Resumo Engenharia de Software – NP2

Prototipação

O que é um protótipo?

Um protótipo é uma versão inicial de um sistema de software. Usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções

Um protótipo de software pode ser usado em um processo de desenvolvimento de software para ajudar a antecipar as mudanças que podem ser requisitadas. Protótipos do sistema permitem aos usuários ver quão bem o sistema dá suporte a seu trabalho. Além disso, o desenvolvimento do protótipo pode revelar erros e omissões nos requisitos propostos.

Enquanto o sistema está em projeto, um protótipo do sistema pode ser usado para a realização de experimentos de projeto visando à verificação da viabilidade da proposta.

Prototipação também é uma parte essencial do processo de projeto da interface de usuário. Devido à natureza dinâmica de tais interfaces, descrições textuais e diagramas não são bons o suficiente para expressar seus requisitos. Portanto, a prototipação rápida com envolvimento do usuário final é a única maneira sensata de desenvolver interfaces gráficas de usuário para sistemas de software.

Estabelecer Objetivo

Os **objetivos** da prototipação devem ser explicitados desde o início do processo.

Estes podem ser o desenvolvimento de um sistema para prototipar a interface de usuário, o desenvolvimento de um sistema para validação dos requisitos funcionais do sistema ou o desenvolvimento de um sistema para demonstrar aos gerentes a viabilidade da aplicação.

Definir funcionalidade

O próximo estágio do processo é decidir o que colocar e, talvez mais importante ainda, o que deixar de fora do sistema de protótipo. Para reduzir os custos de prototipação e acelerar o cronograma de entrega, pode-se deixar alguma funcionalidade fora do protótipo.

Avaliar Protótipo

Durante esse estágio, provisões devem ser feitas para o treinamento do usuário, e os objetivos do protótipo devem ser usados para derivar um plano de avaliação. Os usuários necessitam de um tempo para se sentir confortáveis com um sistema novo e para se situarem em um padrão normal de uso.

Problemas

Um problema geral com a prototipação é que o protótipo pode não ser necessariamente usado da mesma forma como o sistema final. O testador do protótipo pode não ser um usuário típico do sistema ou o tempo de treinamento durante a avaliação do protótipo pode ter sido insuficiente, por exemplo.

Às vezes, os desenvolvedores são pressionados pelos gerentes para entregar protótipos descartáveis, especialmente quando há atrasos na entrega da versão final do software. No entanto, isso costuma ser desaconselhável:

1. Pode ser impossível ajustar o protótipo para atender aos requisitos não funcionais, como requisitos de desempenho, proteção, robustez e confiabilidade, que foram ignorados durante o desenvolvimento do protótipo.

2. Mudanças rápidas durante o desenvolvimento inevitavelmente significam que o protótipo não está documentado. A única especificação de projeto é o código do protótipo. Para a manutenção a longo prazo, isso não é bom o suficiente. Prototipação

3. As mudanças durante o desenvolvimento do protótipo provavelmente terão degradado a estrutura do sistema. O sistema será difícil e custoso de ser mantido.

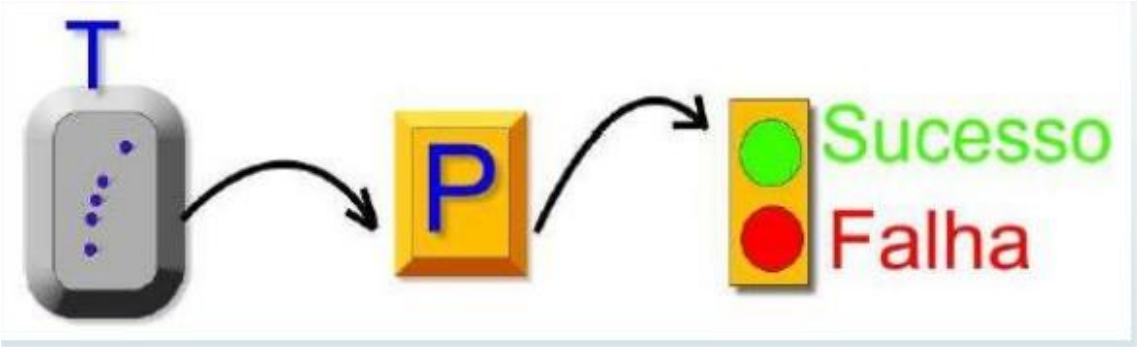
4. Padrões de qualidade organizacional geralmente são relaxados para o desenvolvimento do protótipo.

