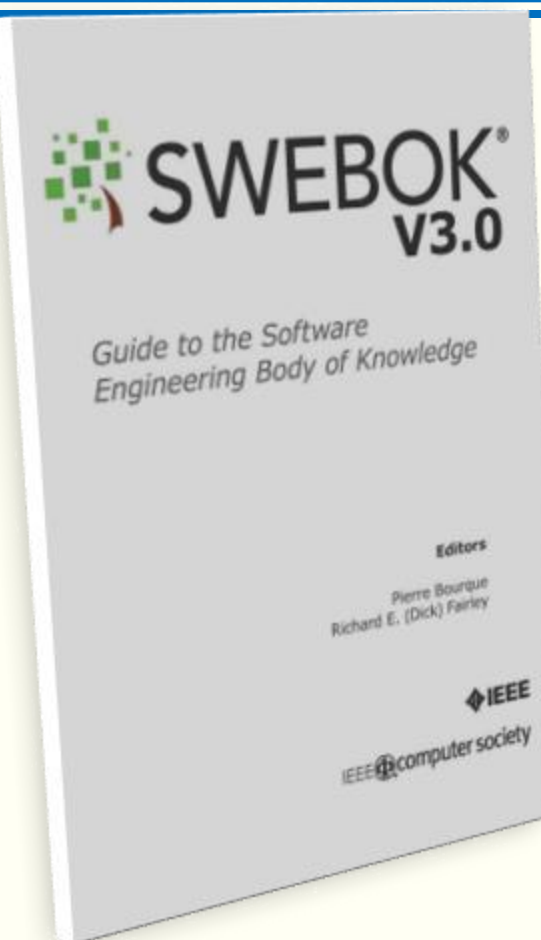


# 3

# CONSTRUÇÃO DE SOFTWARE

Mariana Cossetti Dalfior  
20211100064@pq.uenf.br



# Chapter 3: Software Construction

---

- **1. Software Construction Fundamentals**

- **1.1. Minimizing Complexity**
- **1.2. Anticipating Change**
- **1.3. Constructing for Verification**
- **1.4. Reuse**
- **1.5. Standards in Construction**

- **2. Managing Construction**

- **2.1. Construction in Life Cycle Models**
- **2.2. Construction Planning**
- **2.3. Construction Measurement**

- **3. Practical Considerations**

- **3.1. Construction Design**
- **3.2. Construction Languages**
- **3.3. Coding**
- **3.4. Construction Testing**
- **3.5. Construction for Reuse**
- **3.6. Construction with Reuse**
- **3.7. Construction Quality**
- **3.8. Integration**

- **4. Construction Technologies**

- **4.1. API Design and Use**
- **4.2. Object-Oriented Runtime Issues**
- **4.3. Parameterization and Generics**

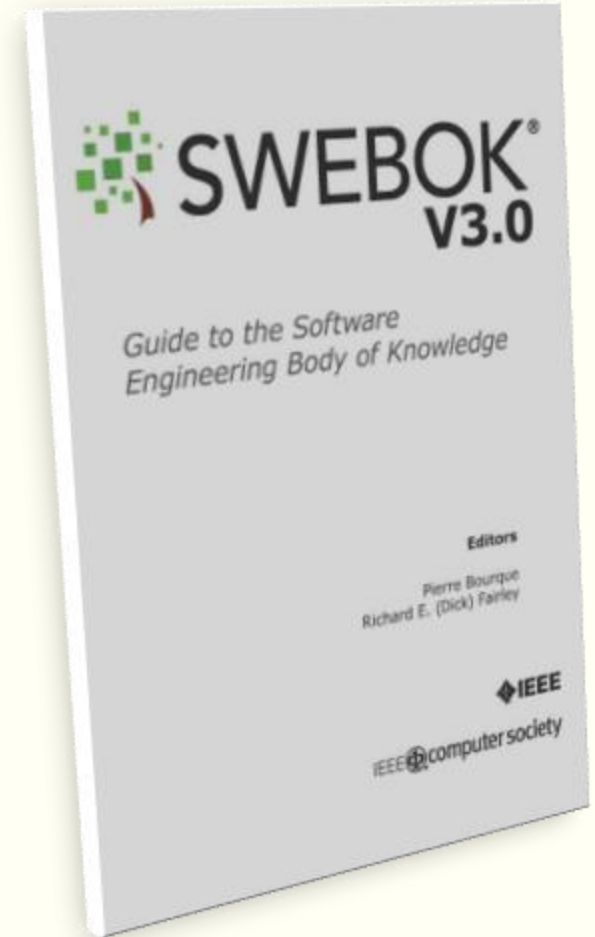
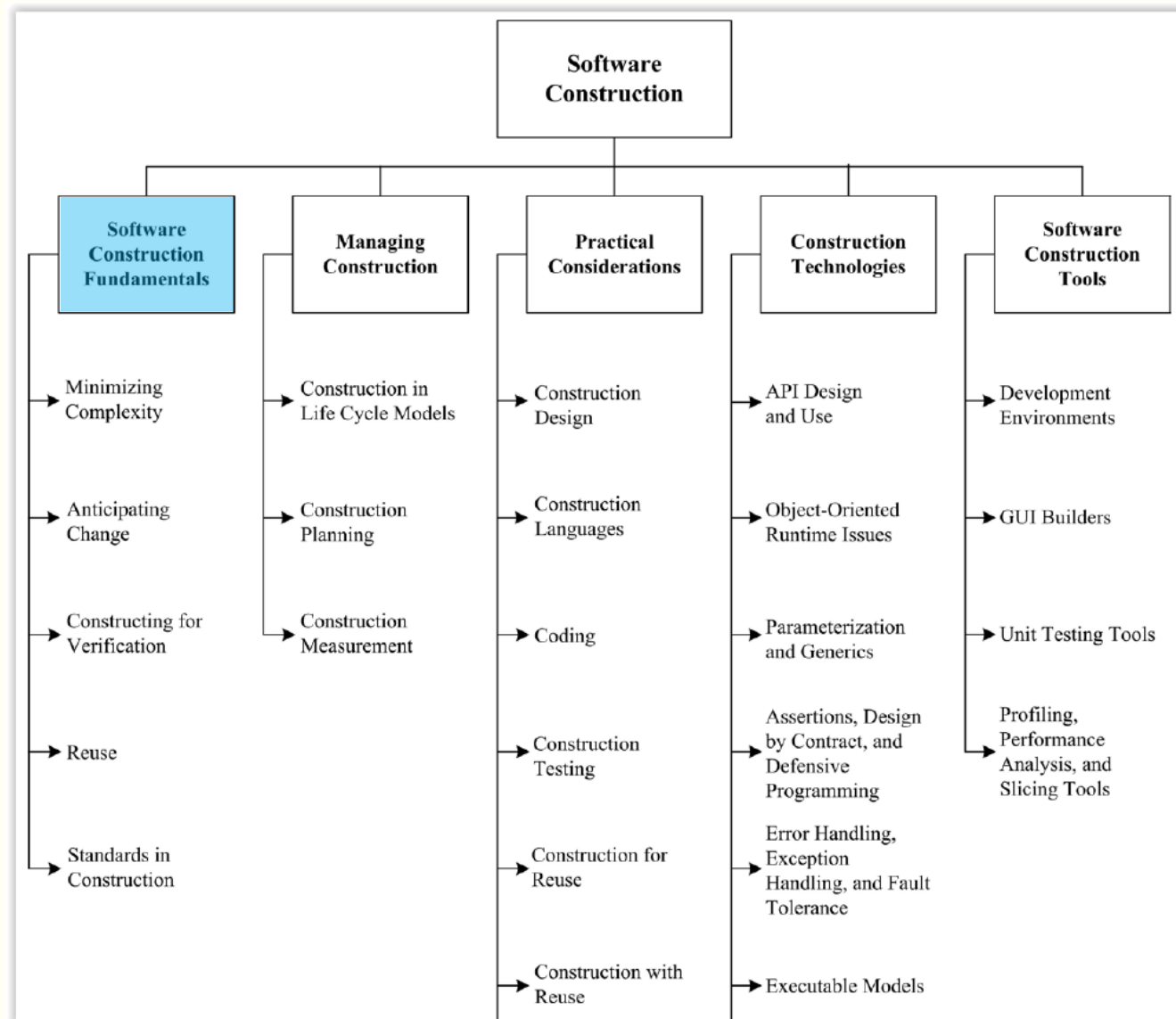
- **4. Construction Technologies**

- **4.4. Assertions, Design by Contract, and Defensive Programming**
- **4.5. Error Handling, Exception Handling, and Fault Tolerance**
- **4.6. Executable Models**
- **4.7. State-Based and Table-Driven Construction Techniques**
- **4.8. Runtime Configuration and Internationalization**
- **4.9. Grammar-Based Input Processing**
- **4.10. Concurrency Primitives**
- **4.11. Middleware**
- **4.12. Construction Methods for Distributed Software**
- **4.13. Constructing Heterogeneous Systems**
- **4.14. Performance Analysis and Tuning**
- **4.15. Platform Standards**
- **4.16. Test-First Programming**

- **5. Software Construction Tools**

- **5.1. Development Environments**
- **5.2. GUI Builders**
- **5.3. Unit Testing Tools**
- **5.4. Profiling, Performance Analysis, and Slicing Tools**

# Chapter 3: Software Construction



# 3.1 Fundamentos de Construção de Software

- **3.1.1 Minimizando Complexidade**
- **3.1.2 Antecipando Mudança**
- **3.1.3 Construção por Verificação**
- **3.1.4 Reuso**
- **3.1.5 Padrões em Construção de Software**

