

Engenharia de Software



Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico



Introdução à Engenharia de Software

Responsável pelo Conteúdo:

Prof.^a Dr.^a Ana Paula do Carmo Marchetti Ferraz

Revisão Textual:

Prof.^a Me. Luciene Oliveira da Costa Santos



- Introdução
- Engenharia de Software (ES) e produto de software
- A diversidade na ES, sua integração com a internet e a ética relacionada ao desenvolvimento de produto de software
- Reengenharia e Engenharia Reversa



- Apresentar os conceitos gerais sobre Engenharia de Software.
- Entender os fatores que influenciam e dificultam a construção do software.
- Conhecer os reais objetivos da Engenharia de Software.

Primeiramente, é necessária a conscientização de que a exigência – tanto de estudo, quanto de avaliação – da disciplina *online* é, no mínimo, a mesma que ocorre na disciplina presencial.

Assim, é necessário organizar seu tempo para ler atentamente o material teórico e assistir aos vídeos, se houver, inclusive do material complementar.

Organize-se também de forma a não deixar para o último dia a realização das atividades (AS e AP). Podem ocorrer imprevistos e, quando encerrada a Unidade, encerra-se a possibilidade de obter a nota relativa a cada atividade.

Para ampliar seu conhecimento, bem como aprofundar os assuntos discutidos, pesquise, leia e consulte os livros indicados nas Referências e/ou na Bibliografia.

As Referências estão indicadas ao final dos textos de conteúdo de cada Unidade.

A Bibliografia Fundamental para esta Disciplina é: SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011. A Bibliografia Complementar está indicada em item específico, em cada unidade.

Caso ocorram dúvidas, contate o professor tutor por meio do Fórum de Dúvidas, local ideal, pois assim a explicação poderá ser compartilhada por todos.

Existe, ainda, a possibilidade de contatar o professor pelo link Mensagens, caso seja algum assunto relativo somente a você, e não uma dúvida sobre a matéria, que pode ser também dúvida de outros alunos.

Fique atento(a) quanto a possíveis dúvidas e problemas, lembrando-se de que o professor tutor está à sua disposição para resolver assuntos pedagógicos, isto é, dúvidas quanto ao conteúdo da matéria.

Se suas dúvidas ou problemas se referirem ao sistema, aos programas etc., contate o Suporte, ou procure auxílio de um monitor no webclass de seu campus.

Contextualização

O mundo atual não seria o que é sem que tivéssemos software, estruturas de redes e toda tecnologia que integra esses sistemas. Serviços nacionais e internacionais são controlados por software. Produtos são desenvolvidos num determinado país, fabricados em outro e comercializados num terceiro. Todo esse processo está informatizado, da produção de matéria-prima ao controle de satisfação de um produto finalizado, sem contar a indústria da música, jogos, cinema e televisão.

Nesta unidade da disciplina **Engenharia de Software (ES)**, apresentaremos alguns conceitos gerais sobre ES e como, aos poucos, ela foi dando forma a uma atividade (desenvolvimento de software) que era considerada caótica, sem muito controle e passível de não cumprimento de prazo em grandes proporções.

Aqui, você entrará em contato com os assuntos principais deste conteúdo de forma breve e geral e terá a oportunidade de aprofundar essas questões no estudo de cada unidade. No entanto, esta unidade inicial visa fornecer-lhe o conhecimento básico necessário para que você possa construir um referencial teórico com base sólida – científica e cultural – e o exerça, no futuro exercício de sua profissão, com competência cognitiva, ética e responsabilidade social.

Vamos começar nossa aventura pela apresentação das ideias e dos princípios básicos que fundamentam esta disciplina.

Como forma de despertá-lo(a) para a matéria e de direcionamento, vamos retornar à seguinte pergunta:

Somente o domínio de uma linguagem é suficiente para uma pessoa ou equipe produzir softwares de qualidade?

1. Introdução



Para iniciar o estudo desta unidade, é importante que você tenha em mente que desenvolver software é muito mais do que a atividade de codificação, ou seja, é muito mais do que utilizarmos adequadamente uma linguagem de programação.

Antes, desenvolver sistemas era apenas “programá-los”, mas à medida que a tecnologia evoluiu os sistemas baseados em computador ficaram mais complexos. O que antes era trabalho de um profissional ou uma equipe fisicamente próximo, começou a ser desenvolvido por inúmeras pessoas fisicamente distantes. E como garantir a qualidade do produto final? Como garantir prazos? Como mensurar prazo de entrega, tempo de desenvolvimento, teste e valor do produto?