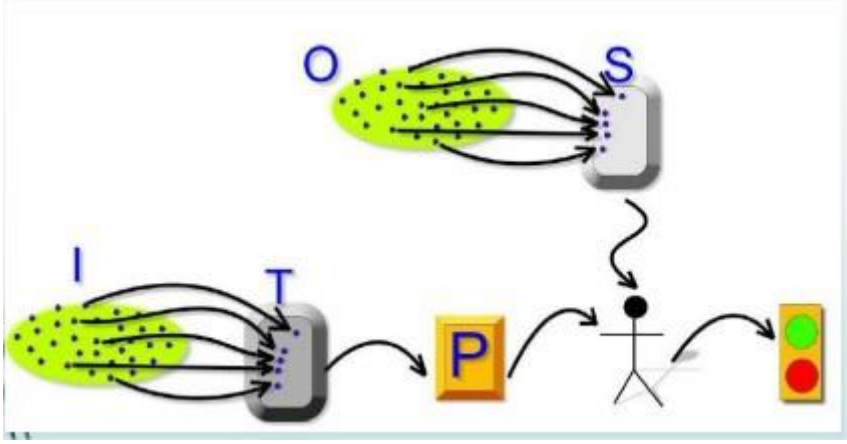
Protótipos não precisam ser executáveis para serem úteis. Maquetes em papel da interface de usuário do sistema podem ser eficazes em ajudar os usuários a refinar o projeto de interface e trabalhar por meio de cenários de uso.

Teste de Software

O que é teste de software?

Atividade de executar um programa e verificar se o seu comportamento é o esperado com o objetivo de revelar erros.





I: Domínio de Entrada: conjunto de todos os dados que o programa deve tratar

O: Domínio de Saída: todos os possíveis resultados que o programa deve fornecer

T: Dados de Teste: subconjunto de domínio de entrada

S: Resultados esperados: subconjunto correspondente do domínio de saída.

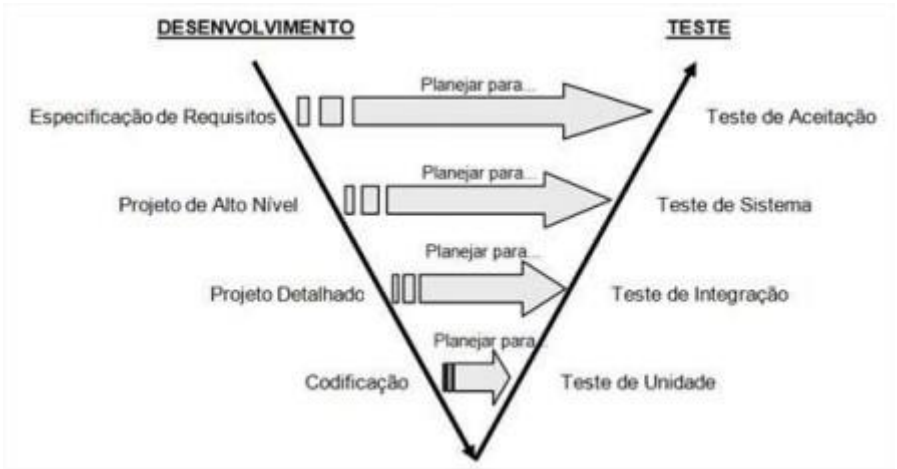
Oráculo: mecanismo que decide sobre a correção de uma execução.

Erro, Defeito e Falha

Erro: é o fruto da ação humana, que produz um resultado incorreto.

Defeito: também conhecido como bug, é o resultado de um erro no código, gerando uma anomalia no funcionamento no sistema.

Falha: é o resultado da execução de um defeito no código.



Níveis de Teste

➢ Teste de <u>Aceitação</u>

Verifica se o sistema está em conformidade com os requisitos esperados pelo cliente. Realizado pelo **cliente** ou pelo **testador**.

O sistema é utilizado para capacitação dos usuários de forma que eles validem todos os requisitos do sistema. Realizado de forma **manual** ou **automática.**

➢ Teste de <u>Sistema</u>

Verifica se o sistema está em conformidade com a especificação de requisitos. Realizado pelo **testador**, o qual tem acesso apenas a interface do sistema. Realizado de forma **manual** ou **automática**.

➢ Teste de <u>Integração</u>

Verifica se ao juntar vários componentes do sistema, se eles se comunicam corretamente. Realizado pelos **desenvolvedores** ou **analistas** de sistema para testar um módulo do sistema. Realizado de forma **automática**.

➢ Teste de <u>Unidade</u>

Teste realizado em uma unidade ou componente para verificar sua corretude. Realizado de forma **automática**.

Caixa-Branca	Caixa-Preta
Teste de Unidade	Testes Funcionais
Teste de Integração	Testes de Aceitação
	Testes Exploratórios

Técnicas do Teste

➢ Funcional

Identificar as funcionalidades que o software deve realizar. Projetar casos de teste capazes de verificar se essas funcionalidades estão sendo realizadas pelo software.

➢ Estrutural

Requisitos de testes com base no código fonte. Relacionada ao conhecimento da estrutura interna do programa.

Utilizam uma representação de programa conhecida como Grafo de Fluxo de Controle (GFC).

➢ Baseado em Defeitos

Visa a medir a adequação de um conjunto de casos de teste. Utiliza um conjunto de programas com erros introduzidos em sua estrutura, criando diversas versões denominadas de mutantes.

O objetivo é fazer que o programa mutante falhe, demonstrando a eficácia dos casos de teste.