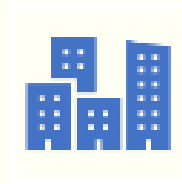


## 3.1.1 Minimizando Complexidade

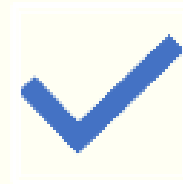
---



Limitações da  
capacidade humana



Forte controle na  
construção de  
software



Simplicidade e  
legibilidade do  
código



Utilização de padrões  
e técnicas específicas

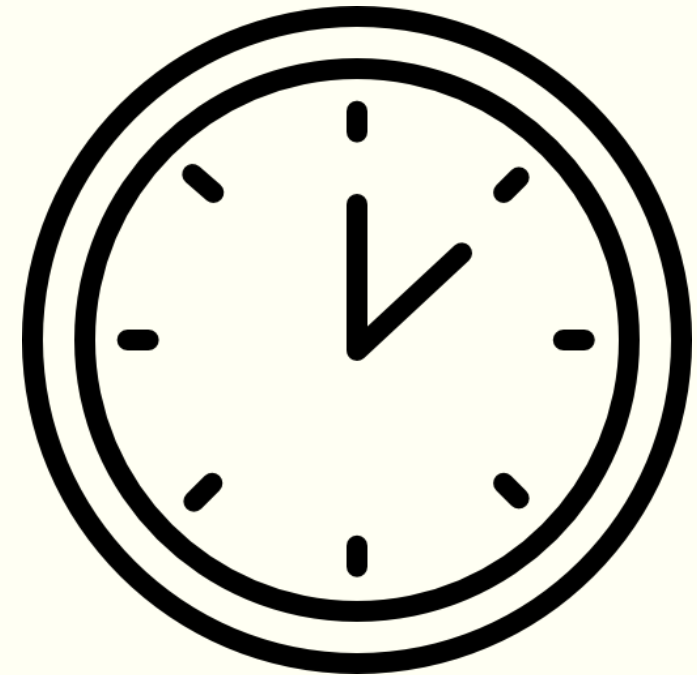


Foco na qualidade de  
construção

## 3.1.2 Antecipando Mudanças

---

- **Mudança inevitável**
- **Antecipação da mudança**
- **Impacto das mudanças externas**
- **Técnicas de antecipação de mudança**



## 3.1.3 Construção por Verificação

---

- **Importância da construção por verificação**
  - **Garante que as falhas sejam identificadas prontamente**
  - **Falhas durante o desenvolvimento, testes independentes e operações**
- **Técnicas específicas de suporte**
  - **Padrões de codificação**
  - **Teste de unidades**
  - **Organização de código**
  - **Restrição do uso de estruturas complexas**
- **Padrões de codificação**
  - **Escrever código consistente**
  - **Facilita a revisão de código**



## 3.1.3 Construção por Verificação

---

- **Teste de unidades**
  - **Verifica o funcionamento de partes individuais**
  - **Detecção precoce de falhas**
- **Organização de código**
  - **Facilita a automação de testes**
  - **Identificar falhas de forma automatizada e rápida**
- **Restrição do uso de estruturas complexas**
  - **Simplifica a detecção de falhas**
  - **Torna o código mais legível e manutenível**





## 3.1.4 Reuso

---

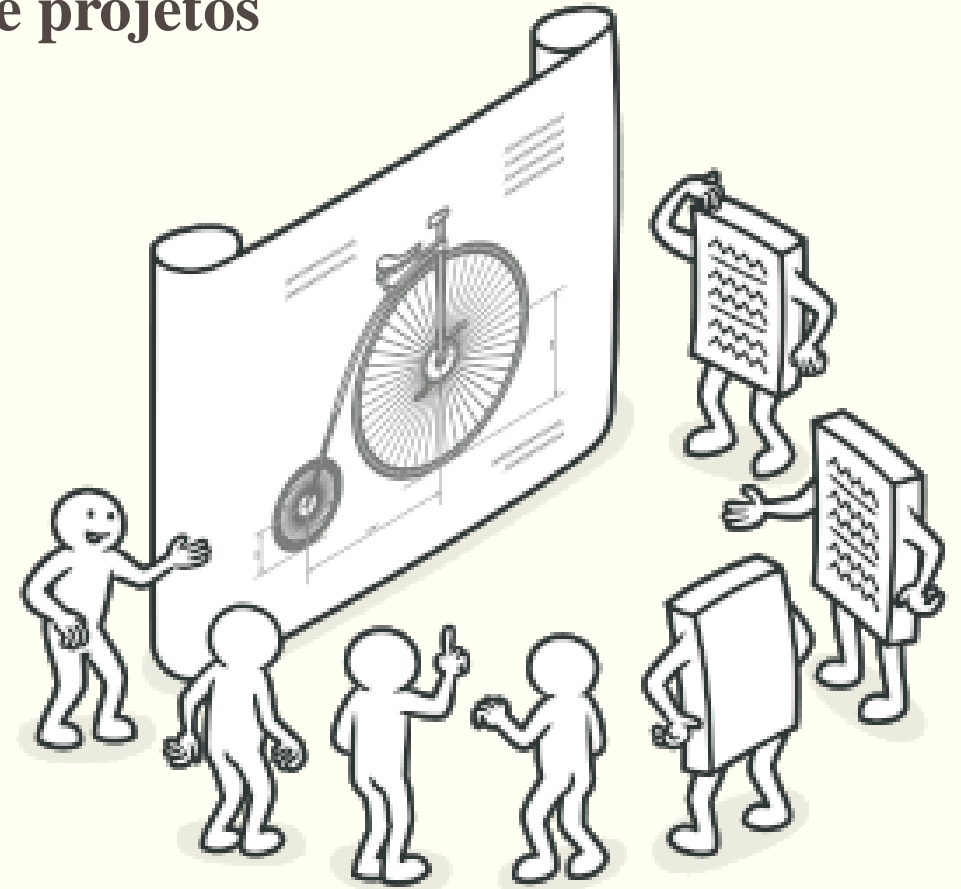
- O que é o reuso?
- Ativos reutilizados
  - Bibliotecas
  - Módulos
  - Componentes
  - Código fonte
  - Ativos comerciais prontos a utilizar
- Facetas da reutilização
  - Construção para Reutilização
  - Construção com Reutilização
- Transcendência dos projetos



## 3.1.5 Padrões em Construção de Software

---

- **Importância da aplicação no desenvolvimento de projetos**
  - **Eficiência**
  - **Qualidade**
  - **Custo**
- **Padrões que afetam diretamente a construção**
  - **Métodos de comunicação**
  - **Linguagens de programação**
  - **Normas de codificação**
  - **Plataformas**
  - **Ferramentas**

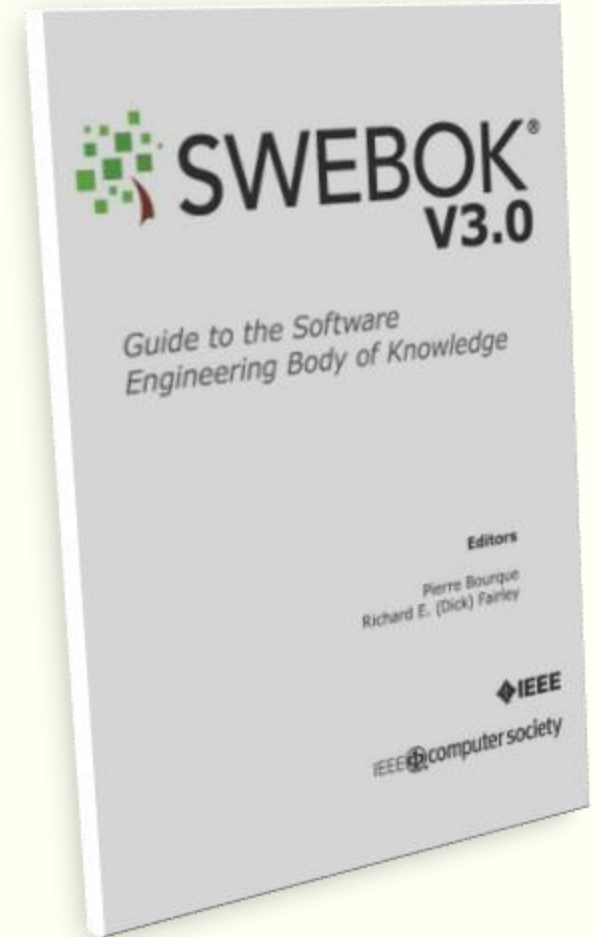
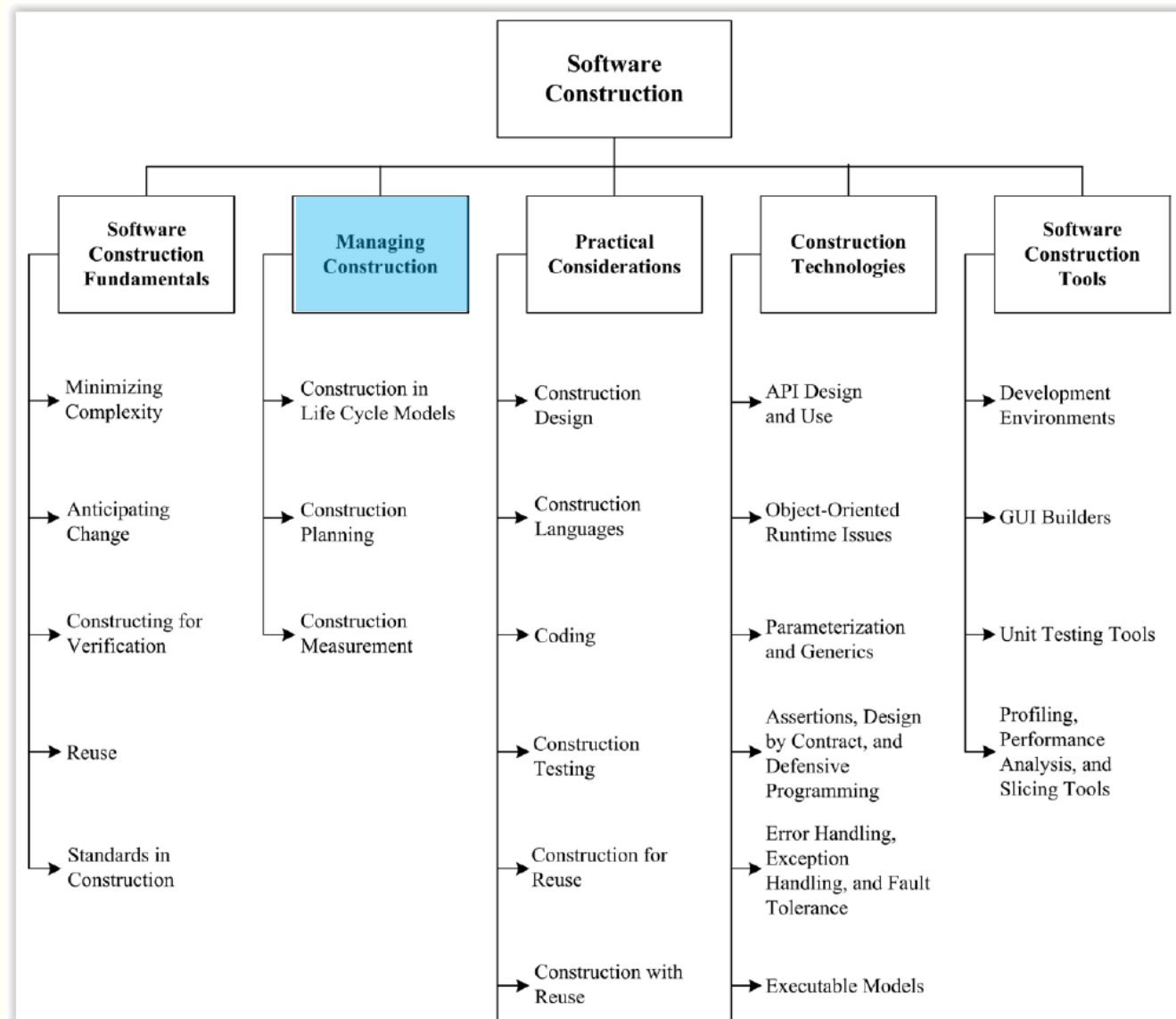


