Software e Engenharia de Software

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Software

Definindo software

Software de computador é o produto que profissionais da tecnologia da informação desenvolvem e ao qual dão suporte no longo prazo. Abrange programas executáveis em um computador de qualquer porte ou arquitetura, conteúdos, informações descritivas tanto na forma impressa (*hard copy*) como na virtual, abrangendo praticamente qualquer mídia eletrônica.

Definindo software

Software consiste em:

1. instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados;
2. estruturas de dados que possibilitam aos programas manipular informações adequadamente;
3. informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Definindo software

Características:

- Software é desenvolvido ou passa por um processo de engenharia; ele não é fabricado no sentido clássico.
    - Embora existam algumas similaridades entre o desenvolvimento de software e a fabricação de hardware, as duas atividades são fundamentalmente diferentes.
    - Em ambas, alta qualidade é obtida por meio de bom projeto, entretanto, a fase de fabricação de hardware pode gerar problemas de qualidade inexistentes (ou facilmente corrigíveis) para software.
    - Ambas requerem a construção de um “produto”, entretanto, as abordagens são diferentes.
    - Os custos de software concentram-se no processo de engenharia.

Definindo software

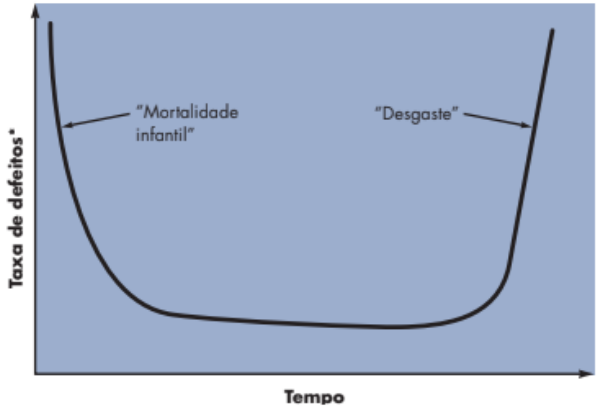
Características:

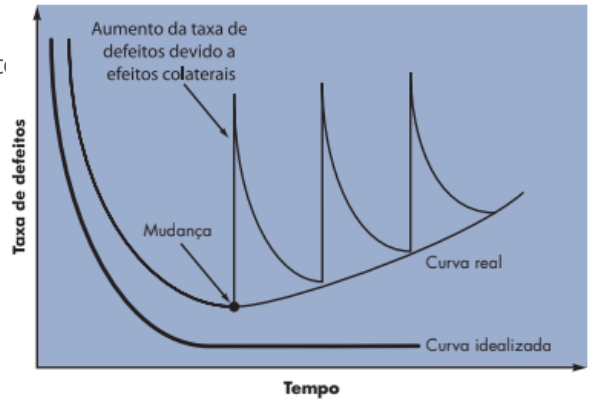
- Software não “se desgasta”.
  - Software não é suscetível aos males ambientais que fazem com que o hardware se desgaste.
  - Não existem peças de reposição de software.
  - Cada defeito de software indica um erro no projeto ou no processo pelo qual o projeto foi traduzido em código de máquina executável.

Definindo software

Características:

- Software não “se desgasta”.
  - Software não é suscetível aos males ambientais que fazem com que o hardware se desgaste.
  - [illegible]
  - projeto [illegible] zido





Definindo software

Características:

- Embora a indústria caminhe para a construção com base em componentes, a maioria dos softwares continua a ser construída de forma personalizada (sob encomenda).
  - À medida que a disciplina da engenharia evolui, uma coleção de componentes de projeto padronizados é criada.
  - Um componente de software deve ser projetado e implementado de modo que possa ser reutilizado em muitos programas diferentes.

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software de sistema

Conjunto de programas feito para **atender a outros programas**. Certos softwares de sistema (por exemplo, compiladores, editores e utilitários para gerenciamento de arquivos) processam estruturas de informação complexas, porém, determinadas.