Foi nesse contexto que a Engenharia de Software (ES) surgiu.

Antes de abordarmos os conceitos de ES vamos apresentar um conceito importante e bastante utilizado na área de informática, assim como nesta disciplina: o de **sistemas** e suas aplicações – sistemas computacionais.

É interessante, porém, que você entenda o conceito dessa palavra, que é muito empregada não só na informática, mas em muitas outras áreas. Certamente, você já deve ter utilizado tal expressão para se referir, por exemplo, ao sistema solar, ao sistema educacional, ao sistema respiratório etc.

No nosso contexto, **sistema computacional é um conjunto de programas – softwares – integrados**.

Segundo Pressman (2006), se quisermos definir **software**, numa forma clássica, podemos até dizer que ele faz parte de um **conjunto de instruções** que, quando executadas, produzem a função e o desempenho desejados, possui **estruturas de dados** que permitem que as informações relativas ao problema a resolver sejam manipuladas adequadamente e documentação que é necessária para um melhor entendimento da sua operação e uso.

Pressman (1995, p. 179) inclui como elementos de um sistema computacional:

1. **Software:** programas de computador, estruturas de dados e documentação correlata que servem para efetivar o método, processo ou controle lógico necessário.
2. **Hardware:** dispositivos eletrônicos que fornecem a capacidade ao computador e dispositivos eletromecânicos que oferecem funções ao mundo externo.
3. **Pessoas:** usuários e operadores de hardware e software.
4. **Banco de dados:** uma grande e organizada coleção de informações a que se tem acesso pelo software e é parte integrante da função do sistema.
5. **Documentação:** manuais, formulários e outras informações descritivas que retratam o uso do sistema.
6. **Procedimentos:** os passos que definem o uso específico de cada elemento do sistema ou o contexto processual em que o sistema reside.

Esses elementos interagem uns com os outros para transformar informações. Assim, para que um sistema computacional cumpra realmente seus objetivos, é importante não apenas se preocupar com a construção de seus elementos, mas também é fundamental que todos os elementos estejam integrados entre si de forma efetiva.

Entretanto, se pensarmos no contexto da Engenharia de Software, o software deve ser visto como um sistema que se transforma em **produto** a ser desenvolvido para ser vendido.

Antes de abordarmos esse conceito de produto, devemos entender que, segundo Sommerville (2011, p. 2):

[...] os sistemas de software são abstratos e intangíveis. Eles não são restringidos pelas propriedades dos materiais, nem governados pelas leis da física ou pelos processos de manufatura. Isso simplifica a ES, porque não há limites para o potencial de software. Existem vários tipos de sistemas de software, desde os mais simples sistemas embutidos até os sistemas de informações complexos de alcance mundial.

É nesse contexto que iniciaremos a nossa discussão sobre ES, ou seja, existem diferentes tipos de sistemas computacionais e não faz sentido procurarmos por métodos, notações, técnicas comuns e universais, porque diferentes softwares precisam de diferentes abordagens de desenvolvimento. Desenvolver e comercializar um software corporativo é completamente diferente de atuar no controle de instrumentos científicos, que é diferente também de atuar na telefonia e ainda é completamente distinto de desenvolver um *game*.

Todas essas aplicações precisam dos conceitos de ES no software no desenvolvimento de um produto que é denominado **produto de software**.

Os conceitos abordados nesta disciplina estão relacionados a produtos de software de acesso **não restrito**, ou seja, para clientes maiores, que possuem características de integração com outros programas/sistemas e bases de dados internas e externas.

Para softwares menores, restritos, com um único usuário/cliente, os conceitos de Engenharia de Software não se aplicam em toda sua extensão. No caso desses programas, documentação, testes, manutenção, técnicas de modelagem, portabilidade, flexibilidade etc., a associada é pequena ou, na maioria das vezes, inexistente, pois, segundo Sommerville (2011, p. 3), a ES “[...] tem por objetivo apoiar o desenvolvimento profissional de software, mais do que a programação individual”.

2. Engenharia de software e produto de software



Engenharia de Software é uma disciplina de Engenharia cujo foco está em todos os aspectos da produção de software.

Engenharia se refere a obter resultados de qualidade requeridos dentro do cronograma e do orçamento. Isso frequentemente envolve ter compromissos (SOMMERVILLE, 2011, p. 5).

