de reuso:

* Aplicação
* Componentes
* Objetos/Funções

A engenharia de software baseada em reuso é uma estratégia na qual o processo de desenvolvimento é voltado para o reuso de software. Hoje é empregado extensivamente no desenvolvimento de novos sistemas de negócios.

Uma óbvia vantagem do reuso de software é que os custos gerais do desenvolvimento são menores. Há menos componentes de software para especificar, projetar, implementar e validar.

Benefícios do reuso:

* Desenvolvimento acelerado
* Uso eficaz de especialistas
* Maior dependabilidade (o software a ser reutilizado foi testado e aprovado em sistemas em funcionamento, deve ser mais confiável que um novo. Seus defeitos de projeto e implementação devem ter sido descobertos e corrigidos).
* Custos de desenvolvimento mais baixo
* Menos riscos para o processo (quando o custo diminui, os riscos de aumentar os custos por conta de erros no projeto diminuem)
* Conformidade com padrões

Problemas com o reuso:

* Criar, manter e usar uma biblioteca de componentes
* Encontrar, entender e adaptar componentes
* Maiores custos de manutenção (se o código-fonte de um sistema de software ou componente reusado não estiver disponível, então os custos de manutenção podem ser mais altos porque os elementos reutilizados podem se tornar incompatíveis com as mudanças feitas no sistema)
* Falta de suporte na ferramenta
* Síndrome do “não inventado aqui”.

Abordagens que apoiam o reuso de software:

* Frameworks de aplicação
* Padrões de arquitetura
* Engenharia de software baseada em componentes
* Padrões de projeto
* Sistemas ERP
* Bibliotecas de programa
* Sistemas orientados a serviço

Cronograma de desenvolvimento:

* Se o software tiver de ser desenvolvido rapidamente deve-se tentar reusar sistemas completos em vez de componentes individuais.

O tempo de vida previsto para o software:

* Para desenvolver um sistema de vida longa, é necessário se preocupar/concentrar na capacidade de manutenção do sistema. Se não houver acesso ao código-fonte dos componentes pode ser preferível evitar componentes de prateleira e sistemas de fornecedores externos.

Formação e habilidade do time de desenvolvimento:

* É necessário concentrar o reuso nas áreas que os times possuem muita experiência.

A criticidade do software e seus requisitos não-funcionais:

* Para um sistema crítico, que precisa ser certificado por um regulador externo, pode ser necessário criar um caso de segurança (safety) ou de segurança da informação (security) para esse sistema. Isso é difícil se não for possível ter acesso ao código-fonte.