## 3.1.5 Padrões em Construção de Software

---

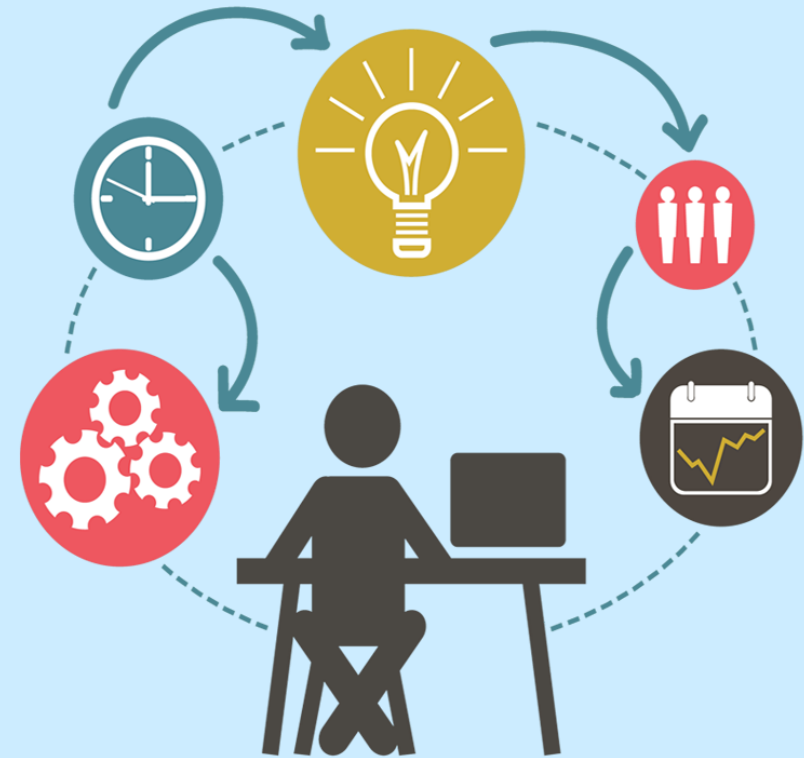
- **Utilização de padrões externos**
  - **Fontes de padrões externos**
  - **Exemplos de organizações que fornecem normas**
    - **OMG**
    - **IEEE**
    - **ISO**
- **Utilização de padrões internos**
  - **Vantagens das normas internas**
  - **Processo de criação de normas internas**
  - **Apoio à coordenação de atividades**
  - **Simplificação do desenvolvimento**

# Chapter 3: Software Construction



## 3.2 Gerenciamento da Construção

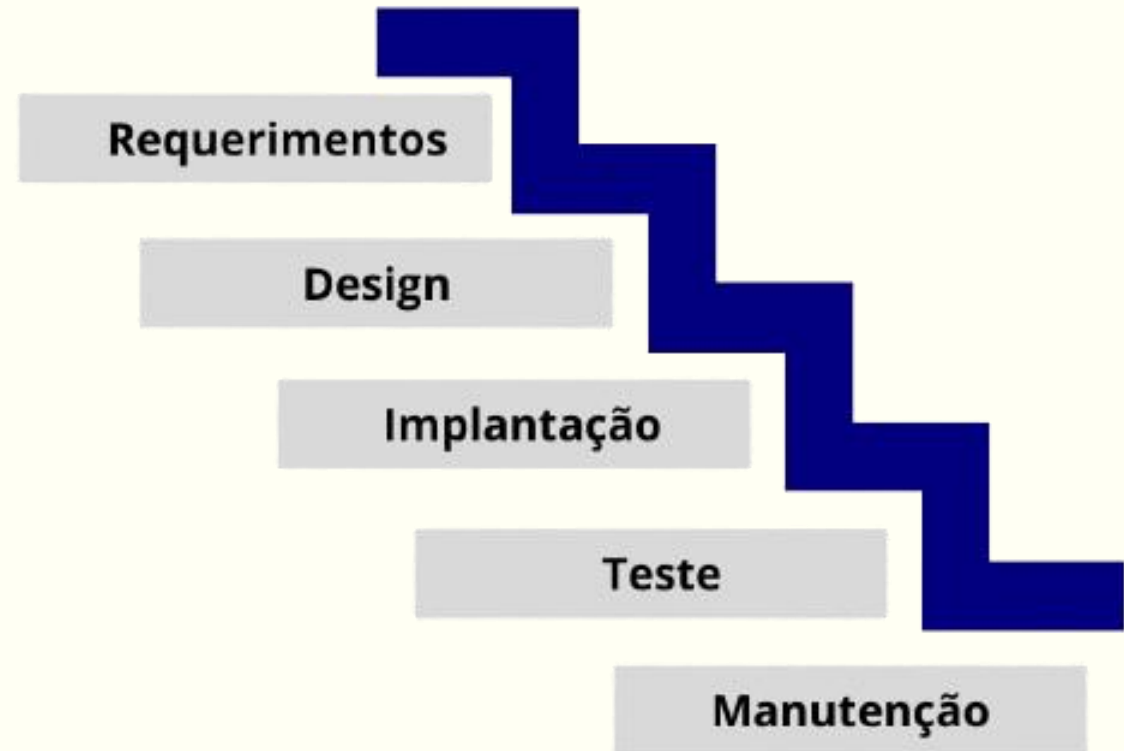
- 3.2.1 Construção nos Modelos de Ciclo de Vida
- 3.2.2 Planejamento da Construção
- 3.2.3 Medidas na Construção



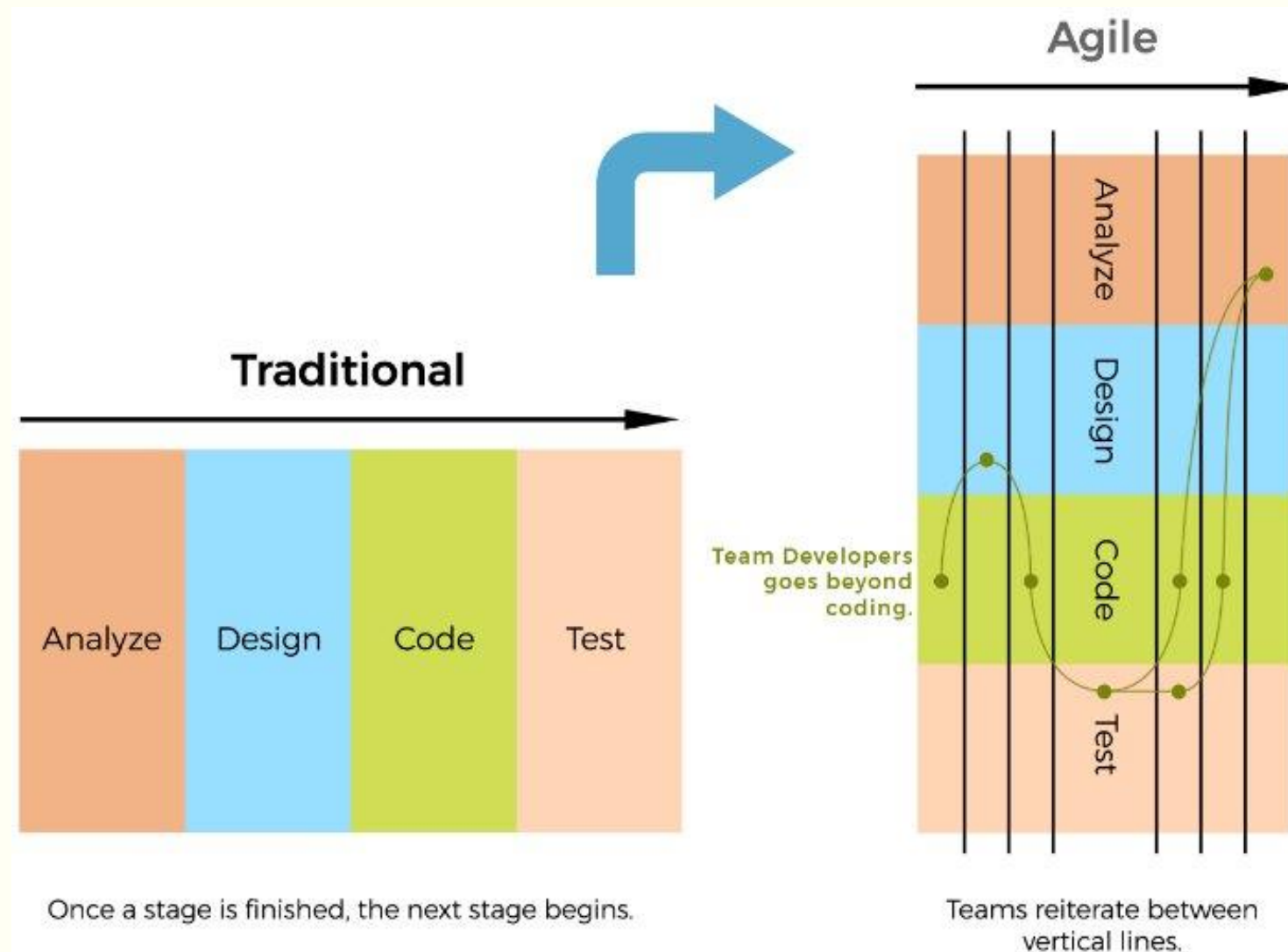
## 3.2.1 Construção nos Modelos de Ciclo de Vida