Outras aplicações de sistema (por exemplo, componentes de sistema operacional, drivers, software de rede, processadores de telecomunicações) processam dados amplamente indeterminados.

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software de aplicação

Programas sob medida que solucionam uma **necessidade específica de negócio**. Aplicações nessa área processam dados comerciais ou técnicos de uma forma que facilite operações comerciais ou tomadas de decisão administrativas/técnicas.

Exemplo: processamento de transações em pontos de venda, controle de processos de fabricação em tempo real.

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software científico/de engenharia

Tem sido caracterizado por algoritmos “***number crunching***” (para “processamento numérico pesado”). As aplicações vão da astronomia à vulcanologia, da análise de tensões na indústria automotiva à dinâmica orbital de ônibus espaciais, e da biologia molecular à fabricação automatizada.

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software embutido

Residente num produto ou sistema é utilizado para implementar e controlar características e funções para o usuário final e para o próprio sistema.

Exemplos:

- Funções limitadas e específicas (controle do painel de um forno micro-ondas)
- Função significativa e capacidade de controle (funções digitais de automóveis, tal como controle do nível de combustível, painéis de controle e sistemas de freios).

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software para linha de produtos

Projetado para prover capacidade específica de utilização por **muitos clientes** diferentes. Pode focalizar um mercado limitado e particularizado ou direcionar-se para mercados de consumo de massa.

Exemplos :

- Produtos para controle de estoques
- Processamento de texto
- planilhas eletrônicas
- computação gráfica
- multimídia
- entretenimento
- gerenciamento de bancos de dados
- aplicações financeiras pessoais e comerciais

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Aplicações para a Web

Chamadas de “*WebApps*”, essa categoria de software **centralizada em redes** abarca uma vasta gama de aplicações. Em sua forma mais simples, as WebApps podem ser pouco mais que um conjunto de arquivos de hipertexto interconectados, apresentando informações por meio de texto e informações gráficas limitadas.

Campos de aplicação de software

Pode-se dividir em sete grandes categorias:

- Software de sistema
- Software de aplicação
- Software científico/de engenharia
- Software embutido
- Software para linha de produtos
- Aplicações para a Web
- Software de inteligência artificial

Software de inteligência artificial

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

Software legado

Sistemas de software legado... Foram desenvolvidos **décadas atrás** e têm sido continuamente modificados para se adequar a mudanças dos requisitos de negócio e a plataformas computacionais. A proliferação de tais sistemas está causando dores de cabeça para grandes organizações que os consideram dispendiosos de manter e arriscados de evoluir.

Software legado

Os sistemas legados, algumas vezes, têm projetos não expansíveis, código intrincado, documentação pobre ou inexistente, casos de testes e resultados que nunca foram arquivados, um histórico de modificações mal administrado — a lista pode ser bem longa. Ainda assim, esses sistemas dão suporte a “funções vitais de negócio e são indispensáveis para ele”.

O que fazer então?

Software legado

A única resposta razoável talvez seja: Não faça nada, pelo menos até que o sistema legado tenha que passar por alguma modificação significativa. Se o software legado atende às necessidades de seus usuários e roda de forma confiável, ele não está “quebrado” e não precisa ser “consertado”.

A Natureza Única da s Webapps

Nos primórdios da *World Wide Web* (por volta de 1990 a 1995), os sites eram formados por nada mais que um conjunto de arquivos de hipertexto lincados que apresentavam informações usando texto e gráficos limitados.

WebApps são apenas uma dentre uma série de diferentes categorias de software. Ainda assim, pode-se afirmar que elas são diferentes.

Sistemas e aplicações baseados na Web “envolvem uma mistura de publicação impressa e desenvolvimento de software, de marketing e computação, de comunicações internas e relações externas e de arte e tecnologia”.

A Natureza Única da s Webapps

Os seguintes atributos são encontrados na grande maioria das WebApps:

- **Uso intensivo de redes**
  - Uma WebApp reside em uma rede e deve atender às necessidades de uma comunidade diversificada de clientes.
- **Simultaneidade**
  - Um grande número de usuários pode acessar a WebApp ao mesmo tempo. Em muitos casos, os padrões de utilização entre os usuários finais variam amplamente.
- **Carga não previsível**
  - O número de usuários da WebApp pode variar, em ordem de grandeza, de um dia para outro.

