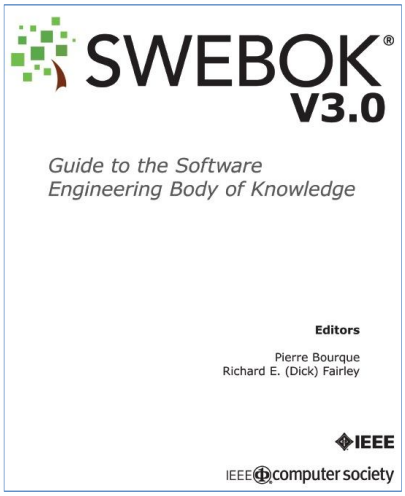
Construção de Software

Fundamentos da Construção de Software

Prof. Rubens de Castro Pereira, Me.
rubens@inf.ufg.br

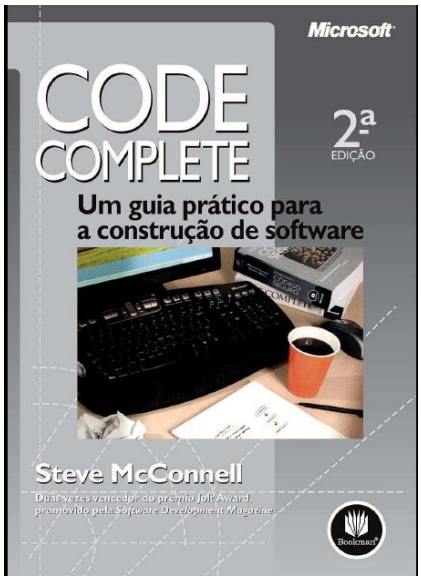
INSTITUTO DE INFORMÁTICA
UFG



Chapter 3

Software Construction

IEEE Computer Society

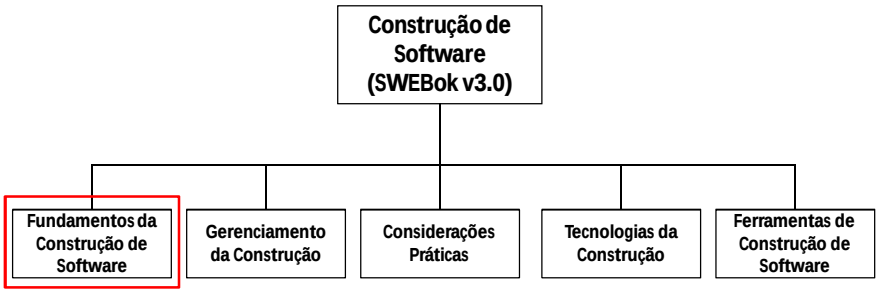


Code Complete
Um guia prático para a construção de software, 2ª ed.

Steven McConnell

Bookman

Construção de Software
Introdução



Fonte: SWEBOK - Guide to the Software Engineering Body of Knowledge Version 3.0.

Fundamentos da Construção de Software

1. Minimizando Complexidade;
2. Antecipando Mudanças;
3. Construindo para Verificação;
4. Reuso;
5. Padrões de Construção.

1. Minimizando Complexidade

- As pessoas são limitadas na capacidade de manter estruturas e informações complexas na memória de trabalho, sobretudo por longo período de tempo;
- A necessidade de reduzir a complexidade aplica-se à construção e testes de software
- É alcançada enfatizando a criação de código simples e legível ao invés de código muito inteligente;
- É alcançada usando:
  - Padrões de construção;
  - Desenho modular;
  - Outras técnicas para a melhor codificação do sistema.

1. Minimizando Complexidade

Figura 2-3 A pena por um erro cometido em uma estrutura simples é somente a perda de um pouco de tempo e talvez algum constrangimento.

1. Minimizando Complexidade

Metáforas para um melhor entendimento do desenvolvimento de software (cap. 2, McConnell)

- Metáforas na computação : Sala estéril, vírus, cavalos de tróia, *bugs*, *worms*, incêndio, erros fatais ...
- Metáforas visuais descrevem fenômenos de software específicos.
- Importância: contribuem para um melhor entendimento dos problemas de desenvolvimento de software .
- As metáforas têm como virtude:
  - Um comportamento esperado que é compreendido por todos.
  - A comunicação desnecessária e os mal-entendidos são reduzidos.
  - O aprendizado e a educação são mais rápidos.
  - São uma maneira de interiorizar e abstrair conceitos, permitindo que o pensamento de alguém esteja em um plano mais alto e que os enganos de baixo nível sejam evitados.
- Caligrafia de software: escrita de software
- Cultivo de software: criação de sistemas
- Cultura de ostras de software: incremento de sistemas
- Construção de software: edificação de software
- Aplicação de técnicas de software: a caixa de ferramentas intelectual

