GOVERNO DO ESTADO DO CEARÁ
Secretaria da Ciência, Tecnologia e Educação Superior

UNIVERSIDADE ESTADUAL VALE DO ACARAÚ

Engenharia de Software

1º Bloco

NEaD UVA
Núcleo de Educação a Distância
Universidade Estadual Vale do Acaraú
SOBRAL - 2013

CURSO DA CIÊNCIA DA COMPUTAÇÃO
ENGENHARIA DO SOFTWARE

Capítulo 1

INTRODUÇÃO

O que é?

Software de computador é o produto que os profissionais de software constroem e, depois, mantêm ao longo do tempo. Abrange programas que executam em computadores de qualquer tamanho e arquitetura, conteúdo que é apresentado ao programa a ser executado e documentos tanto em forma impressa quanto virtual que combinam todas as formas de mídia eletrônica.

Sistemas Operacionais são um exemplo de softwares

Quem faz?

Engenheiros de software, cientistas da computação e profissionais da área constroem, mantêm, e praticamente todas as pessoas do mudo industrializado usam direta ou indiretamente.

Por que é importante?

Por que afeta praticamente todos os aspectos de nossas vidas e tornou-se difundido o nosso comércio, na nossa cultura e nas nossas atividades do dia-a-dia.

Quais são os passos?

Você constrói o software de computadores como constrói qualquer produto bem-sucedido, aplicando um processo ágil e adaptável que leva a um resultado de alta qualidade e que satisfaz ás necessidades das pessoas que vão usar o produto. Você aplica uma abordagem de engenharia de software.

Qual é o produto do trabalho?

Do ponto de vista do engenheiro de software, o produto do trabalho são os programas, conteúdo (dados) e documentos que compõem um software de computador. Mas, pelo ponto de vista do usuário, o produto do trabalho é a informação resultante que, de algum modo, torna melhor o mundo do usuário.

O PAPEL EVOLUTIVO DO SOFTWARE

Software como produto

Nos dias atuais o software assume um duplo papel. Ele é produto e, ao mesmo tempo, o veículo de entrega do produto. Como produto ele disponibiliza o potencial de computação presente no hardware do computador ou, mais amplamente, por uma rede de computadores acessível pelo hardware local. Quer resida em um telefone celular, quer opere em um computador de grande porte, o software é um transformador de informações. Produzindo, gerindo, adquirindo, modificando, exibindo ou transmitindo informações que pode ser tão simples como um bit ou tão complexas quanto uma apresentação multimídia. Como veículo usando para entrega do produto, o software age como base para o controle do computador (Sistema Operacional), para a comunicação das informações (rede) e para criação e o controle de outros programas (ferramentas e ambientes de software).


Software servindo de veículo para entrega de produto

A importância do software de computadores tem passado por mudanças significativas em pouco mais de 50 anos. Uma melhora surpreendente no desempenho do hardware, profundas modificações na arquitetura de computadores, aumento significativo na memória e na capacidade de armazenamento, e uma variedade de opções incomuns de entrada e saída levaram a sistemas baseados em computador mais sofisticados e completos. Sofisticação e complexidade podem produzir resultados magníficos quando um sistema é bem-sucedido, mais também podem causar enormes problemas para quem precisa construir sistemas complexos.

Hoje em dia uma enorme indústria de software tornou-se fator dominante nas economias do mundo industrializado. O programador solitário de antigamente foi substituído por uma equipe de especialistas de software, cada um se concentrando numa parte da tecnologia necessária para produzir uma aplicação complexa. No entanto, as questões que eram feitas ao programador solitário são as mesmas questões que são feitas quando modernos sistemas baseados em computador são construídos.

- Porque leva tanto tempo para concluir o software?
- Por que os custos de desenvolvimento são tão altos?
- Por que não podemos achar todos os erros antes de entregar o software aos clientes?
- Por que gastamos tanto tempo e esforço mantendo programas existentes?
- Por que continuamos a ter dificuldade em avaliar o progresso enquanto o software é desenvolvido e mantido ?