---

- **Modelo linear**
  - **Ciclo de vida cascata**
  - **Atividade sequencial**
  - **Foco na codificação**
  - **Separação entre atividades**
- **Modelos interativos**
  - **Atividade simultânea**
  - **Mistura:**
    - **Design**
    - **Codificação**
    - **Teste**



## 3.2.1 Construção nos Modelos de Ciclo de Vida



## 3.2.2 Planejamento da Construção

---

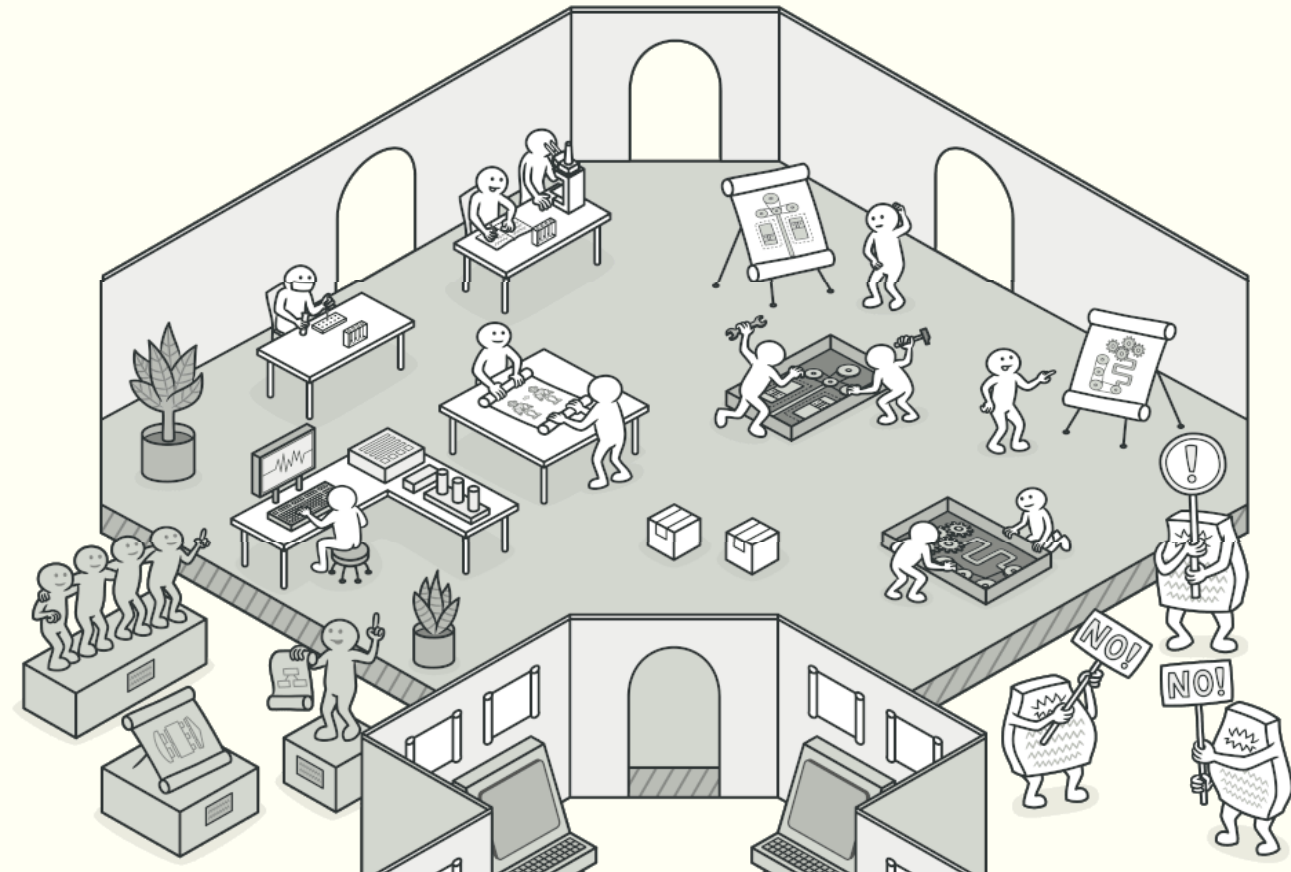
- **Métodos de construção**
  - **Afeta a realização dos pré-requisitos**
  - **Define a ordem e o momento da realização**
- **Abordagem de construção do projeto**
  - **Afeta a capacidade de reduzir complexidades**
  - **Influencia a antecipação de mudanças**
  - **Determina a verificação para construção**
  - **Influenciado pelos métodos de construção**



## 3.2.2 Planejamento da Construção

### ▪ Define:

- **Ordem de criação e integração de componentes**
- **Estratégia de integração**
  - **Integração por fases**
  - **Integração incremental**
- **Gerenciamento de qualidade**
- **Distribuição de tarefas específicas**



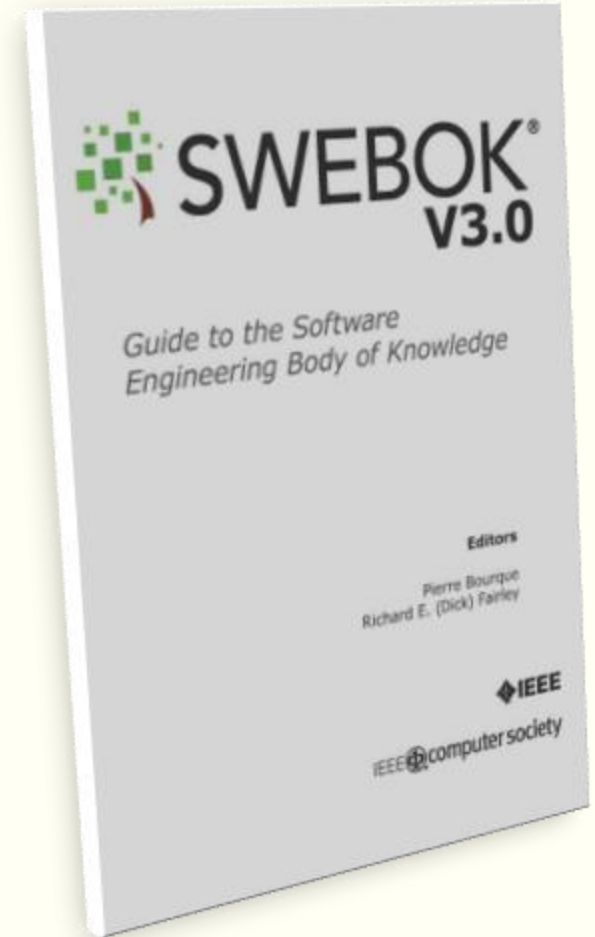
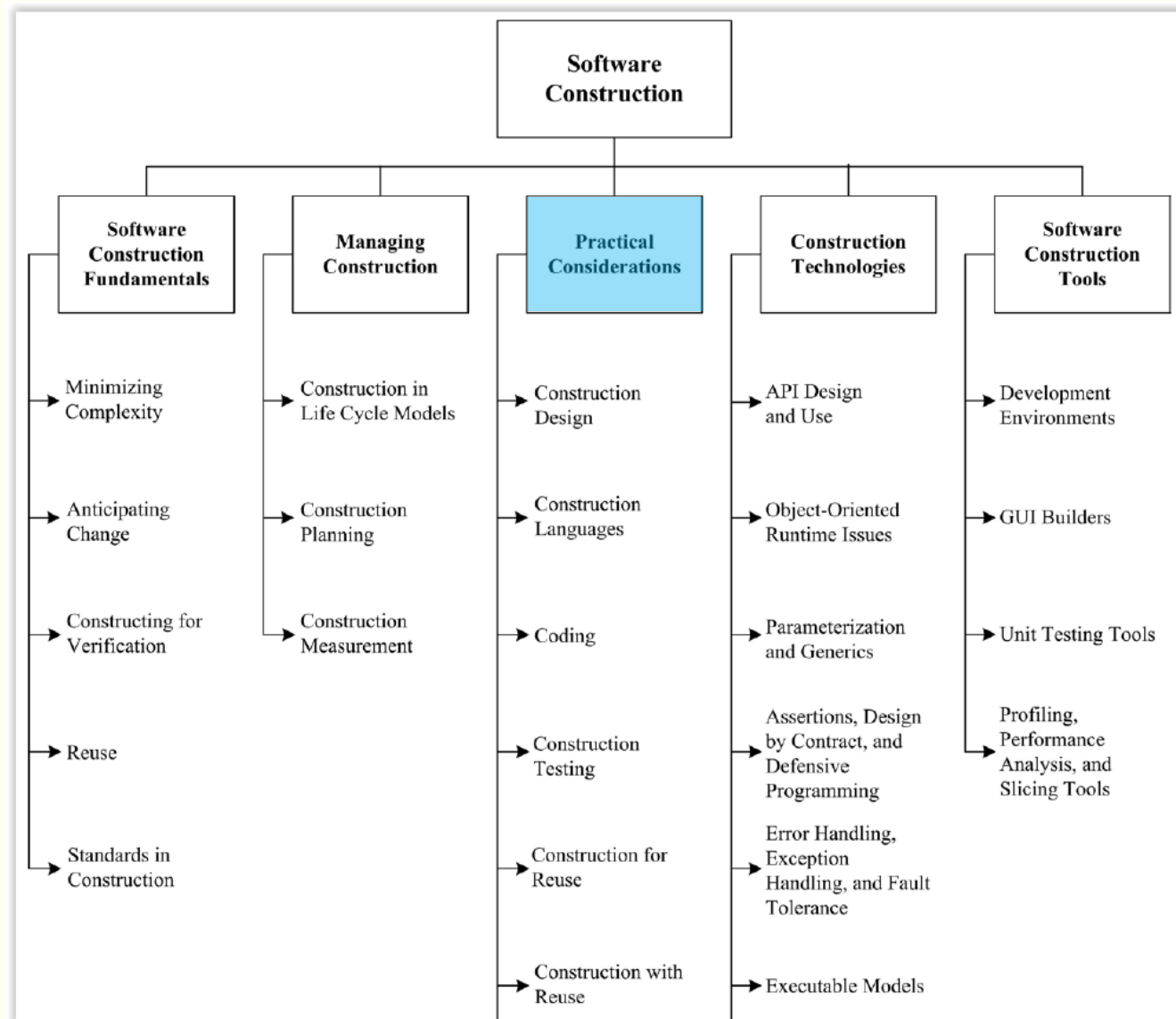
## 3.2.3 Medidas na Construção

