VOLTAR

As melhores práticas de engenharia de software: desenvolver de forma iterativa e gerenciar requisitos

Apresentar os conceitos relativos ao desenvolvimento iterativo e gerenciamento de requisitos.

NESTE TÓPICO

Introdução

Normalmente, há várias atividades envolvidas no desenvolvimento da solução de um determinado problema. Precisamos, por exemplo, compreender o problema, reunir os requisitos para a solução, traduzir esses requisitos em um projeto, construir a solução e testá-la.

Essa ordem é natural e geralmente correta. Problemas surgem quando tentamos fazer isso em uma sequência linear, isto é, quando tentamos reunir todos os requisitos, para só então fazer todo o projeto, e depois, na sequência, todo o desenvolvimento e no final executar todos os testes.

Infelizmente, a prática no desenvolvimento de software demonstra que dificilmente essa abordagem linear e sequencial, de executar cada fase de uma vez, funciona. Em certo sentido, muitas coisas em um projeto de desenvolvimento de software são teorias, ou mais precisamente, afirmações que precisam ser avaliadas.

Os requisitos podem sofrer alterações ao longo do desenvolvimento do software, e uma abordagem linear e sequencial quase sempre não conseguirá acomodar essas mudanças.

Isso nos leva a adotar um estilo de desenvolvimento de software em que as afirmações inerentes ao plano são repetidamente avaliadas pela concepção e desenvolvimento de versões demonstráveis do sistema. Cada uma dessas versões diminui gradativamente os riscos do projeto e são construídas em cima das anteriores para formar a solução final. Esse estilo de desenvolvimento é conhecido como desenvolvimento iterativo e incremental.

O que é desenvolvimento iterativo e incremental?

É um estilo de desenvolvimento que envolve a aplicação repetível (cíclica) de um conjunto de atividades destinadas a avaliar um conjunto de afirmações (requisitos), resolver um conjunto de riscos, realizar um conjunto de objetivos de desenvolvimento, e, de forma incremental, produzir e refinar uma solução eficaz.

Essa abordagem é iterativa porque envolve o refinamento sucessivo da compreensão do problema, a definição da solução e a implementação da solução pela aplicação repetitiva das atividades centrais de desenvolvimento. Ela é incremental porque em cada passagem pelo ciclo iterativo, cresce a compreensão do problema e a capacidade (e funcionalidades) implementadas na solução.

Várias repetições do ciclo iterativo são sequencialmente arranjadas para compor um projeto. Você deve tomar cuidado, porque o desenvolvimento pode ser iterativo sem ser incremental.

Por exemplo, as atividades de desenvolvimento podem ser aplicadas repetidamente (ou seja, de um modo iterativo), mas sem que faça crescer a compreensão do problema ou sem que faça que a solução acrescente novas funcionalidades, deixando o projeto no mesmo ponto que estava antes da iteração iniciar.

Pode acontecer também do desenvolvimento ser incremental sem ser verdadeiramente iterativo.

Por exemplo, o desenvolvimento de uma grande solução pode ser quebrado em um número de incrementos sem a aplicação repetitiva das atividades básicas de desenvolvimento.

Para ser verdadeiramente efetivo, o desenvolvimento deve ser iterativo e incremental.

Por que desenvolver iterativamente?

Sabemos que todo projeto têm riscos. Para diminuí-los, você deve desenvolver gradativamente de modo iterativo. Cada iteração resultará em um release executável (incremento).

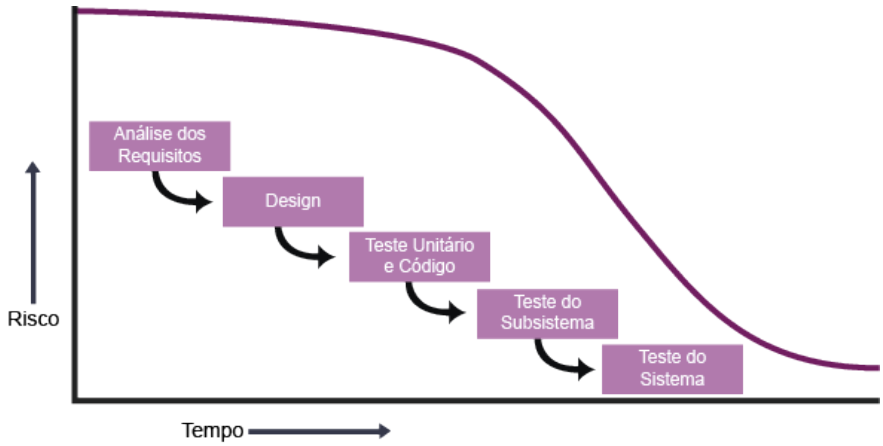
A experiência mostra que um design inicial provavelmente vai conter falhas em relação a seus principais requisitos.