Considerando o mundo de tecnologia integrada em que vivemos, a ES é importante por dois motivos: cada vez mais pessoas dependem dos sistemas informatizados e temos que ser capazes de desenvolvê-los de forma confiável, rápida e econômica. Além disso, é mais barato quando utilizamos técnicas, métodos e processos ao invés de desenvolvê-los de forma aleatória, como se fossem um projeto pessoal sem controle de prazos e custos.

Considerando o apresentado, os engenheiros de software se preocupam em desenvolver **produtos de software**, os quais podem ser vendidos para clientes (SOMMERVILLE, 2011).

Um **produto de software** propriamente dito é sistematicamente destinado ao uso por pessoas com formações e experiências diferentes, o que significa que uma preocupação não apenas nas características de desenvolvimento, como também de interface, documentação, seja ela de sistemas ou de usuário. Sem contar que, nesse caso, deve-se testar o software exaustivamente antes de entregá-lo ao cliente para detectar e corrigir os eventuais erros de execução.

Um programa desenvolvido para resolver um dado problema (usuário restrito e único) e um produto de software destinado à resolução do mesmo problema (para vários clientes e com a meta de comercialização) são duas coisas diferentes. É óbvio que o esforço e o consequente custo associado ao desenvolvimento de um produto serão superiores aos de acesso restrito até mesmo devido à sua concepção de comercialização futura.

Existem, segundo Sommerville (2011), dois tipos de produtos de software:

1. **Produtos genéricos:** sistemas stand-alone, produzidos por uma organização de desenvolvimento e vendidos no mercado para qualquer cliente que deseja utilizá-los. Ex. ferramentas de banco de dados, processadores de texto etc.
2. **Produtos sob encomendas:** sistemas desenvolvidos para um cliente em particular. Ex. sistemas de controle de tráfego aéreo.

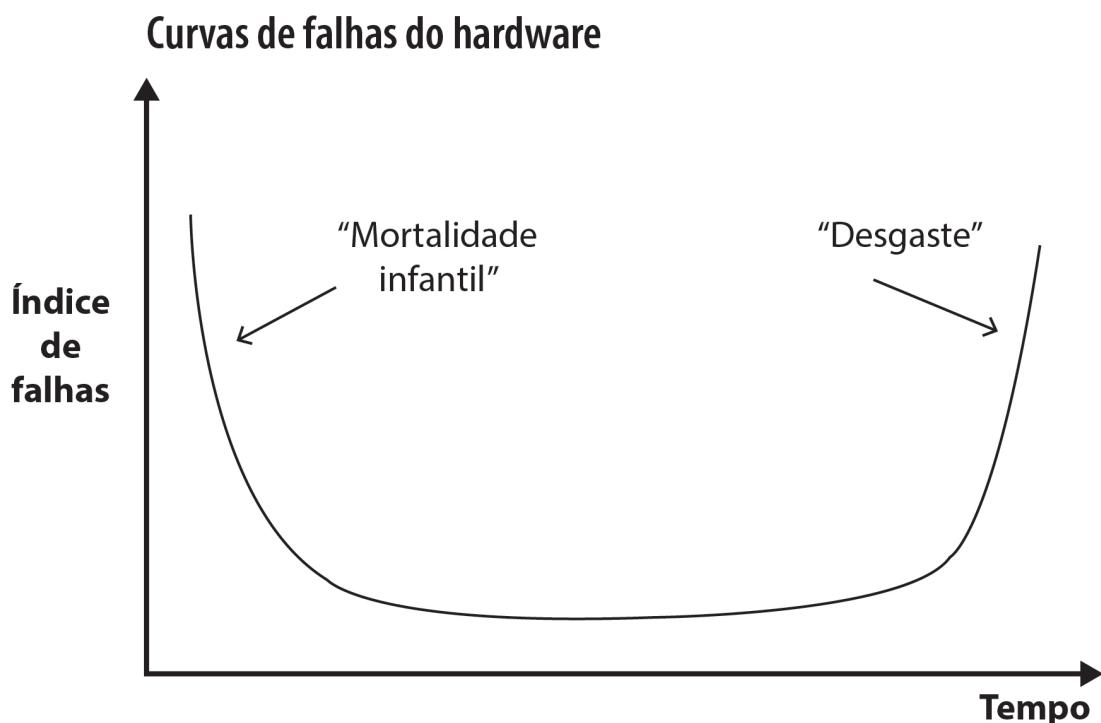
A diferença primordial nesses dois sistemas é que, no primeiro, a organização que o desenvolve controla suas especificações, no segundo é o cliente que define as especificações.