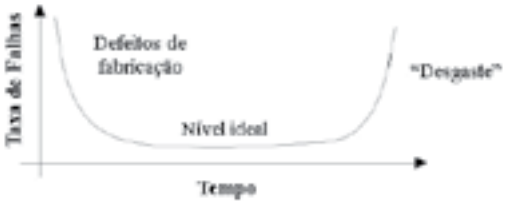
Essas e muitas outras questões demonstram a preocupação da indústria com o software e a maneira pela qual ele é desenvolvido, preocupação que tem levado à adoção da prática de engenharia de software.

SOFTWARE

Para entendermos de software, é importante examinar as características do software que o tornam diferente de outras coisas que os seres humanos produzem. O software é um elemento de um sistema lógico e não de um sistema físico. Assim, ele possui características que são consideravelmente diferentes daquelas do hardware.

O software é desenvolvido ou passa por um processo de engenharia, não é fabricado no sentido clássico.

Apesar de existirem algumas semelhanças entre o desenvolvimento de software e a fabricação de hardware, as duas atividades são fundamentalmente diferentes. Em ambas, a alta qualidade é conseguida por um bom projeto, mas a fase de fabricação do hardware pode gerar problemas de qualidade, que são inexistentes (ou facilmente corrigidos) para o software. Ambas as atividades são dependentes de pessoas, mas a relação entre as pessoas envolvidas e o trabalho realizado é inteiramente diferente. As duas atividades requerem a construção de um "produto", porém as abordagens são diferentes. Os custos do software são concentrados na engenharia. Isso significa que os projetos de software não podem ser geridos como se fosse projetos de fabricação.

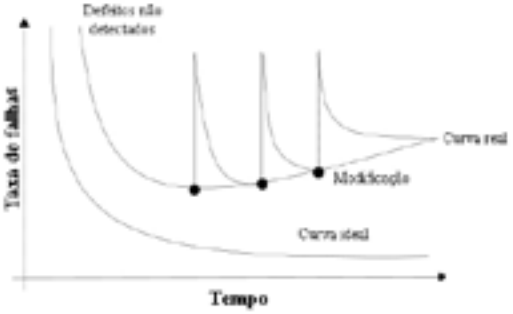


Software não "se desgasta".

Usando uma análise de o gráfico a seguir relacionado a curva de falhas para o hardware no qual exibe taxas de falhas relativamente altas no início de sua vida (essas falhas são frequentemente atribuíveis a defeitos de projeto de fabricação).

Figura 1.1: Curva de falhas para o hardware

Os defeitos de fabricação são depois corrigidos e a taxa de falhas cai para um nível constante (idealmente bastante baixo) por certo período de tempo. Com o passar do tempo, no entanto, a taxa de falhas aumenta novamente, visto que os componentes de hardware sofrem o efeito cumulativo da poeira, vibração, maus tratos, gradientes de temperatura e muitos outros males ambientais, Falando simplesmente, o hardware começa a se "desgastar".



O software não é suscetíveis males ambientais que causam desgaste do hardware. Teoricamente, entretanto, a curva da taxa de falhas para o software deveria tomar a forma de "curva ideal" mostrada na figura. Defeitos não detectados causarão altas taxas de falhas no inicio da vida de um programa. Todavia, eles são corrigidos (idealmente sem a introdução de outros erros) e a curva se achata, conforme mostrado. A curva ideal é uma simplificação grosseira dos modelos reais de falhas para softwares. Todavia, a implicação é clara, o software não se desgasta mais se deteriora.

Hardware desgastado por sujeira

Falhas de software deteriorado

Essa aparente contradição pode ser mais bem explicada considerando a "curva real". Durante a sua vida, o software passará por modificações. Conforme as modificações são feitas, é provável que err0s sejam introduzidos causando um dente na curva da taxa de falhas, conforme mostra na figura. Antes que a curva possa voltar ao seu estado original de taxa de falhas estável, outras modificações são solicitadas causando novo dente na curva. A taxa de falhas mínima começa a subir lentamente, o software está se deteriorando em razão das modificações.

Outro aspecto do desgaste ilustra a diferença entre hardware e software. Quando um componente de hardware se desgasta, é substituído por uma peça sobressalente. Não há pelas sobressalentes de software. Toda falha de software indicam erro no projeto, ou no processo pelo qual o projeto foi traduzido para o código de máquina executável. Assim, a manutenção de software envolve consideravelmente mais complexidade do que a manutenção de hardware.

Apesar de a indústria estar se movendo em direção à montagem baseada em componentes, a maior parte dos softwares continua a ser construída sob encomenda.