A descoberta tardia dos defeitos de design resulta em correções caras e, em alguns casos, até mesmo no cancelamento do projeto.

Conforme citado anteriormente, todos os projetos têm um conjunto de riscos envolvidos. Quanto mais cedo você puder verificar que evitou um risco no ciclo de vida, mais precisos serão seus planos.

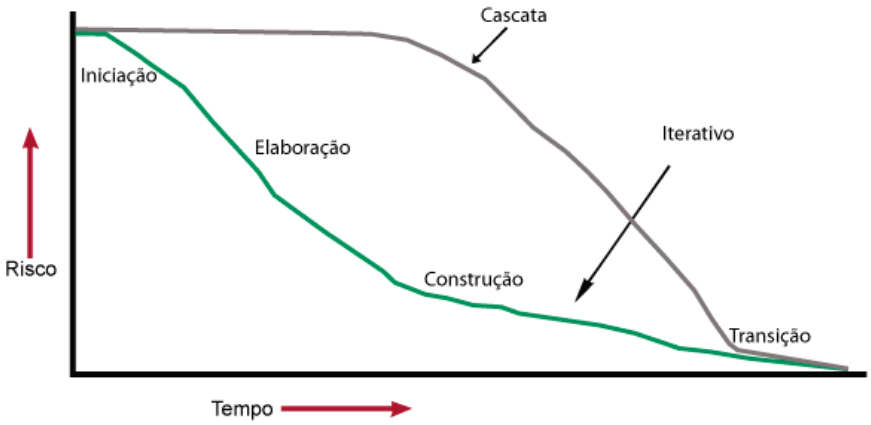
Muitos riscos não são descobertos até que você tente integrar o sistema. É impossível prever todos eles, por mais experiente que seja a equipe de desenvolvimento.

Em um ciclo de vida em cascata você não poderá verificar se ficou livre de um risco até o final do ciclo, conforme apresentado na figura 1 a seguir.



Risco no modelo cascata Fonte: [PRESSMAN, 2010]

Em um ciclo de vida iterativo, a seleção do incremento a ser desenvolvido em uma iteração é feita com base em uma lista dos principais riscos. Como a iteração produz um executável testado (incremento), você poderá verificar se os riscos diminuíram, conforme apresentado na figura 2.



Vantagens de uma abordagem iterativa

Geralmente, uma abordagem iterativa é superior a uma abordagem linear ou em cascata, por vários motivos:

- Os riscos são reduzidos mais cedo, pois os elementos são integrados progressivamente.

- As táticas e os requisitos variáveis são acomodados.
- A melhoria e o refinamento do produto são facilitados, resultando em um produto mais robusto.
- As organizações podem aprender a partir dessa abordagem e melhorar seus processos.
- A capacidade de reutilização aumenta.

O que é gerenciamento de requisitos?

Gerenciamento de requisitos é uma abordagem sistemática para localizar, documentar, organizar e controlar os requisitos variáveis em um sistema.

Definimos um requisito como "uma condição ou uma capacidade com a qual o sistema deverá estar em conformidade".

O gerenciamento de requisitos é definido formalmente como uma abordagem sistemática para: identificar, organizar e documentar os requisitos do sistema, além de firmar e atualizar acordos entre o cliente e a equipe do projeto sobre os requisitos variáveis do sistema.

As chaves para o gerenciamento eficaz de requisitos incluem manter uma sentença clara dos requisitos, junto dos atributos aplicáveis e a rastreabilidade para outros requisitos e outros artefatos do projeto.

As seguintes habilidades são importantes para o gerenciamento de requisitos:

I. Análise do problema.
II. Noções básicas sobre as necessidades dos envolvidos.
III. Definição do sistema.
IV. Gerenciamento do escopo do projeto.
V. Refinamento da definição do sistema.
VI. Gerenciamento dos requisitos variáveis.
VII. Análise.

Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

EXERCÍCIO (https://ead.uninove.br/ead/disciplinas/impressos/_g/pdsoft80_100/a09ex

Referências

PRESSMAN, R. S. Engenharia de software. 7. ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, Ian. Engenharia de software. São Paulo: Addison-Wesley, 2007.

Avalie este tópico

ANTERIOR
Engenharia orientada a servicos

Índice

PRÓXIMO
As melhores práticas de engenharia de soft
ware: modelar visualmente (UML) e usar a
tetura baseada em componentes