1. Minimizando Complexidade

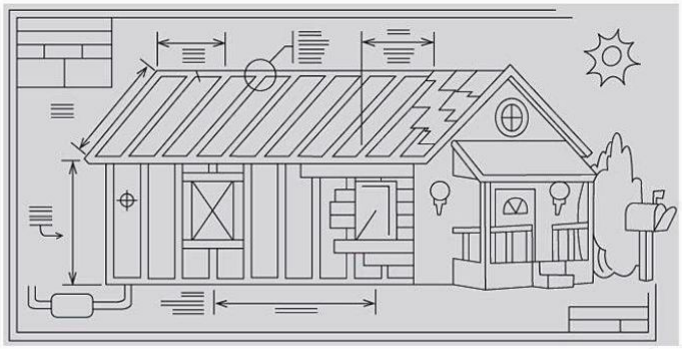


Figura 2-4 Estruturas mais complexas exigem um planejamento mais criterioso.

2. Antecipando Mudanças

- A grande maioria dos sistemas são alterados ao longo do tempo;
- Antecipar as mudanças pode direcionar a construção do software, tornando-o mais adaptável/ajustável a elas;
- Mudanças no ambiente operacional também afetam o sistema em diversas maneiras;
- Pensar em mudanças auxilia os engenheiros de software na construção de software extensível, possibilitando melhorias sem afetar a estrutura.

Pré-requisitos da Construção

- Pré-requisitos para a Definição do Problema
- Pré-requisitos para os Requisitos
- Pré-requisitos para a Arquitetura

Pré-requisitos para a Definição do Problema

- Exposição clara do problema que o sistema deve resolver.
- Sinônimos: Visão do produto, Exposição da visão, Exposição da missão, Definição do produto e Documento de visão.
- Visa resolver o problema certo.

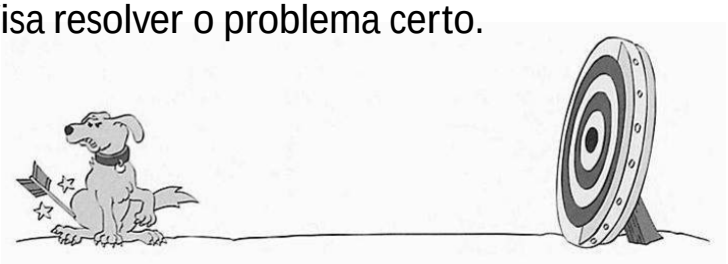


Figura 3-5 Certifique-se de que sabe para que alvo está apontando antes de atirar.

Pré-requisitos para os Requisitos

- Elaborar uma lista de requisitos explícitos.
- Nem sempre os requisitos são estáveis e, portanto, deve-se estar atento às mudanças que ocorrem nesta fase.
- Lista de verificação de requisitos.

Figura 3-6 Sem bons requisitos, você poderá reconhecer o problema em seus aspectos gerais, mas falhar em identificar seus aspectos específicos.

13

Pré-requisitos para a Arquitetura

- A Arquitetura é a etapa de alto nível do projeto de software que acomodará as partes mais detalhadas do *design*.
- Sinônimos: arquitetura do sistema, *design* de alto nível, *design* de nível superior.
- A qualidade da arquitetura determina a integridade conceitual do sistema, culminando na qualidade final do sistema.
- Desmembra o trabalho de modo eu vários desenvolvedores ou várias equipes de desenvolvimento possam trabalhar independente.

Pré-requisitos para a Arquitetura

Figura 3-7 Sem uma boa arquitetura de *software*, você pode ter o problema certo, mas a solução errada. Dificilmente a construção será bem-sucedida.

- Organização do programa
- Classes principais
- Projeto dos dados
- Regras de negócio
- Projeto da interface com o usuário
- Gerenciamento de recursos
- Segurança
- Desempenho
- Extensibilidade (crescimento futuro)
- Interoperabilidade
- Internacionalização/Localização (idioma local)
- Entrada/saída
- Processamento de erros
- Tolerância a falhas
- Praticabilidade arquitetônica
- A decisão entre comprar e construir
- Decisões de reutilização
- Estratégia de alteração
- Qualidade arquitetônica geral

3. Construindo para Verificação

- Construção de software de modo que as falhas possam facilmente ser localizadas pelos engenheiros de software, testados e usuários.
- Técnicas específicas que suportam a construção para verificação incluem padrões de codificação para suportar revisões de código e testes unitários, testes automatizados e restringindo o uso de estruturas complexas ou de difícil compreensão.