---

- O que pode ser medido?
  - Desenvolvimento e modificação de código
  - Reutilização e destruição
  - Complexidade e inspeção estatística
  - Identificação e correção de falhas
  - Esforço e agendamento
- Úteis para:
  - Gerenciamento da construção
  - Assegurar qualidade durante a construção
  - Melhorar o processo de construção



# Chapter 3: Software Construction



## 3.3 Considerações Práticas da Construção

---

- **3.3.1 Projeto da Construção**
- **3.3.2 Linguagens de Construção**
- **3.3.3 Codificação**
- **3.3.4 Testando a Construção**
- **3.3.5 Construção por Reuso**
- **3.3.6 Construção com Reuso**
- **3.3.7 Qualidade da Construção**
- **3.3.8 Integração**

## 3.3.1 Projeto da Construção

---

- **Desenvolvimento em escala real**
  - **Restrições do mundo real**
  - **Adaptação contínua**
- **Física X Software**
  - **Modificações necessárias**
  - **Contabilização de falhas**
- **Detalhes do design em escala menor**
  - **Continuidade do processo**
  - **Ajustes graduais**



## 3.3.2 Linguagens de Construção

---

- Linguagens de configuração
  - **Opções predefinidas**
  - **Exemplos**
    - Arquivos de configuração baseados em texto
      - Windows
      - UNIX
    - Listas de seleção de estilos menu
- Kit de ferramentas de linguagem
  - **Criar aplicativos**
  - **Linguagens de programação de aplicativos**



## 3.3.2 Linguagens de Construção

---

- **Linguagens de programação**
  - **Mais flexível de linguagem de construção**
  - **Formação e habilidade para utilizá-las de forma eficaz**
  - **Três tipos: linguística, formal e visual**
- **Linguística**
  - **Cadeias de palavras-chave**
  - **Sintaxe de frase semelhante para compreensão intuitiva**
  - **Forte conotação semântica**
- **Visual**
  - **Interpretação visual direta**
  - **Posicionamento das entidades visual**

## 3.3.2 Linguagens de Construção

---

- **Visual**
  - Limitada
  - Ferramenta poderosa
- **Formal**
  - Menos significados intuitivos
  - Mais definições precisas e sem ambiguidades
  - Formas de sistema de programação
    - Precisão
    - Comportamento de tempo
    - Testabilidade
  - Formas definidas de combinar os símbolos



## 3.3.3 Codificação

---

- **Compreensão do código fonte**
  - **Convenções de nomenclatura**
  - **Layout do código fonte**
- **Uso eficaz de entidades**
  - **Classes**
  - **Tipos enumerados**
  - **Variáveis e constantes nomeadas**
- **Estruturas de controle**
- **Tratamento de Erros**
  - **Condições antecipadas**
  - **Condições excepcionais**



### 3.3.3 Codificação

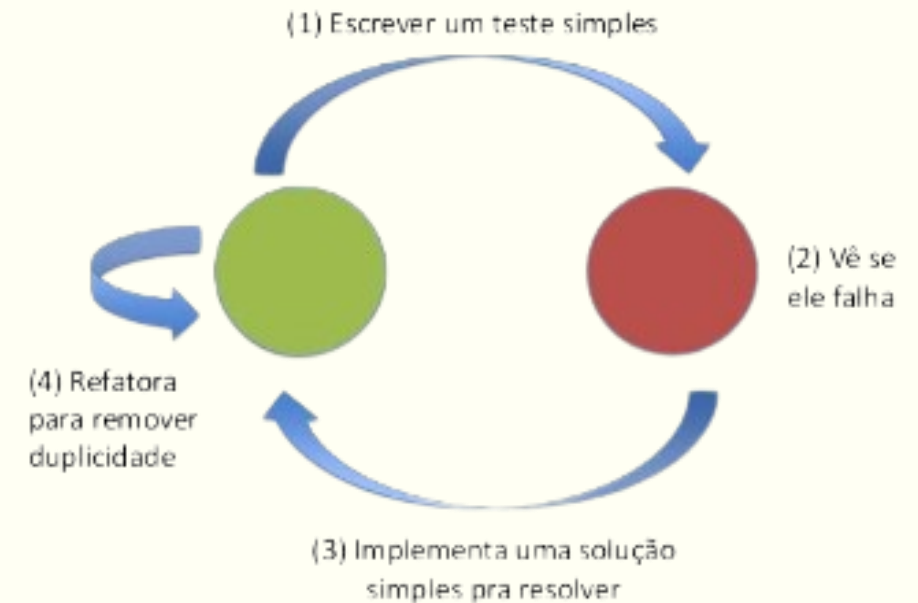
---

- **Medidas de segurança**
  - **Prevenção de violações de segurança**
  - **Estouro de buffer**
  - **Limites de índice de array**
- **Gestão de recursos**
  - **Mecanismos de exclusão**
  - **Disciplina no acesso a recursos**
  - **Manipulação de recursos reutilizáveis em série**
- **Organização do código fonte**
  - **Declarações**
  - **Rotinas**
  - **Classes**
  - **Pacotes**
- **Documentação**
- **Otimização**
  - **Ajuste de código**
  - **Melhoria de desempenho**

## 3.3.4 Testando a Construção

- Formas de teste
  - Teste de unidade
  - Teste de integração
- Propósito dos testes
  - Após a escrita
  - Antes da escrita
- Subconjunto de testes
  - Não incluem testes especializados:
    - testes de sistema
    - teste de configuração
    - testes de usabilidade

### TDD – Test Driven Development



# Reuso e Construção de Software

---

## ▪ 3.3.5 Construção por Reuso

- **Potencial para reutilização futura**
- **Baseada em análise de variabilidade e design**
- **Tarefas**
  - Implementação de variabilidade
  - Encapsulação de variabilidade
  - Testar a variabilidade
  - Descrição e publicação de ativos

## • 3.3.6 Construção com Reuso

- **Criar novo software com a reutilização**
- **Tarefas**
  - A seleção das unidades reutilizáveis
  - A avaliação da reutilização
  - A integração no software atual
  - A geração de relatórios



## 3.3.7 Qualidade da Construção

---

- Vulnerabilidades de segurança
- Técnicas para garantir a qualidade
  - Testes de unidade e teste de integração
  - Desenvolvimento orientado a teste
  - Uso de assertivas e programação defensiva
  - Debugging
  - Inspeções
  - Revisões técnicas
  - Análise estática
- Concentram-se no código

## 3.4.8 Integração

---

- **Preocupações da integração**
  - **Planejamento da sequência**
  - **Identificação dos requisitos**
  - **Criação de infraestrutura**
  - **Teste e qualidade dos componentes**
- **Abordagens de integração**
  - **Faseada**
  - **Incremental**