A Natureza Única da s Webapps

Os seguintes atributos são encontrados na grande maioria das WebApps:

- **Desempenho**
  - Se um usuário de uma WebApp tiver de esperar muito (para acesso, processamento no servidor, formatação e exibição no cliente), talvez ele procure outra opção,
- **Disponibilidade**
  - Embora a expectativa de 100% de disponibilidade não seja razoável, usuários de WebApps populares normalmente exigem acesso 24 horas por dia, 7 dias por semana, 365 dias por ano,
- **Orientadas a dados**
  - A função principal de muitas WebApps é usar hipermídias para apresentar texto, gráficos, áudio e vídeo para o usuário final.

A Natureza Única da s Webapps

Os seguintes atributos são encontrados na grande maioria das WebApps:

- **Sensibilidade no conteúdo**
  - A qualidade e a natureza estética do conteúdo são fatores importantes que determinam a qualidade de uma WebApp.
- **Evolução contínua**
  - Diferentemente de softwares de aplicação convencionais que evoluem ao longo de uma série de versões planejadas e cronologicamente espaçadas, as WebApps evoluem continuamente.

A Natureza Única da s Webapps

Os seguintes atributos são encontrados na grande maioria das WebApps:

- **Imediatismo**
  - Embora imediatismo — a imperativa necessidade de colocar rapidamente um software no mercado — seja uma característica de diversos campos de aplicação, as WebApps normalmente apresentam um tempo de colocação no mercado que pode consistir de poucos dias ou semanas,
- **Segurança**
  - Pelo fato de estarem disponíveis via acesso à Internet, torna-se difícil, senão impossível, limitar o número dos usuários finais que podem acessar as WebApps.

A Natureza Única da s Webapps

Os seguintes atributos são encontrados na grande maioria das WebApps:

- **Estética**
  - Parte inegável do apelo de uma WebApp consiste na sua aparência e na impressão que desperta. Quando uma aplicação é desenvolvida para o mercado ou para vender produtos ou ideias, a estética pode ser tão importante para o sucesso quanto o projeto técnico.

Engenharia de Software

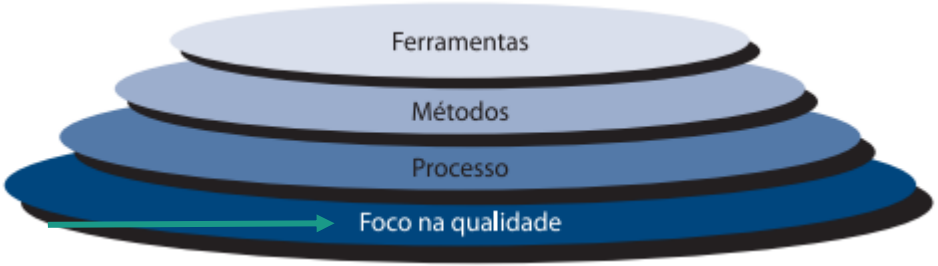
Engenharia de Software

Engenharia de software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais. [Fritz Bauer]

Engenharia de software: A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software. [IEEE]

Engenharia de Software

A engenharia de software é uma tecnologia em camadas. Qualquer abordagem de engenharia (inclusive engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade. A pedra fundamental que sustenta a engenharia de software é o foco na **qualidade.** [PRESSMAN]

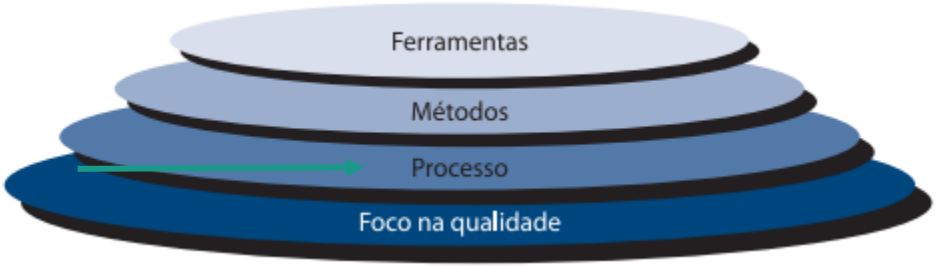


Engenharia de Software

A base para a engenharia de software é a camada de **processos**. O processo de engenharia de software é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo. O processo define uma metodologia que deve ser estabelecida para a entrega efetiva de tecnologia de engenharia de software.