3. Construindo para Verificação

- Proteger o programa/classe de entradas inválidas por meio de verificações:
  - Dados de fontes externas;
  - Parâmetros de entrada da rotina (função, procedimento e método);
  - Tratamento de entradas incorretas.

3. Construindo para Verificação

Exemplo em Visual Basic do uso de assertivas para documentar pré-condições e pós-condições

```
Private Function Velocity ( _
   ByRef latitude As Single, _
   ByRef longitude As Single, _
   ByRef elevation As Single _
   ) As Single

   ' Pré-condições
   Debug.Assert ( -90 <= latitude And latitude <= 90 )
   Debug.Assert ( 0 <= longitude And longitude < 360 )
   Debug.Assert ( -500 <= elevation And elevation <= 75000 )
   ...

   ' Desinfeta os dados de entrada. Os valores devem estar dentro dos inter-
   ' valos asseverados acima, mas se um valor não estiver dentrode seu
   ' intervalo válido, ele será alterado para o valor válido mais próximo
   If ( latitude < -90 ) Then
      latitude = -90
   ElseIf ( latitude > 90 ) Then
      latitude = 90
   End If
   If ( longitude < 0 ) Then
      longitude = 0
   ElseIf ( longitude > 360 ) Then
```

Aqui está o código da assertiva.

Aqui está o código que trata de dados de entrada inválido em tempo de execução.

4. Reuso

- Utilizando de ativos existentes para resolver problemas diferentes;
- Ativos (software): bibliotecas, módulos, componentes, código fonte e ativos comerciais (*commercial off-the-shelf*);
- Benefícios: melhor se praticado sistematicamente com o aumento da produtividade, qualidade e custo;
- Construção para reuso: criar ativos de software reutilizáveis;
- Construção com reuso: criar novas soluções com os ativos pré-existentes;
- Frequentemente transcende as fronteiras do projeto, possibilitando atender a outras projetos e organizações.

5. Padrões de Construção

- O uso de padrões internos e externos durante a construção auxilia no alcance dos objetivos do projeto para eficiência, qualidade e custo.
- Alguns padrões que afetam diretamente as questões relativas a construção:
  - Métodos de comunicação interna ao projeto: padrões para formato e conteúdo de documentos;
  - Linguagens padrão de programação como Java, C++, .Net, etc;
  - Padrões de codificação: convenção de nomes, leiaute e indentação;
  - Plataformas: padrão de interfaces para chamadas ao sistema operacional, etc;
  - Ferramentas: padrões diagramáticos para notações como UML (*Unified Modeling Language*).

5. Padrões de Construção

- **Padrões internos:**
  - Padrões definidos pela organização a serem utilizados em nível corporativo ou em projetos específicos;
  - Auxiliam na coordenações de grupo de atividades, minimização da complexidade, antecipação de mudanças e construção para verificação.
- **Padrões externos:**
  - Uso em linguagens de programação, ferramentas de construção, interfaces técnicas e interações entre as áreas de conhecimento;
  - Oriundos de diversas fontes, incluem especificações de interface de hardware e software (*Object Management Group* - OMG) e organismos internacionais (*Institute of Electrical and Electronics Engineers* – IEEE ou *International Organization for Standardization* - ISO).

Estratégias para Construção

Sequencial

Iterativa

Estratégias para Construção Sequencial

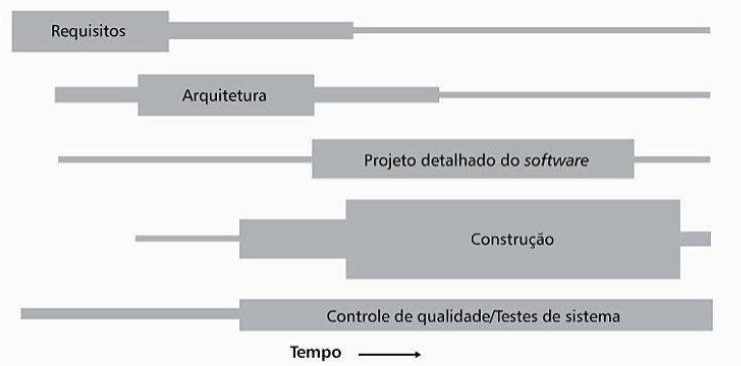


Figura 3-2 As atividades coincidirão até certo ponto na maioria dos projetos, mesmo naqueles que são altamente seqüenciais.