## 3.4.8 Integração

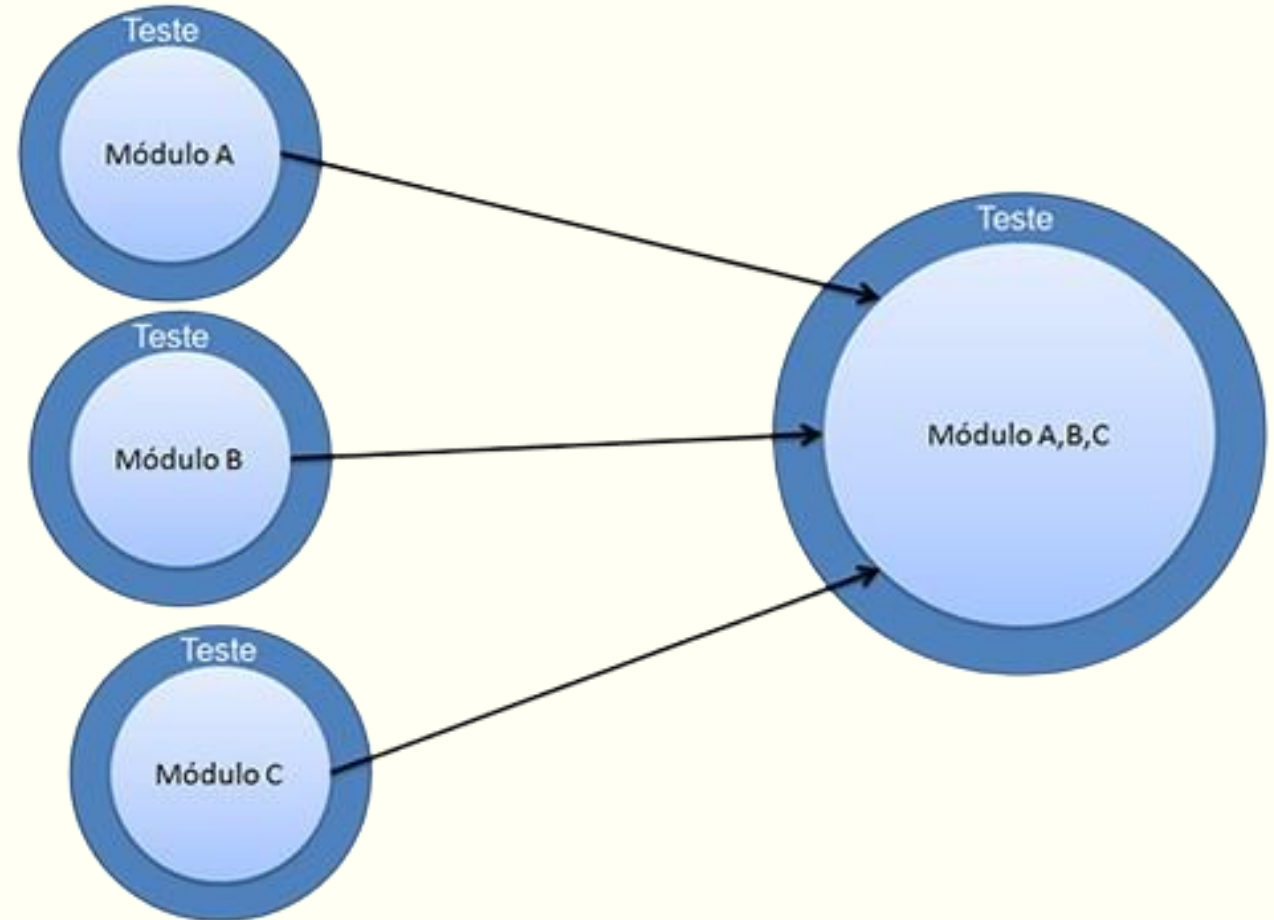
---

- Faseada

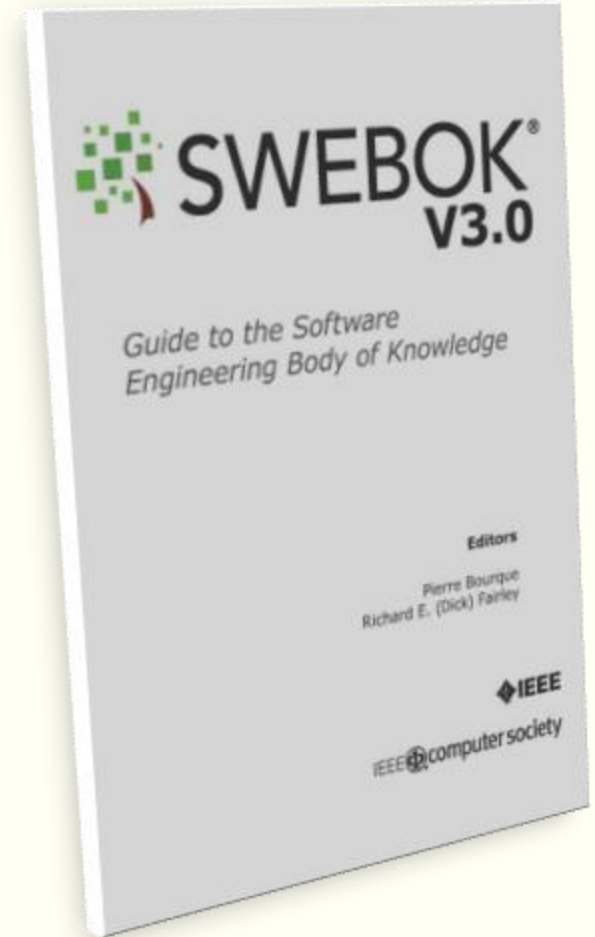
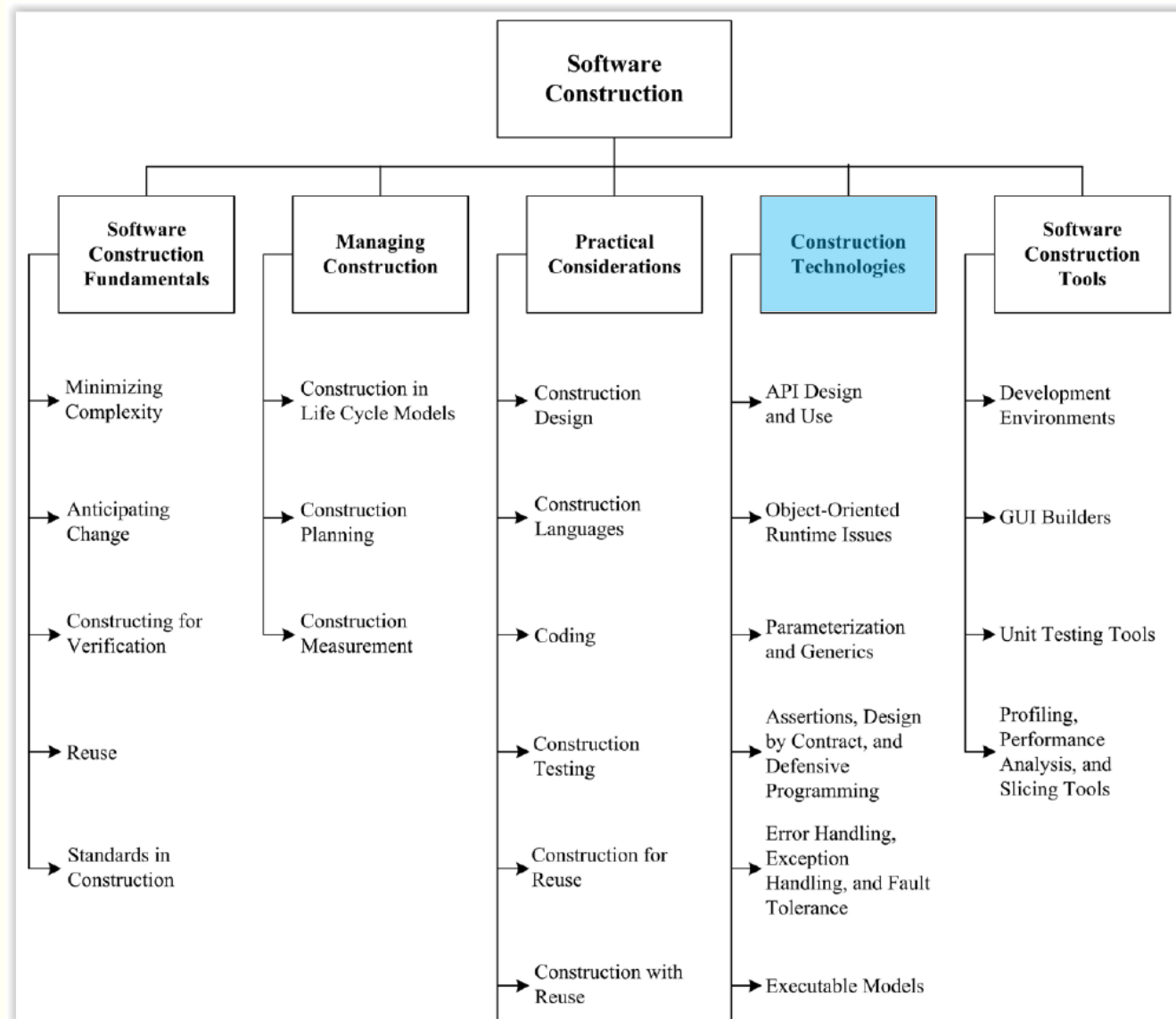
- **"Big Bang"**
- **Adiantamento das partes**

- Incremental

- **Escrever, testar e combinar**
- **Infraestrutura adicional**
- **Feedback precoce**
- **Testes e qualidade**
- **Localização mais fácil de erros**



# Chapter 3: Software Construction





## 3.4 Tecnologias de Construção

---

- **3.4.1 Projeto e Uso de APIs**
- **3.4.2. Mecanismos de Execução Orientado a Objetos**
- **3.4.3 Parametrização**
- **3.4.4 Assertivas, Desenho por contrato, e Programação defensiva**
- **3.4.5 Manipulação de Erro e Exceções, e Tolerância a Falhas**
- **3.4.6 Modelos executáveis**
- **3.4.7 Técnicas de Construção Baseadas em Estado e Orientadas por Tabela**
- **3.4.8 Configuração em Tempo de Execução e Internacionalização**
- **3.4.9 Processamento de Entrada Baseado em Gramática**

## 3.4 Tecnologias de Construção

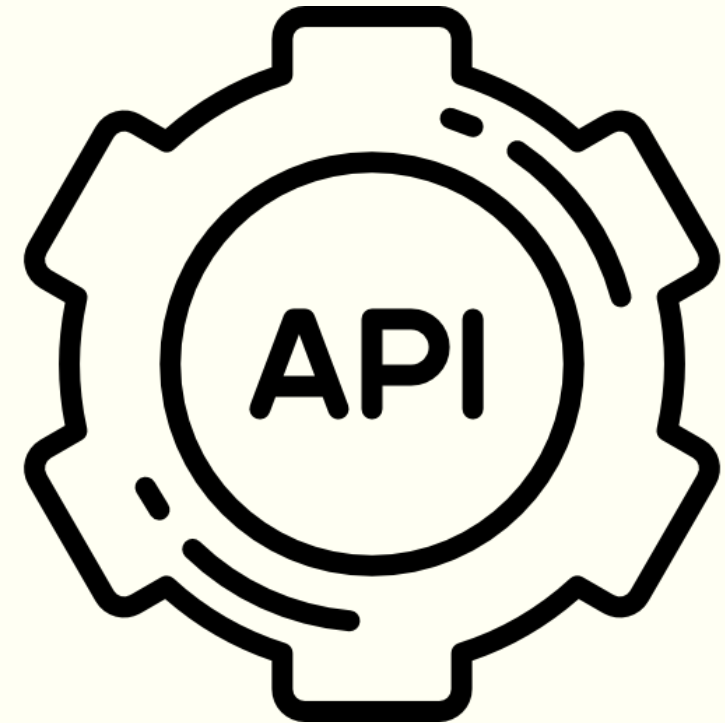
---

- **3.4.10 Primitivas de Concorrência**
- **3.4.11 Middleware**
- **3.4.12 Métodos de Construção para Software Distribuído**
- **3.4.13 Construindo Sistemas Heterogêneos**
- **3.4.14 Análise e Ajuste de Desempenho**
- **3.4.15 Padrões de Plataforma**
- **3.4.16 Programação Orientada a Testes**