Entretanto, é importante paramos um momento para levantar algumas comparações entre produto de software que é composto por partes lógicas e físicas e outros tipos de produtos:

- **O produto de software é concebido e desenvolvido como resultado de um trabalho de engenharia** e não manufaturado no sentido clássico. Neste sentido, é interessante que você reflita novamente sobre o conceito de engenharia. Pense em uma atividade de engenharia que lhe seja familiar (engenharia civil, mecânica, produção etc.). Imagine-a como Ciência de construção, na qual, por meio de técnicas, ela pode desenvolver partes que pertencerão a um produto específico e chegue às suas próprias conclusões, fundamentado(a) no fato de que o software não é um produto manufaturado, embora possua algumas características conceituais relacionadas à manufatura.
- **O produto de software não se desgasta**, ou seja, ao contrário da maioria dos produtos, o software não se caracteriza por um aumento na possibilidade de falhas, à medida que o tempo passa, devido ao desgaste físico e à ação ambiental (como, por exemplo, a poeira e o calor). O que pode acontecer é ele se tornar obsoleto

e suas funcionalidades não mais satisfazerem à necessidade do usuário. Portanto, pode-se dizer que o software não se desgasta, mas, sim, deteriora-se. Você poderá compreender melhor as diferenças entre o desgaste do hardware e a deterioração do software observando o gráfico das curvas de falhas, proposto por Pressman (2006). O gráfico 1, a seguir, exemplifica isto:

Gráfico 1



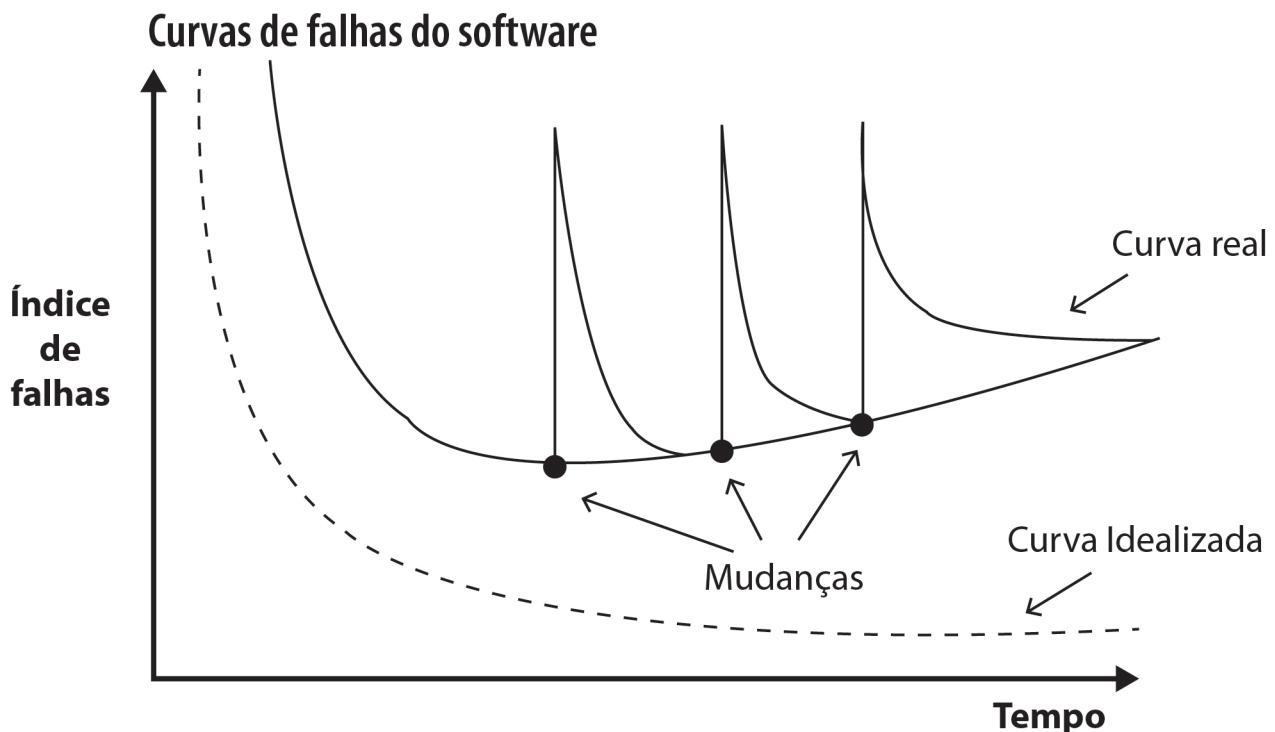
Fonte: Adaptado de Pressman (2006, p. 14).