Engenharia de Software

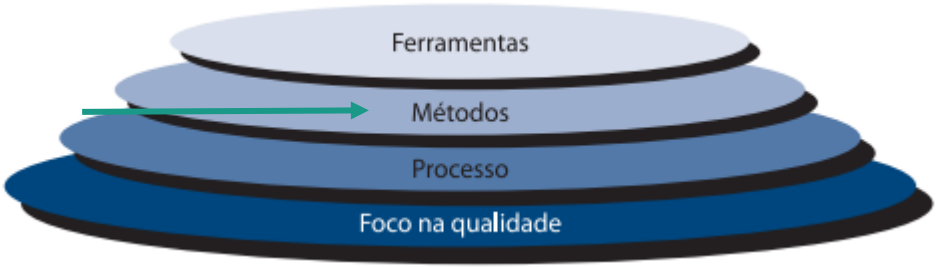
O **processo** de software constitui a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos produtos derivados (modelos, documentos, dados, relatórios, formulários etc.), são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada.



Engenharia de Software

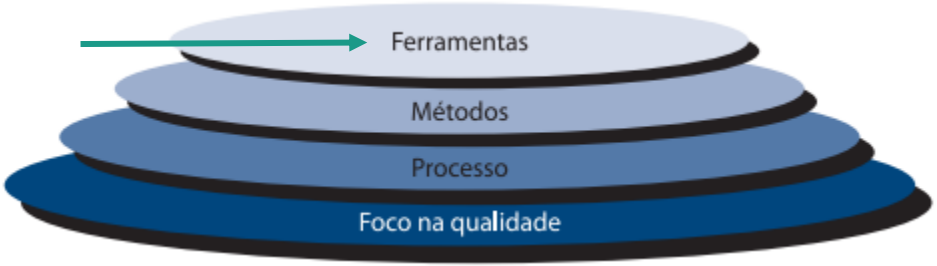
Os **métodos** da engenharia de software fornecem as informações técnicas para desenvolver software. Os métodos envolvem uma ampla gama de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte.

Os **métodos** da engenharia de software baseiam-se em um conjunto de princípios básicos que governam cada área da tecnologia e inclui atividades de modelagem e outras técnicas descritivas.



Engenharia de Software

As ferramentas da engenharia de software fornecem suporte automatizado ou semi automatizado para o processo e para os métodos.



O Processo de Software

Processo é um conjunto de **atividades**, **ações** e **tarefas** realizadas na criação de algum produto de trabalho (*work product*).

No contexto da engenharia de software, um processo não é uma prescrição rígida de como desenvolver um software. Ao contrário, é uma **abordagem adaptável** que possibilita às pessoas (a equipe de software) realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas.

A intenção é a de sempre entregar software dentro do **prazo e com qualidade** suficiente para satisfazer àqueles que patrocinaram sua criação e àqueles que irão utilizá-lo.

O Processo de Software

- **atividade:** esforça-se para atingir um objetivo amplo (por exemplo, comunicar-se com os interessados) e é utilizada independentemente do campo de aplicação, do tamanho do projeto, da complexidade de esforços ou do grau de rigor com que a engenharia de software será aplicada.
- **ação:** (por exemplo, projeto de arquitetura) envolve um conjunto de tarefas que resultam num artefato de software fundamental (por exemplo, um modelo de projeto de arquitetura).
- **tarefa:** se concentra em um objetivo pequeno, porém, bem definido (por exemplo, realizar um teste de unidades) e produz um resultado tangível.