## 3.4.1 Projeto e Uso de APIs

---

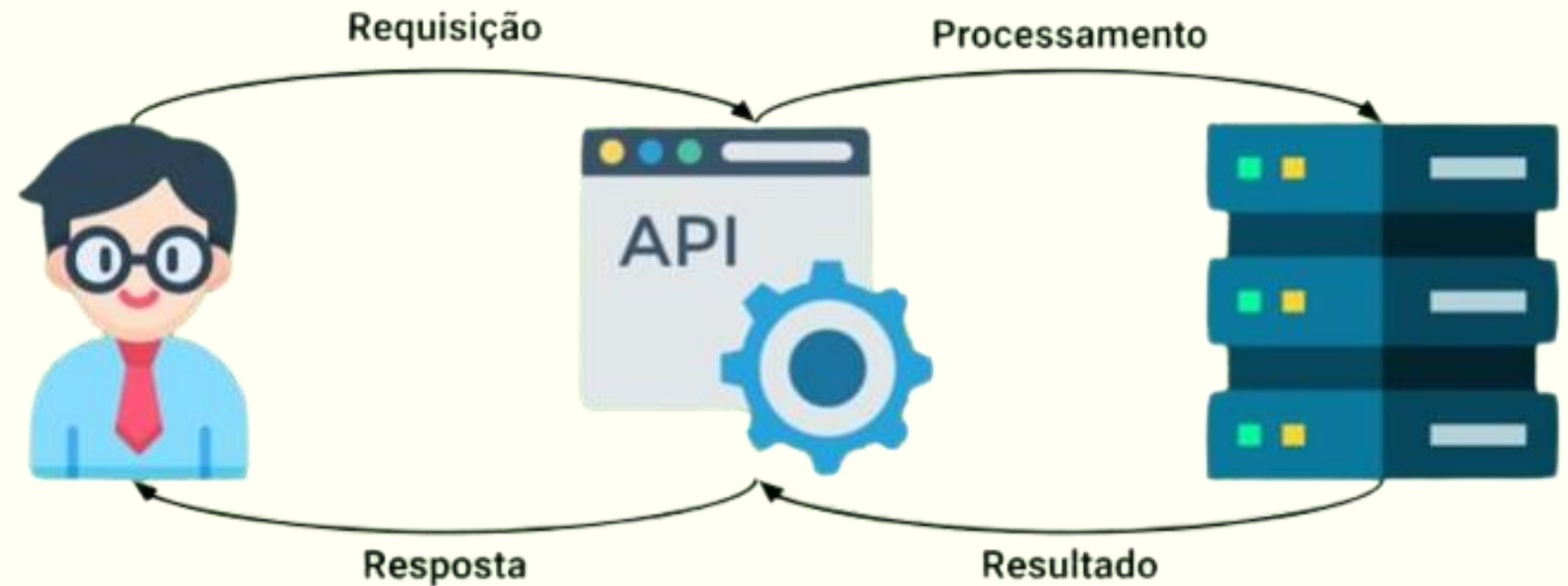
- **Princípios de design**
  - **Facilidade de aprendizado e memorização**
  - **Legibilidade do código**
  - **Prevenção de má utilização**
  - **Facilidade de extensão**
  - **Compatibilidade com versões anteriores**
- **API bem projetada**
  - **Facilita desenvolvimento e manutenção**
  - **Estabilidade ao longo do tempo**



## 3.4.1 Projeto e Uso de APIs

---

- **Uso da API**
  - **Seleção**
  - **Aprendizado**
  - **Teste**
  - **Integração**
  - **Extensão**



## 3.4.2. Mecanismos de Execução Orientado a Objetos

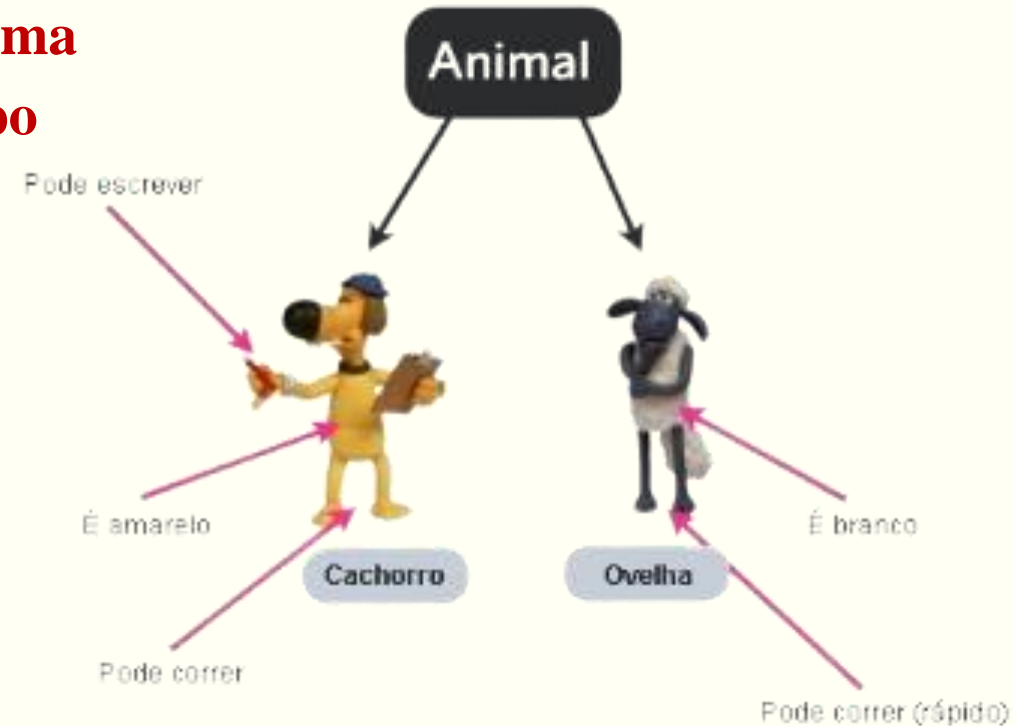
---

- **Polimorfismo**

- Suportar operações gerais
- Determinação do comportamento em tempo de execução

- **Reflexão**

- Observação e modificação da estrutura do programa
- Possibilidades de inspeção e instanciação em tempo de execução
- Permite
  - Instanciação de novos objetos
  - Invocação de métodos



## 3.4.3 Parametrização

---

- Genéricos
  - **Ada**
  - **Eiffel**
- Templates
  - **C++**
- Permite a definição de um tipo ou classe
- Tipos não especificados fornecidos como parâmetros
- Terceira maneira de compor comportamentos
  - **Complementa a herança de classe**
  - **Composição Software Orientado a Objetos**



## 3.4.4 Assertivas, Programação por contrato e defensiva

---

- **Assertivas**

- **Predicado executável**
  - **Permite verificação em tempo de execução**
- **Alta confiabilidade**
- **Identificação rápida de interfaces incompatíveis e erros**
- **Compiladas durante o desenvolvimento**
- **Removidas para não degradar desempenho**

- **Programação por contrato**

- **Inclusão de pré-condições e pós-condições em rotinas**
- **Fornece uma especificação precisa da semântica**
- **Melhora na qualidade**



## 3.4.4 Assertivas, Programação por contrato e defensiva

---

- Programação defensiva
  - Proteger rotina prejudicada por entrada inválida
  - Formas
    - Verificar valores de todos os parâmetros
    - Decidir como lidar com entradas ruins
  - Uso de assertivas para verificar valores





## 3.4.5 Manipulação de Erro e Exceções, e Tolerância a Falhas

---

- Correção, robustez e atributos não funcionais

- Tratamento de erros

- Retorno de valor neutro
- Substituição de dados inválidos
- Registro de mensagens de aviso
- Retorno de código de erro
- Encerramento do software

- Uso de exceções

- Detecção e processamento de erros ou eventos excepcionais.
- Estrutura básica: throw e try-catch

```
1  try {  
2      const isAuthenticated = login(email, senha)  
3  
4      if (!isAuthenticated) {  
5          throw new Error('E-mail ou senha inválidos')  
6      }  
7  } catch (err) {  
8      window.alert(err.message)  
9      // E-mail ou senha inválidos  
10 }
```

## 3.4.5 Manipulação de Erro e Exceções, e Tolerância a Falhas

---

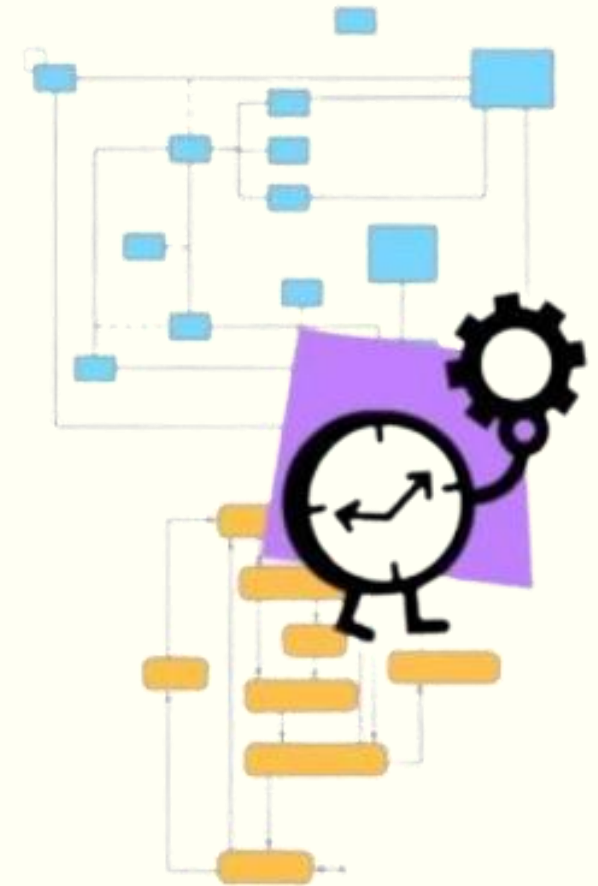
- **Uso de exceções**
  - **Políticas de tratamento:**
    - Incluir mensagem de exceção
    - Evitar blocos catch vazios
    - Conhecer as exceções lançadas pelo código da biblioteca
    - Construir um relator de exceções centralizado
    - Padronização do uso de exceções
- **Tolerância a falhas**
  - **Aumento da confiabilidade do software**
  - **Estratégias**
    - Backup e tentar novamente
    - Código auxiliar
    - Algoritmos de votação
    - Substituição por valor falso