Como você pode observar, no Gráfico 1, inicialmente, o hardware apresenta um elevado índice de falhas, as quais, normalmente, são atribuídas a problemas de projeto ou de fabricação. Com o passar do tempo, aferições são realizadas no hardware e essas falhas são corrigidas. Em seguida, são realizados ajustes e a curva se estabiliza, indicando que todos os defeitos foram eliminados, ou que o nível de falhas se encontra em um limite aceitável para o produto. No entanto, com o passar do tempo, o índice de falhas aumenta devido ao desgaste do produto, levando à sua substituição.

Observe, agora, o Gráfico 2, em que são apresentadas as duas curvas de falhas de software: **a curva real e a curva ideal**.

Na curva idealizada, representada pela linha tracejada no Gráfico 2, o software, sendo um elemento lógico, não sofre os desgastes físicos e ambientais comuns ao hardware. Assim, teoricamente, uma vez eliminadas as falhas iniciais inseridas durante o projeto e a construção do software (semelhante à construção do hardware), a curva se manterá estabilizada sem falhas ou com o menor índice possível de falhas. Portanto, o software não se deterioraria.

Gráfico 2



Fonte: Adaptado de Pressman (2006, p. 15).

No entanto, a curva ideal apresenta uma utopia do desenvolvimento de software: uma vez construído o software, todos os problemas acabaram. A realidade, porém, é bem diferente, e é mostrada na curva real de falhas do software, também no Gráfico 2. Durante a vida do software, ele passará por várias e inevitáveis mudanças. À medida que as mudanças são realizadas, novas falhas podem ser inseridas, representando na curva real os picos de aumento de índice de falhas. Antes mesmo de se eliminarem as falhas inseridas na última modificação, nova necessidade de mudança pode ocorrer. Com o passar do tempo e as frequentes modificações, o índice mínimo de falhas vai aumentando, isto é, o software deixa de atender aos requisitos para os quais foi construído. Portanto, o software não se desgasta, mas se deteriora.

- **A maioria dos produtos de software é concebida inteiramente sob medida**, sem a utilização de componentes pré-existentes.

Em função dessas características diferenciais, o **processo de desenvolvimento** de produto de software gera um conjunto de dificuldades, com influência direta na **qualidade final do produto**.

Embora hoje conceitos de componentes e reuso sejam bastante utilizados no desenvolvimento de software, eles fazem parte de um conceito maior que é o de melhorar as técnicas de desenvolvimento de software

3. A diversidade na ES, sua integração com a internet e a ética relacionada ao desenvolvimento de produto de software



Engenharia de Software é uma abordagem sistemática para produção de software; ela analisa questões práticas de custo, prazo e confiança, assim como as necessidades dos clientes e produtos do software (SOMMERVILLE, 2011, p. 6).

Existem, no mercado, segundo Sommerville (2011) e Pfleeger (2004), diferentes tipos de software comercializados:

- *Stand-alone*: aplicações executadas em um computador local. Exemplo: processador de texto.
- Aplicações interativas baseadas em transações: aplicações executadas num computador remoto. Nesta categoria temos os softwares corporativos integrados à web. Exemplo: software de e-mail.
- Sistemas de controle embutido: aplicações que controlam e gerenciam dispositivos de hardware. Exemplo: softwares de um micro-ondas, de telefone celular etc.
- Sistemas de processamento em lotes: aplicações corporativas desenvolvidas para processar grandes quantidades de informação, em lote. Exemplo: sistemas de cobranças telefônicas.
- Sistemas de entretenimento: aplicações relacionadas à diversão. Exemplo: jogos.
- Sistemas para modelagem e simulação: desenvolvidos por cientistas, compostos por vários objetos que podem ser combinados entre si.

Para cada um desses sistemas, e para outros não mencionados, não existem barreiras físicas sobre onde um tipo começa e outro termina. Por exemplo, você pode desenvolver um jogo para telefone celular no qual deverá ser cobrado um valor mensal/anual para todos os usuários do sistema de uma telefonia específica. Assim, para cada tipo de software, temos que utilizar diferentes técnicas de desenvolvimento, como veremos na próxima unidade.