Considere a maneira pela qual o hardware de controle para um produto é projetado e construído. O engenheiro de projeto desenha um esquema simples do circuito digital, faz uma análise fundamental para assegurar a função pretendida e, depois, consulta os catálogos de componentes digitais. Cada circuito integrado tem um código de componente, uma função definida e valida, uma interface bem delineada e um conjunto padrão de diretrizes de integração. Depois que cada componente é selecionado ele pode ser requisitado em estoque.

A NATUREZA MUTÁVEL DO SOFTWARE

Atualmente, sete amplas categorias de software de computadores apresentam desafios contínuos para os engenheiros de software, são elas:

Software de sistemas.

É uma coleção de programas escritos para servir outros programas, por exemplo, compiladores, editores e utilitários para gestão de arquivos, processam estruturas de informação complexas mais determinadas. Outras aplicações de sistema como exemplo, componentes de sistemas operacionais, acionadores e software de rede processam dados amplamente indeterminados. Em ambos os casos, a área de software de sistemas é caracterizada por interação intensa com o hardware de computador, uso intenso por múltiplos usuários, operação concorrente que requer ordenação, compartilhamento de recursos e sofisticada gestão de processo, estruturas de dados complexas e interfaces externas múltiplas.

Software de aplicação

Consiste em programas isolados que resolvem uma necessidade específica do negócio. Aplicações nessa área processam dados comerciais ou técnicos de um modo que facilita as operações ou gestão/tomada de decisões técnicas do negócio. Além das aplicações convencionais de processamento de dados, o software de aplicação é usado para controlar funções do negócio em tempo real como processamento de transações no ponto-de-venda e controle de processo de fabricação em tempo real.

Software científico e de engenharia

Caracterizado antigamente por "number crunching" (que processam números), suas aplicações vão desde a astronomia à vulcanologia, da análise automotiva de tensões â dinâmica orbital do ônibus espacial, e da biologia molecular à manufatura automatizada. Todavia, as aplicações modernas nesta área estão se afastando dos algoritmos numéricos convencionais. Projeto apoiado por computadores, simulação de sistemas e outras aplicações interativas começaram a adquirir características de tempo real e até de software de sistemas.

Software embutido

Reside dentro de um produto ou sistema e é usado para programar e controlar características e funções para o usuário final e para o próprio sistema. O software embutido pode realizar funções muito limitadas e particulares (como exemplo o controle de teclado para um forno de micro-ondas) ou fornecer função significativa e capacidade de controle (como exemplo funções digitais em um automóvel tais como controle de combustível, mostradores do painel e sistemas de frenagem).

Ex: Os softwares mobile que rodam "dentro" de outro software (Sistema Operacional mobile).

Software para linhas de produtos

Projetado para fornecer uma capacidade especifica usada por muitos clientes diferentes, o software para as linhas de produtos pode focalizar um mercado limitado e especial (como exemplo, produtos de controle de estoque) ou dirigir-se ao mercado de consumo de massa (por exemplo, processamento de texto, planilhas, gráficos por computador, multimídia, entretenimento, gestão de banco de dados, aplicações financeiras pessoas e empresariais).

Aplicações da Web

Conhecidas como ApsWeb, cobrem uma ampla gama de aplicações. Na sua forma mais simples, ApsWeb podem ser pouco mais que um conjunto de arquivos ligados por hipertexto que apresentam informações usando texto e poucos gráficos. No entanto, conforma as aplicações

de comércio eletrônico (e-commerce) e B2B (Bussines to Bussines) crescem em importância, as ApsWeb evoluem para sofisticados ambientes computacionais que fornecem não apenas características isoladas, funções de computação e conteúdo para o usuário final, mais também estão integradas ao banco de dados da empresa e às aplicações de negócio.

Software para inteligência artificial.

Faz uso de algoritmos não-numéricos para resolver problemas complexos que não são passíveis de computação ou análise direta. Aplicações nessa área incluem robótica, sistemas especialistas, reconhecimento de padrões (de imagem e voz), redes neurais, prova de teoremas e jogos.

Ex: Siri, software "inteligente" de reconhecimento de voz usado nos produtos Apple.

SOFTWARE LEGADO

Centenas de milhares programas de computador caem em um dos sete amplos domínios de aplicação já citados anteriormente. Alguns deles são software do estado da arte, recém lançados para indivíduos, industria e governo. Mas outros programas são mais antigos, em alguns casos muito mais antigos.