Estratégias para Construção Sequencial

- Os requisitos são bastante estáveis.
- O projeto do software é simples e está bem assimilado.
- A equipe envolvida no desenvolvimento está familiarizada com a área de aplicação.
- O projeto apresenta pouco risco.
- A previsibilidade a longo prazo é importante.
- O custo da mudança posterior dos requisitos, do projeto do software e do código provavelmente será alto.

Estratégias para Construção Iterativa

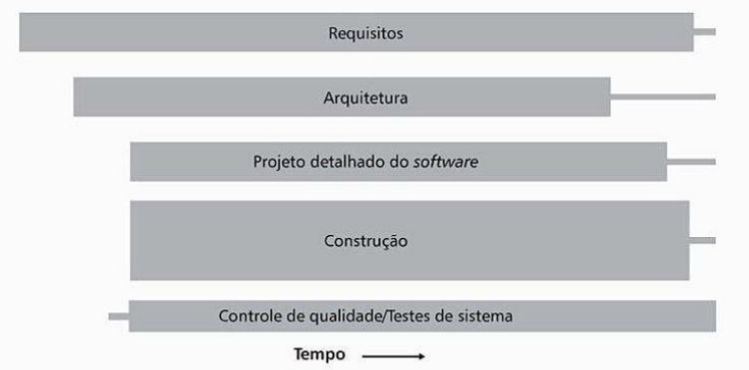


Figura 3-3 Em outros projetos, as atividades coincidirão durante sua construção. Um segredo da construção bem-sucedida é entender até que ponto os pré-requisitos foram concluídos e ajustar a estratégia adequadamente.

Estratégias para Construção Iterativa

- Os requisitos não estão bem entendidos ou você espera que eles sejam instáveis por outros motivos.
- O projeto do software é complexo, desafiador ou ambos.
- A equipe de desenvolvimento não está familiarizada com a área de aplicação.
- O projeto apresenta muito risco.
- A previsibilidade a longo prazo não é importante.
- O custo de mudança posterior dos requisitos, do *design* (projeto/desenho) e do código provavelmente será baixo.

Comparação entre as estratégias: Sequencial x Iterativo

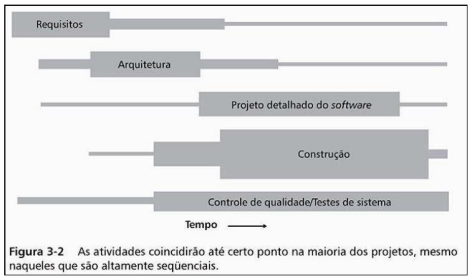


Figura 3-2 As atividades coincidirão até certo ponto na maioria dos projetos, mesmo naqueles que são altamente seqüenciais.

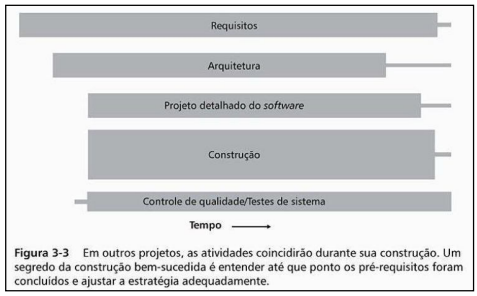


Figura 3-3 Em outros projetos, as atividades coincidirão durante sua construção. Um segredo da construção bem-sucedida é entender até que ponto os pré-requisitos foram concluídos e ajustar a estratégia adequadamente.

27

Fundamentos da Construção de Software

Leitura extra aula:

1. SWEBok - chapter 3 - Software Construction
   - 1. Software Construction Fundamentals – pág. 3-1 a 3-4

2. Code Complete – Um guia prático para a construção de software, 2ª ed., Steven McConnell, Bookman
   - Cap. 2 – Metáforas para um melhor entendimento do desenvolvimento de software – pág. 47 a 58
   - Cap. 3 – Meça duas vezes, corte uma: determinando os pré-requisitos – pág. 59 a 91
   - Cap. 4 – Principais decisões de construção – pág. 95 a 102