Mesmo com tantas características individuais, segundo Pfleeger (2004) e Sommerville (2011), existem alguns conceitos fundamentais comuns:

- Todo produto de software deve ser desenvolvido em um processo gerenciado e compreendido. A empresa que o desenvolve deve possuir mecanismos de planejamento e controle do próprio processo de desenvolvimento.
- Todos os produtos devem possuir características como confiança e desempenho adequado, ou seja, devem se comportar conforme o esperado, sem falhas.
- É fundamental entender e gerenciar todos os requisitos do sistema a fim de garantir que o produto final seja o esperado.
- Você deve fazer o melhor uso possível dos recursos existentes e isso, hoje, significa reutilizar objetos e softwares desenvolvidos ao invés de iniciar o processo do zero.

Além dessas características, ainda temos que considerar a integração com a internet.

Se voltarmos no tempo e tentarmos montar uma linha cronológica sobre desenvolvimento de software, encontraremos acontecimentos que foram fundamentais para definir e estruturar o processo e a tecnologia existentes, principalmente em relação à integração com a internet.

Foi na década de 1940 que se iniciou a evolução dos sistemas computadorizados. Grande parte dos esforços e consequentes custos era concentrada no desenvolvimento do hardware, em razão, principalmente, de limitações e dificuldades encontradas na época. O foco era desenvolvimento de hardware, uma vez que o software tinha uma característica secundária.

À medida que a tecnologia de hardware foi dominada, as preocupações se voltaram ao software e, no início da década de 1950, o desenvolvimento dos sistemas operacionais começou a entrar em foco. Foi nessa época que as linguagens de programação de alto nível, como *Fortran* e *Cobol*, e respectivos compiladores, começaram a ser o centro da atenção.

A tendência da época foi de poupar cada vez mais o usuário de um computador de conhecer profundamente as questões relacionadas ao funcionamento interno da máquina, permitindo que ele pudesse concentrar seus esforços na resolução dos problemas computacionais, em lugar de preocupar-se com os problemas relacionados ao funcionamento da máquina (PRESSMAN, 1996).

Já no início da década de 1960, com o surgimento dos sistemas operacionais com características de multiprogramação, a eficiência e a utilidade dos sistemas computacionais tiveram um considerável crescimento. Esse fato contribuiu, de forma significativa, para a queda de preço do hardware e um aumento significativo de vendas de máquinas e software.

Uma característica desse período foi a necessidade, cada vez maior, de desenvolver grandes sistemas de software em substituição aos pequenos programas aplicativos que eram utilizados até então – a empresa conseguia comprar as máquinas e queria softwares “que funcionassem nelas”.

Disso surgiu um problema nada trivial devido à falta de experiência e a não adequação dos métodos de desenvolvimento existentes para pequenos programas. Esse problema, entre o que era necessidade de mercado e o que os desenvolvedores podiam suprir, foi caracterizado, ainda na década de 1960, como a “crise do software”.



Explore:

- **Pesquise mais sobre a crise do software da década de 1960.**

Com o passar do tempo, o preço de hardware foi, significativamente, baixando enquanto o de software não obedeceu a essa mesma tendência.

Por volta de 2000, a Internet começou a evoluir, e mais e mais recursos passaram a ser adicionados aos navegadores. Isso significou uma portabilidade diferente, ou seja, foi possível desenvolver um sistema e utilizar a interface do navegador para acessá-lo. Isso levou ao desenvolvimento de um número significativo de sistemas que puderam ser acessados via web. Outra característica é que não apenas foi possível acessá-lo a distância, mas também dar manutenção, atualizá-lo, excluí-lo. Isso também reduziu custos de produção, de manutenção e de testes.

Considerando essa evolução, é possível perceber que junto com essa conexão de sistemas também temos conceitos de ética relacionados.

Existem, na área de desenvolvimento de software, relações de confiabilidade, competência, uso inadequado de sistemas etc. Enfim, existe um código ético social além daqueles formalmente definidos, como o de direito de propriedade intelectual, que deve ser respeitado dentro dos parâmetros éticos da profissão.

Sociedades e instituições profissionais têm um papel importante a desempenhar na definição de padrões éticos. Organizações como ACM, IEEE (*Institute of Electrical em Eletronic Engineers*) e British Computer Society publicam códigos de conduta profissional, ou códigos de ética (SOMMERVILLE, 2011, p. 9).

Pessoas que se associam a essas instituições se comprometem a seguir tais códigos. A filiação a elas é comum a todos os profissionais de ES, uma vez que isso traz credibilidade mercadológica à empresa e ao produto desenvolvido.



Explore

IEEE, ACM, British Computer Society

- <http://www.computer.org/portal/web/about/sistersocieties>.