Esses programas antigos frequentemente referidos como software legado onde têm sido foco de atenção e preocupação contínuas desde a década de 1960. Muitos sistemas legados permanecem dando suporte a funções importantes do negocio e são indispensáveis ao negócio. Assim o software legado é caracterizado por longevidade e criticalidade para o negócio.

A Qualidade do Software Legado

Infelizmente existe uma característica adicional que pode estar presente no software legado, algumas vezes, têm projetos não extensíveis, código complicado, documentação pobre ou inexistente, casos de teste e resultados que nunca fora arquivados, um histórico de informações mal gerido, a lista pode ser bem longa. E, no entanto, esses sistemas apoiam "funções importantes do negócio e são indispensáveis para o mesmo". O que pode ser feito?

A única resposta razoável pode ser não fazer nada, pelo menos ate que o sistema legado precise passar por modificações significativas. Se o software legado satisfaz às necessidades de seus usuários e funciona confiavelmente, não este danificado e não precisa ser consertado. No entanto, à medida que o tempo passa os sistemas legados frequentemente evoluem, por uma ou mais das seguintes razões:

- Adaptado para satisfazer ás necessidades do novo ambiente ou tecnologia computacional.
- Estendido para torná-lo interoperável com os sistemas ou bancos de dados mais modernos.
- Aperfeiçoado para implementar novos requisitos do negócio.
- Rearquitetado para torná-lo viável em um ambiente de rede.

Quando esses modos de evolução ocorrem, um sistema legado precisa ser "reengenhado" de modo que permaneça viável no futuro. O objetivo da engenharia de software moderna é "conhecer metodologias que sejam fundamentadas na noção de evolução", ou seja, a noção de que "sistemas de software modificam-se continuamente, novos sistemas de software são construídos a partir dos antigos e... todos precisam interoperar e cooperar uns com os outros".

EVOLUÇÃO DO SOFTWARE

Independentemente do tamanho, da complexidade ou domínio de aplicação, o software de computador vai evoluir com o tempo. As modificações (frequentemente denominadas "manutenção de software") orientam esse processo e ocorrem quando os erros são corrigidos, quando o software é adaptado a um novo ambiente, quando o cliente solicita novas características ou funções quando a aplicação é "reengenhada" para fornecer benefícios em um contexto moderno.

Sistemas tipo-E

São softwares que foram implementados em um sistema de computação do mundo real e, portanto, evoluirão com o tempo.

- Lei da Modificação Contínua (1974)

Sistemas tipo-E devem ser continuamente adaptados ou então eles se tornam progressivamente menos satisfatórios.

-Lei da Complexidade Crescente (1974)

À medida que um sistema tipo-E evolui sua complexidade aumenta, a menos que seja realizado esforço para mantê-la ou reduzi-la.

- Lei da Auto-Regulação (1974)

O processo de evolução de sistemas tipo-E é auto-regulatório, com medidas de distribuição do produto e do processo perto do normal.

- Lei da Conservação da Estabilidade Organizacional (1980)

A velocidade média da atividade global efetiva em um sistema evolutivo tipo-E é invariante ao longo do tempo de vida do produto.

-Lei da Conservação da Familiaridade (1980)

À medida que um sistema tipo-E evolui, todos os que estão a ele associados, desenvolvedores, pessoal de vendas e usuários, por exemplo, devem manter o domínio de seu conteúdo e comportamento para conseguir evolução satisfatória. O crescimento excessivo diminui esse domínio. Assim, o crescimento médio incremental permanece invariante à medida que o sistema evolui.

Ex: Softwares da "família" Autodesk, como o Maya, AutoCAD e etc.

- Lei do Crescimento Contínuo (1980)

O conteúdo funcional de sistemas tipo-E deve ser continuamente aumentado para manter a satisfação do usuário ao longo do tempo de vida do sistema.

- Lei da Qualidade Declinante (1996)

A qualidade de sistemas tipo-E perecerá estar declinando a menos que eles sejam rigorosamente mantidos e adaptados às modificações no ambiente operacional.

- Lei de Realimentação do Sistema (1996)

Os processos de evolução do tipo-E constituem sistemas de realimentação em multiníveis, em multi-ciclos, por multiagentes e precisam ser tratados como tal para conseguir aperfeiçoamento significativo sobre qualquer base razoável.