Geração de Dados de Teste

É um processo de identificação de dados do domínio de entrada válidos para um determinado programa de acordo com os critérios de teste.

Manutenção de Software

O que é manutenção de software?

É o processo geral de mudança em um sistema depois que ele é liberado para uso.

Podem ser:

- Mudanças para correção de erros

- Correção de erros de projeto
- Acomodar novos requisitos

Existem 3 tipos de manutenção de software:

- Correção de defeitos
- Adaptação ambiental
- Adição de funcionalidades

Correção de defeitos

- Erros de codificação são relativamente baratos para serem corrigidos.

- Erros de projeto são mais caros, pois podem implicar reescrever vários componentes de programa.

- Erros de requisitos são os mais caros para se corrigir devido ao extenso reprojeto de sistema que pode ser necessário.

Adaptação Ambiental

- É necessário quando algum aspecto ambiente do sistema sofre mudança como:

- Hardware
- Sistema Operacional
- Outro software de apoio

O aplicativo deve ser modificado para se adaptar a essas mudanças.

Adição de Funcionalidades

É necessário quando os requisitos de sistema mudam em resposta às mudanças organizacionais ou de negócios.

A escala de mudança necessária para o software é muito maior do que para os outros tipos de manutenção.

Manutenção de software ocupa proporção maior dos orçamentos de TI que o desenvolvimento. Manutenção detém, aproximadamente, **dois terços** do orçamento, contra um terço para desenvolvimento.

- Adicionar uma nova funcionalidade após a liberação é caro porque é necessário tempo para aprender o sistema e analisar o impacto das alterações propostas.

- O trabalho feito durante o desenvolvimento para tornar a compreensão e a mudança no software mais fáceis, provavelmente resuzirá os custos de evolução.

- Geralmente é mais caro adicionar funcionalidade depois que um sistema está em operação do que implementar a mesma funcionalidade durante o desenvolvimento.

Razões para isso são:

- Estabilidade da equipe
- Más práticas de desenvolvimento
- Qualificações de pessoal
- Idade do programa e estrutura.

Estabilidade da Equipe

A nova equipe ou as pessoas responsáveis pela manutenção do sistema não entendem o sistema ou não entendem a estrutura para tomar as decisões de projeto.

Antes de implementar alterações é preciso investir tempo em compreender o sistema existente

Más práticas de Desenvolvimento

O contrato para a manutenção de um sistema é geralmente separado do contrato de desenvolvimento do sistema.

O contrato de manutenção pode ser dado a uma empresa diferente da do desenvolvedor do sistema original. Significa que não há incentivo para a equipe de desenvolvimento escrever um software manutenível.

Qualificações do Pessoal

A equipe de manutenção é relativamente inexperiente e não familiarizada com o domínio de aplicação.

A manutenção tem uma imagem pobre entre os engenheiros de software.

É vista como um processo menos qualificado do que o desenvolvimento, e é muitas vezes atribuída ao pessoal mais jovem.

A equipe pode não ter muita experiência de desenvolvimento nessas linguagens e precisa primeiro aprender para depois manter o sistema.

Idade do programa e estrutura

Com as alterações feitas no programa, sua estrutura tende a degradar, consequentemente, como os programas envelhecem, tornam-se mais difíceis de serem entendidos e alterados.

Podem nunca ter sido bem-estruturados e talvez tenham sido otimizados para serem mais eficientes do que inteligíveis.

As documentações podem ter-se perdido ou ser inconsistentes.

Os sistemas mais antigos podem não ter sido submetidos a um gerenciamento rigoroso de configuração, então desperdiça muito tempo para encontrar as versões certas dos componentes do sistema para a mudança.

Os primeiros três problemas decorrem do fato de que muitas organizações ainda consideram o desenvolvimento e a manutenção como atividades separadas.

Manutenção é vista como uma atividade de segunda classe, e não há incentivo para gastar dinheiro, durante o desenvolvimento, para reduzir os custos na alteração do sistema.

Previsão de manutenção

Deve-se tentar prever quais mudanças no sistema podem ser propostas e que partes do sistema são, provavelmente, as mais difíceis de serem mantidas.

É necessário também estimar os custos globais de manutenção para um sistema em determinado período de tempo.

Prever o número de solicitações de mudança para um sistema requer uma compreensão do relacionamento entre sistema e seu ambiente externo.

Para avaliar os relacionamentos entre um sistema e seu ambiente, deve-se avaliar:

- O número e a complexidade das interfaces de sistema.

Quanto maior o número de interfaces e mais complexas elas forem, maior a probabilidade de serem exigidas as alterações de interface quando novos requisitos forem propostos.

- O número de requisitos inerentemente voláteis de sistema.

Os requisitos que refletem as políticas e procedimentos organizacionais são provavelmente mais voláteis do que requisitos baseados em características estáveis de domínio.

- Os processos de negócio em que o sistema é usado.

Como processos de negócios evoluem, eles geram solicitações de mudança de sistema. Quanto mais processos de negócios usarem um sistema, maior a demanda por mudanças.