Hoje, o software corresponde a uma percentagem cada vez maior no custo global de um sistema informatizado; principalmente porque a tecnologia de desenvolvimento de software implica, ainda, em grande carga de trabalho, geralmente a um grande número de pessoas trabalhando juntas e num prazo relativamente pequeno de desenvolvimento. A elaboração desses sistemas é realizada, na maior parte das vezes, de forma *ad hoc*, o que pode conduzir a uma extração do tempo definido inicialmente para o desenvolvimento, o que acarreta aumento no custo final do sistema.

4. Reengenharia e Engenharia Reversa



Além do conceito de Engenharia de Software, há também dois outros conceitos relacionados: reengenharia e engenharia reversa.

O conceito de reengenharia está bastante relacionado à manutenção do software.

Os sistemas computacionais desenvolvidos para apoiar as atividades empresariais, chamados de sistemas de informação, devem se adaptar às mudanças ocorridas na empresa, para que continuem a atender às necessidades dos usuários. Esses sistemas são os que normalmente mais sofrem mudanças depois de sua implantação.

Um sistema de informação pode ser definido tecnicamente como um conjunto de componentes inter-relacionados que coleta (ou recupera), processa, armazena e distribui informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização. Além de dar suporte à tomada de decisões, à coordenação e ao controle, esses sistemas também auxiliam os gerentes e trabalhadores a analisar problemas, visualizar assuntos complexos e criar novos produtos. (LAUDON; LAUDON, 2006, p. 7)

Segundo Laudon e Laudon (1999), o ambiente empresarial está se alterando e é fluido, isto é, novas tecnologias, tendências econômicas, desenvolvimentos políticos e regulamentações que afetam os negócios estão constantemente emergindo. Geralmente, quando as empresas falham, é porque negligenciaram a resposta ao ambiente mutável.

A empresa que não se adaptar às mudanças ambientais deixará de ser competitiva. Da mesma forma, os sistemas de informação dessas empresas devem também se adaptar às suas novas necessidades, para que continuem úteis e cumpram seu papel de apoiar as decisões.

Porém, muitos dos sistemas de informação fundamentais aos negócios estão se tornando de difíceis alterações. Para Pressman (1996), remendos são feitos sobre remendos, resultando em softwares que funcionam de forma ineficiente e falham com frequência, não correspondendo às necessidades dos usuários. Portanto, a manutenção de sistemas em fase de envelhecimento tornou-se proibitivamente dispendiosa para muitos sistemas de informação.

Quando, ao longo das manutenções, as alterações são feitas sem administração, sem atualização da documentação, na verdade, quando são feitos “remendos”, à medida que o tempo passa, novas alterações ficam cada vez mais difíceis de serem feitas. Além disso, sistemas elaborados com tecnologia muito arcaica, muitas vezes, não acomodam as necessidades atuais. É exatamente nesses casos que é feita a reengenharia do software. Reengenharia, em poucas palavras, significa reconstruir o software utilizando uma nova tecnologia, para melhorar a facilidade de manutenção.

De acordo com Pressman (1996), a reengenharia não somente recupera as informações de projeto de um software existente, mas usa essas informações para alterar ou reconstruir o sistema existente, num esforço para melhorar sua qualidade. Logo, um sistema que tenha sofrido reengenharia reimplementa a função do sistema existente, acrescido de novas funções.

A reengenharia pode envolver redocumentar, organizar e reestruturar o sistema, traduzir o sistema para uma linguagem de programação mais moderna e atualizar a estrutura e os valores dos dados do sistema. (SOMMERVILLE, 2005, p. 533)

A engenharia reversa tem sua origem no mundo do hardware. Uma empresa desmonta o produto de um concorrente para entender os segredos do projeto e da fabricação do mesmo. Esses segredos poderiam ser obtidos através das especificações do projeto e fabricação do produto, mas tais documentos não são disponíveis à empresa que está fazendo a engenharia reversa. A engenharia reversa de software é bastante semelhante. No entanto, na maioria das vezes, o software que passa pela engenharia reversa não é o de um concorrente, mas é trabalho desenvolvido pela própria empresa há muitos anos. Os segredos a serem descobertos são obscuros porque não houve, em casos assim, documentação (PRESSMAN, 2006).

Logo,

[...] a engenharia reversa é o processo de análise de um programa, em um esforço para representá-lo em uma abstração mais alta do que o código-fonte. A engenharia reversa é um processo de recuperação de projeto. (PRESSMAN, 2006, p. 687)

Para Sommerville (2005, p. 534), na “[...] engenharia reversa o programa é analisado e as informações são extraídas dele, a fim de ajudar a documentar sua organização e funcionalidade”.