MITOS DO SOFTWARE

Crenças sobre softwares e sobre o processo usado para construí-los, podem ser encontrados desde os primeiros dias da computação. Os mitos têm uma quantidade de atributos que os têm tornado traiçoeiros. Por exemplo, os mitos parecem ser afirmações de fatos razoáveis (algumas vezes contendo elementos verdadeiros), têm um aspecto intuitivo e são frequentemente divulgados por profissionais experientes que "entendem do assunto".

Mitos da gerência

Os gerentes com responsabilidade sobre o software, como os gerentes na maioria das atividades, estão frequentemente sob pressão para obedecer a orçamentos, manter cronogramas no prazo para melhorar a qualidade. Como uma pessoa que está se afogando e se agarra em um pedaço de madeira, um gerente de software frequentemente se agarra a crença em um mito de software, se isso puder diminuir a pressão (ainda que temporariamente).
Mito: Já temos um livro e está cheio de padrões e procedimentos para elaborar o software. Isso não fornece ao meu pessoal tudo o que ele precisa saber?
Realidade: O livro de padrões pode muito bem existir, mais será que é usado? Os profissionais de software sabem da sua existência? Ele reflete as práticas modernas da engenharia de software? É completo? É adaptável? Está voltando para melhorar o prazo de entrega, mantendo o foco na qualidade? Em muitos casos, a resposta a todas essas perguntas é não.
Mito: Se nos atrasarmos no cronograma, podemos adicionar mais programadores e ficar em dia (algumas vezes denominado conceito da Horda mongólica).
Realidade: O desenvolvimento de software não é um processo mecanizado como o de fabricação. Nas palavras de Brooks (1975): "Adicionar pessoas a um projeto de software atrasado atrasa-o ainda mais". A princípio, essa afirmação pode parecer contra-intuitiva. Todavia, sempre que novas pessoas são adicionadas, o pessoal que estava trabalhando precisa gastar tempo orientando os recém-chegados, reduzindo assim a quantidade de tempo empregado no esforço de desenvolvimento do produtivo. Pessoas podem ser adicionadas, mais apenas de forma planejada e bem coordenada.
Mito: Se eu decidir terceirizar um projeto de software vou poder relaxar e deixar que aquela firma o elabore.
Realidade: Se uma organização não sabe como gerir e controlar projetos de software internamente, certamente terá problemas quando terceirizar esses projetos.

Mitos do cliente

Um cliente que encomenda software para computador pode ser uma pessoa na mesa vizinha, um grupo técnico em outra sala, o departamento de promoções e vendas, ou outra empresa que encomendou software sob contrato. Em muitos casos, o cliente acredita em mitos de software, porque os gerentes e profissionais de software fazem pouco para corrigir essa desinformação. Mitos levam a falsas expectativas (pelo cliente) e, em última análise, à insatisfação com o desenvolvedor.
Mito: O estabelecimento geral de objetivos é suficiente para iniciar a escrita de programa, podemos fornecer os detalhes posteriormente.
Realidade: Embora uma declaração abrangente é estável dos requisitos nem sempre seja possível, uma definição inicial malfeita é a principal causa de esforços malsucedidos de software. Uma descrição formal e detalhada do domínio da informação, da função, do comportamento, do

desempenho, das interfaces, das restrições de projeto e dos critérios de validação é essencial. Essas características podem ser determinadas somente pois de intensa comunicação entre o cliente e o desenvolvedor.

Mito: Os requisitos de projeto mudam continuamente, mas as mudanças podem ser facilmente acomodadas porque o software é flexível.

Realidade: É verdade que os requisitos de software mudam, mas o impacto da mudança varia com a época em que é introduzida. Quando mudanças são solicitadas antecipadamente (antes que o projeto e a codificação tenham começado), o impacto no custo é relativamente baixo. No entanto, à medida que o tempo passa, o impacto de custo cresce rapidamente, recursos foram comprometidos, a estrutura do projeto foi estabelecida e a mudança pode causar consequências que exijam recursos adicionais e grandes modificações no projeto.

- Mitos do profissional

Os mitos que ainda têm crédito entre os profissionais de software sobreviverem a mais de 50 anos de cultura de programação. Durante os primeiros dias do software, a programação era vista como uma forma de arte. Maneiras e atitudes antigas custam a morrer.