O Processo de Software

Uma **metodologia** (*framework*) de processo estabelece o alicerce para um processo de engenharia de software completo, por meio da identificação de um pequeno número de **atividades estruturai**s aplicáveis a todos os projetos de software, independentemente de tamanho ou complexidade.

Além disso, a metodologia de processo engloba um conjunto de **atividades** de apoio (*umbrella activities*— abertas) aplicáveis em todo o processo de software.

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Comunicação

Antes de iniciar qualquer trabalho técnico, é de vital importância **comunicar-se e colaborar com o cliente** (e outros interessados). A intenção é compreender os objetivos das partes interessadas para com o projeto e fazer o levantamento das necessidades que ajudarão a definir as funções e características do software.

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Planejamento

Qualquer jornada complicada pode ser simplificada caso exista um mapa. Um projeto de software é uma jornada complicada, e a atividade de planejamento cria um “mapa” que ajuda a guiar a equipe na sua jornada.

O mapa — denominado plano de projeto de software — define o trabalho de engenharia de software, descrevendo as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a ser produzidos e um cronograma de trabalho.

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Modelagem

Independentemente de ser um paisagista, um construtor de pontes, um engenheiro aeronáutico, um carpinteiro ou um arquiteto, trabalha-se com modelos todos os dias. Cria-se um “esboço” da coisa, de modo que se possa ter uma ideia do todo - qual será o seu aspecto em termos de arquitetura, como as partes constituintes se encaixaram e várias outras características.

Se necessário, refina-se o esboço com mais detalhes, numa tentativa de compreender melhor o problema e como resolvê-lo.

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Construção

Essa atividade combina geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação.

O Processo de Software

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Emprego

O software (como uma entidade completa ou como um incremento parcialmente efetivado) é entregue ao cliente, que avalia o produto entregue e fornece feedback, baseado na avaliação.

Processo de Software

Para muitos projetos de software, as atividades metodológicas são aplicadas iterativamente conforme o projeto se desenvolve. Ou seja, **comunicação**, **planejamento**, **modelagem**, **construção** e **emprego** são aplicados repetidamente quantas forem as iterações do projeto, sendo que cada iteração produzirá um incremento de software.

Processo de Software

As atividades metodológicas do processo de engenharia de software são complementadas por uma série de **atividades de apoio**, em geral, estas são aplicadas ao longo de um projeto, ajudando a equipe a gerenciar, a controlar o progresso, a qualidade, as mudanças e o risco.

Processo de Software

As **atividades** de apoio típicas são:

- **Controle e acompanhamento do projeto**
  - possibilita que a equipe avalie o progresso em relação ao plano do projeto e tome as medidas necessárias para cumprir o cronograma.
- **Administração de riscos**
  - avalia riscos que possam afetar o resultado ou a qualidade do produto/projeto.
- **Garantia da qualidade de software**
  - define e conduz as atividades que garantem a qualidade do software.

Processo de Software

As **atividades** de apoio típicas são:

- **Revisões técnicas**
  - avaliam artefatos da engenharia de software, tentando identificar e eliminar erros antes que se propaguem para a atividade seguinte.
- **Medição**
  - define e coleta medidas (do processo, do projeto e do produto). Auxilia na entrega do software de acordo com os requisitos; pode ser usada com as demais atividades (metodológicas e de apoio).
- **Gerenciamento da configuração de software**
  - gerencia os efeitos das mudanças ao longo do processo.

Processo de Software

As **atividades** de apoio típicas são:

- **Gerenciamento da reusabilidade**
  - define critérios para o reúso de artefatos (inclusive componentes de software) e estabelece mecanismos para a obtenção de componentes reutilizáveis.
- **Preparo e produção de artefatos de software**
  - engloba as atividades necessárias para criar artefatos como, por exemplo, modelos, documentos, logs, formulários e listas.