A engenharia reversa não é o mesmo que reengenharia. O objetivo da engenharia reversa é derivar o projeto ou a documentação de um sistema a partir de seu código-fonte. Já o objetivo da reengenharia é produzir um novo sistema com manutenção mais fácil. Assim, a engenharia reversa para desenvolver uma melhor compreensão de um sistema é com frequência parte do processo de reengenharia (SOMMERVILLE, 2005).

Em outras palavras, a engenharia reversa pode anteceder um processo de reengenharia, num esforço de compreender os requisitos que levaram à construção do software a ser reconstruído.

Sendo assim, podemos concluir que a Engenharia de Software é o processo de desenvolver o software a partir de um estudo das necessidades do cliente e futuros usuários.

O processo de reengenharia de software tem como partida o conhecimento das características do sistema antigo a ser reconstruído. As características do sistema antigo podem ser levantadas com o auxílio da engenharia reversa.

Na nossa próxima unidade, apresentaremos os **modelos de processo de desenvolvimento de software**.

Material Complementar

O objetivo do material complementar é lhe ajudar a entender, sob uma ótica diferente daquela da professora conteudista, assuntos abordados nas unidades teóricas.

É fundamental a leitura deste material para o melhor entendimento sobre o assunto.



Explore

Como nesta unidade abordamos os conceitos gerais da Engenharia de Software, nossa sugestão de material complementar é a unidade introdutória da obra:

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2003. p. 4-16.

Referências

Bibliografia Fundamental

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

Bibliografia Básica

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação**. 4. ed. Rio de Janeiro: LTC, 1999.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais. Administrando a empresa digital**. 5. ed. São Paulo: Pearson Education do Brasil, 2006.

PFLEEGER, S. L. **Engenharia de Software: teoria e prática**. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 2006.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Pearson Addison Wesley, 2005.

Bibliografia Complementar

ALCADE, E.; GARCIA, M.; PENUELAS, S. **Informática Básica**. São Paulo: Makron Books, 1991.

FAIRLEY, R. E. **Software engineering concepts**. New York: McGraw-Hill, 1987.

IEEE **Software Engineering Standards (2013)**. Disponível em: <http://www.ieee.org/portal/innovate/products/standard/ieee_soft_eng.html>. Acesso em: 10 dez. 2013.

LUKOSEVICIUS, A. P.; CAMPOS FILHO, A. N.; COSTA, H. G. **Maturidade em Gerenciamento de Projetos e Desempenho dos Projetos**. Disponível em: <www.producao.uff.br/conteudo/rpep/.../RelPesq_V7_2007_07.doc>. Acesso em 12 nov. 2013.

MAFFEO, B. **Engenharia de software e especialização de sistemas**. Rio de Janeiro: Campus, 1992.

MICHAELIS. **Moderno dicionário da língua portuguesa**. São Paulo: Cia. Melhoramentos, 1998.

PARREIRA JÚNIOR, W. M. O. P. **Apostila de Engenharia de software**. Disponível em: <http://www.waltenomartins.com.br/ap_es_v1.pdf>. Acesso em: 13 nov. 2013.

PAULA FILHO, W. P. **Engenharia de Software: fundamentos, métodos e padrões**. 2. ed. Rio de Janeiro: LTC, 2001.

Revista Engenharia de Software. Disponível em: <<http://www.devmedia.com.br/revista-engenharia-de-software-magazine>>. Acesso em: 12 nov. 2013.

VONSTA, A. **Engenharia de programas.** Rio de Janeiro: LTC, 1983.

WIENNER, R.; SINCOVEC, R. **Software engineering with Modula 2 and ADA.** New York: Wiley, 1984.

WIKIPEDIA (2007a). **ISO 9000.** Disponível em: <http://pt.wikipedia.org/wiki/ISO_9000>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007b). **Melhoria de processo de software brasileiro.** Disponível em: <<http://pt.wikipedia.org/wiki/MPS.BR>>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007c). **CMMI.** Disponível em: <<http://pt.wikipedia.org/wiki/Cmmi>>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007d). **Engenharia de Software.** Disponível em: <http://pt.wikipedia.org/wiki/Engenharia_de_software>. Acesso em: 1 fev. 2007.

Anotações





Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000



Universidade
Cruzeiro do Sul



UNICID
Universidade
Cidade de S. Paulo



UNIFRAN
Universidade
de Franca



UDF
Centro
Universitário



Módulo
Centro
Universitário