Mito: Quando escrevemos um programa e o fazemos funcionar, nosso trabalho está completo.

Realidade: Alguém disse um dia que "quando mais cedo você começar a escrever código mais vai demorar para acabar". Dados da indústria indicam que entre 60% e 80% de todo esforço despedindo em software vai ser despendido depois de ele ser entregue ao cliente pela primeira vez.

Mito: Até que eu esteja com o programa "rodando" não tenho como avaliar a sua qualidade.

Realidade: Um dos mecanismos mais eficazes de garantia de qualidade de software pode ser aplicado a partir do início de um projeto, a revisão formal. Revisões de software são um "filtro de qualidade" que se descobriu ser mais eficaz do que testes para encontrar alguns tipos de erros de software.

Mito: O único produto do trabalho que pode ser entregue para um projeto de software bem sucedido é o programa executável.

Realidade: Um programa executável é apenas uma parte de uma configuração de software que inclui vários elementos. A documentação fornece a base para uma engenharia bem-sucedida e, mais importante, orientada para suporte ao software.

Mito: A engenharia de software vai nos fazer criar documentação volumosa e desnecessária que certamente nos atrasará.

Realidade: A engenharia de software não se relaciona à criação de documentos. Refere-se à criação de qualidade. Melhor qualidade leva à redução de retrabalho. E menor retrabalho resulta em tempos de entrega mais rápidos.

Muitos dos profissionais de software reconhecem a falácia dos mitos descritos. Lamentavelmente, ações e métodos habituais levam a práticas de gestão e técnicas de baixa qualidade mesmo quando a realidade exige melhor abordagem. O reconhecimento das realidades do software é o primeiro passo em direção à formulação de soluções práticas para engenharia de software.

QUESTIONÁRIO

1) Descreva o que você entende por Software com suas próprias palavras.
2) Qual o produto do trabalho de um engenheiro de software, no ponto de vista do usuário e do engenheiro ?
3) Cite alguns exemplos (tanto positivos quanto negativos) que indicam o impacto do software na nossa sociedade.
4) Com a evolução dos softwares proporcionou a criação de programas mais sofisticados e complexos no qual poderão ocorrer problemas na construção do mesmo, que problemas seriam esses ?
5) Descreva com suas próprias palavras sobre as vantagens de passar da era "programador solitário" para uma equipe de especialistas com cada um se concentrando numa parte da aplicação.
6) Muitas perguntas são uma clara manifestação de preocupação sobre a maneira que o software é desenvolvido, são elas:" Por que leva tanto tempo para concluir um software ?" "Por que os custos de desenvolvimento são tão altos ?" Responda-as com suas próprias palavras.
7) Comente o porque o software se deteriora e "não se desgasta".
8) Muitas aplicações modernas modificam-se frequentemente. Sugira alguns modos para se construir software que não se deteriore com as modificações.
9) Em relação a curva de falhas para o software, porque no início da vida do programa é detectado uma alta taxa de falhas ?
10) Quais as vantagens e os motivos que os softwares ainda continuam a serem construídos sob encomenda ?
11) Quais as sete amplas categorias de software de computadores que apresentam desafios contínuos para os engenheiros de software ?
12) Compiladores, editores e utilitários para gestão de arquivos, são exemplos de qual aplicação de que categoria de software de computadores ?
13) "Faz uso de algoritmos não-numéricos para resolver problemas complexos que não são passíveis de computação ou análise direta", estamos falando de qual categoria de software de computador ? Cite 3 (três) exemplos de algumas aplicações nesta área.
14) O que você entendeu por Lei da Conservação da Familiaridade (1980) ? Dê alguns exemplos.
15) Considere as 7 categorias de software. Pode a mesma abordagem de engenharia de software ser aplicada a cada uma delas? Justifique sua resposta.
16) Cite alguns mitos da gerência e comente-os.
17) Comente sobre o Mito de Gerência "Se nos atrasarmos no cronograma, podemos adicionar mais programadores e ficar em dia com a criação do software?", será ideal adicionar mais programadores ou manter a equipe?

REFERÊNCIAS

1. PRESSMAN, Roger S. Engenharia de Software, 6.ed, p. 1-14, 2006.