Princípios Gerais

David Hooker propôs sete princípios que se concentram na prática da engenharia de software como um todo:

- Primeiro princípio: a razão de existir
- Segundo princípio: KISS
- Terceiro princípio: mantenha a visão
- Terceiro princípio: mantenha a visão
- Quinto princípio: esteja aberto para o futuro
- Sexto princípio: planeje com antecedência, visando a reutilização
- Sétimo princípio: pense!

Primeiro princípio: a razão de existir

Um sistema de software existe por uma única razão: **gerar valor a seus usuários**. Todas as decisões deveriam ser tomadas tendo esse princípio em mente. Antes de especificar uma necessidade de um sistema, antes de indicar alguma parte da funcionalidade de um sistema, antes de determinar as plataformas de hardware ou os processos de desenvolvimento, pergunte a si mesmo: **“Isso realmente agrega valor real ao sistema?”**. Se a resposta for “não”, não o faça. **Todos os demais princípios se apoiam neste primeiro.**

Segundo princípio: KISS (*Keep It Simple, Stupid!*, ou seja: Faça de forma simples, tapado!)

O projeto de software não é um processo casual; há muitos fatores a serem considerados em qualquer esforço de projeto — t**odo projeto deve ser o mais simples possível**, mas não tão simples assim. Esse princípio contribui para um sistema mais fácil de compreender e manter. Isso não significa que características, até mesmo as internas, devam ser descartadas em nome da simplicidade.

De fato, frequentemente os projetos mais elegantes são os mais simples, o que não significa "**rápido e malfeito**" — na realidade, simplificar exige muita análise e trabalho durante as iterações, sendo que o resultado será um software de fácil manutenção e menos propenso a erros.

Terceiro princípio: mantenha a visão

Uma visão clara é essencial para o sucesso. Sem ela, um projeto se torna ambíguo. Sem uma integridade conceitual, corre-se o risco de transformar o projeto numa colcha de retalhos de projetos incompatíveis, unidos por parafusos inadequados... Comprometer a visão arquitetônica de um sistema de software debilita e até poderá destruir sistemas bem projetados.

Quarto princípio: o que um produz outros consomem

Raramente um sistema de software de força industrial é construído e utilizado de forma isolada. De uma maneira ou de outra, alguém mais irá usar, manter, documentar ou, de alguma forma, dependerá da capacidade de entender seu sistema. Portanto, **sempre especifique, projete e implemente ciente de que alguém mais terá de entender o que você está fazendo**.

Quinto princípio: esteja aberto para o futuro

Um sistema com tempo de vida mais longo tem mais valor. Nos ambientes computacionais de hoje, em que as especificações mudam de um instante para outro e as plataformas de hardware se tornam rapidamente obsoletas, a vida de um software, em geral, é medida em meses.

Jamais faça projetos limitados, sempre pergunte “e se” e prepare-se para todas as possíveis respostas, criando sistemas que resolvam o problema geral, não apenas aquele específico. Isso muito provavelmente conduziria à reutilização de um sistema inteiro.

Sexto princípio: planeje com antecedência, visando a reutilização

A reutilização economiza tempo e esforço. Alcançar um alto grau de reúso é indiscutivelmente a meta mais difícil de ser atingida ao se desenvolver um sistema de software. A reutilização de código e projetos tem sido proclamada como o maior benefício do uso de tecnologias orientadas a objetos, entretanto, o retorno desse investimento não é automático.

Planejar com antecedência para o reuso reduz o custo e aumenta o valor tanto dos componentes reutilizáveis quanto dos sistemas aos quais eles serão incorporados.

Sétimo princípio: pense!

Este último princípio é, provavelmente, aquele que é mais menosprezado. **Pensar bem e de forma clara antes de agir quase sempre produz melhores resultados.**

Quando se analisa alguma coisa, provavelmente esta sairá correta. Ganha-se também conhecimento de como fazer correto novamente. Se você realmente analisar algo e mesmo assim o fizer da forma errada, isso se tornará uma valiosa experiência. Um efeito colateral da análise é aprender a reconhecer quando não se sabe algo, e até que ponto poderá buscar o conhecimento.

Exercicios

Resolva os exercícios de 1-10 da lista de exercícios I.