## 3.4.6 Modelos executáveis

---

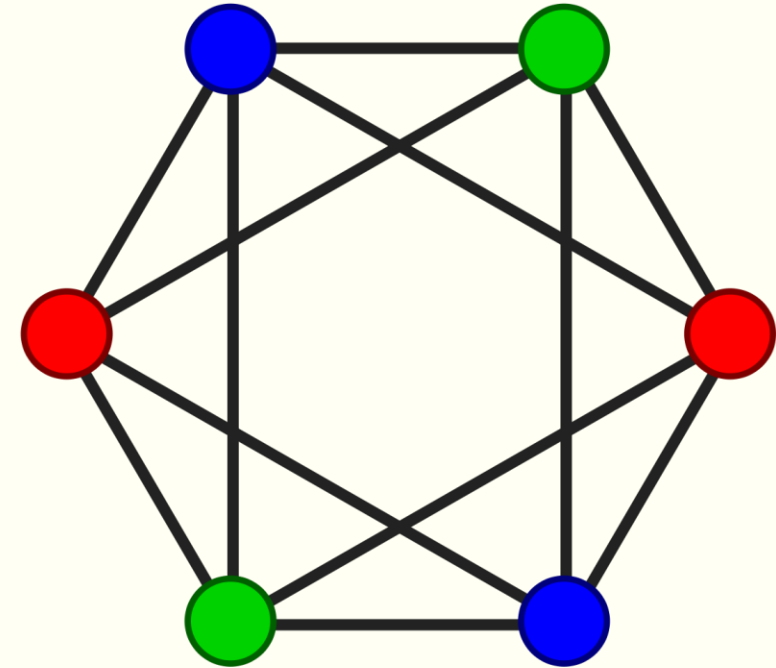
- **Benefícios**
  - **Abstração de detalhes**
  - **Flexibilidade na implantação**
- **Processo de compilação**
  - **xUML (UML executável)**
  - **Modelo executável em implementação**
  - **Constrói software executável**
    - **Arquitetura Orientada a Modelos**
    - **Especificar um Modelo Independente de Plataforma (PIM)**
      - **Não depende de nenhuma tecnologia**
    - **Modelo Específico de Plataforma (PSM)**
      - **Contém os detalhes da implementação**
      - **Produzido entrelaçando o PIM e a plataforma**



## 3.4.7 Técnicas de Construção Baseadas em Estado e Orientadas por Tabela

---


- Baseada em estados ou autômatos
  - Utiliza máquinas de estados finitos
  - Uso dos grafos de transição
  - Etapas
    - Especificação
    - Implementação
    - Depuração
    - Documentação
  - Combinada com POO
  - Programação orientada a objetos baseada em estados



## 3.4.7 Técnicas de Construção Baseadas em Estado e Orientadas por Tabela

---

- Método baseado em tabelas
  - **Buscar informações**
  - **Simplicidade em comparação com lógica complexa**
  - **Facilidade de modificação**
  - **Informações a serem armazenadas nas tabelas**
  - **Eficiência no acesso às informações**

Clientes		
	Nome da Coluna	Tipo de Dados
	Id	int
	Nome	nvarchar(M...
	Idade	int
	Credito	decimal(18, 2)

## 3.4.8 Configuração em Tempo de Execução e Internacionalização

---

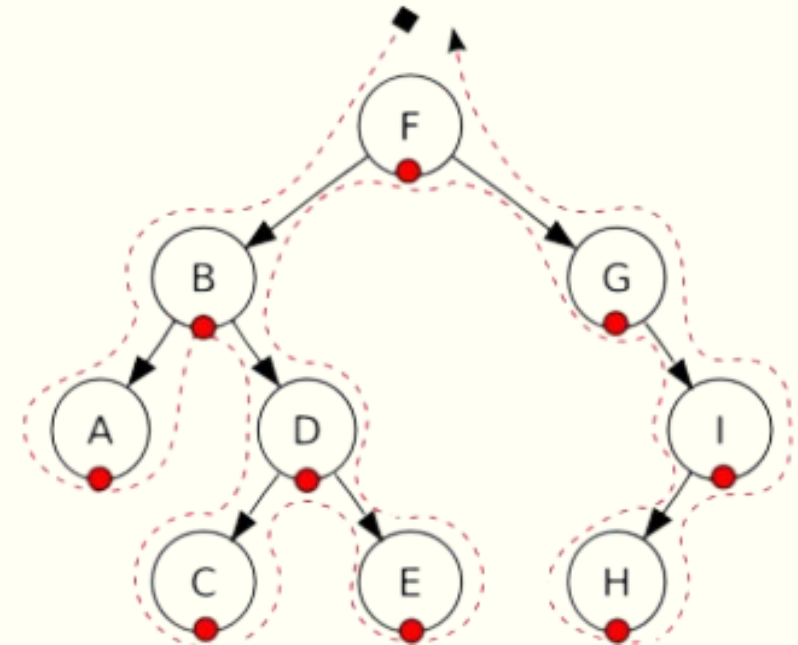
- **Configuração em tempo de execução**
  - **Vinculação de valores e configurações em execução**
  - **Modo just-in-time**
- **Internacionalização**
  - **Software interativo**
  - **Preparação para suportar múltiplos locais**
  - **Localização: compatibilidade com diferentes idiomas**
  - **Adaptação de prompts, mensagens de erro, etc.**
- **Processos de design e construção**
  - **Acomodar problemas de cadeia de caracteres**
  - **Tipos de strings usar**
  - **Minimização do impacto**



## 3.4.9 Processamento de Entrada Baseado em Gramática

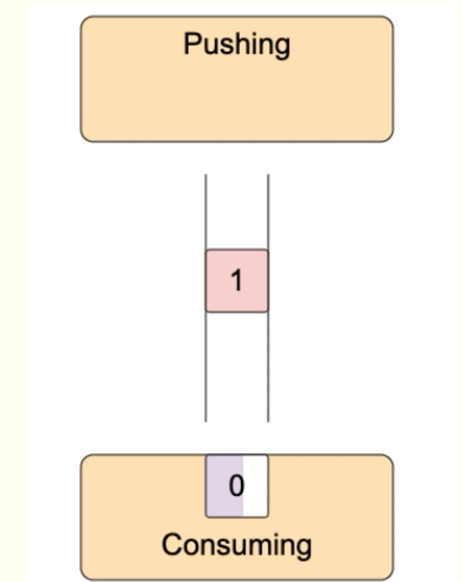
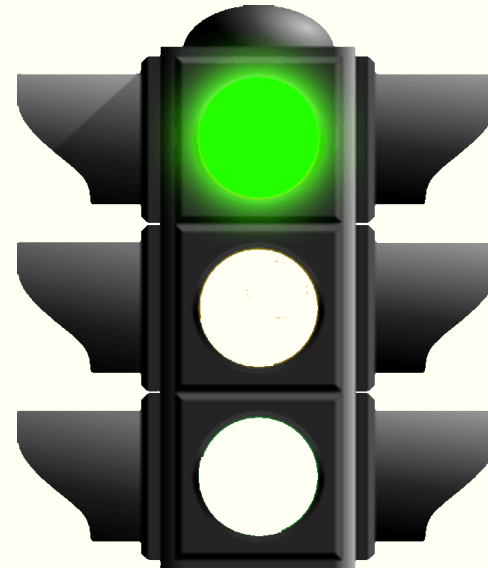
---

- O que é?
  - **Análise da sequência de tokens de entrada**
  - **Criação da árvore de análise**
- Construção e utilização da árvore de análise
  - **Travessia em ordem da árvore**
  - **Verificação da presença de variáveis**
  - **Utilizada como entrada para os processos**



## 3.4.10 Primitivas de Concorrência

- Abstração de programação
- Tipos de primitivos
  - **Semáforos**
    - Abstração simples
    - Controla o acesso
  - **Monitores**
    - Opera com exclusão mútua
    - Um processo por vez
  - **Exclusão Mútua**
    - Primitivo de sincronização
    - Concede acesso exclusivo

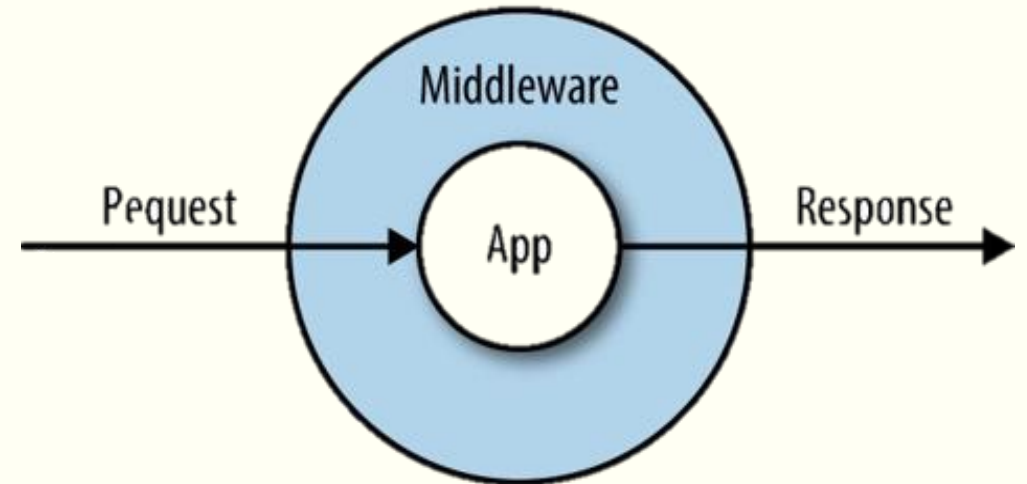




## 3.4.11 Middleware

---

- Entre o sistema operacional e o programa de aplicação
- Contêineres em tempo de execução
- Conector entre os componentes de software
- Orientado a mensagens
  - **Barramento de serviço empresarial (ESB)**
    - **Suporta interação e comunicação**



## 3.4.12 Métodos de Construção para Software Distribuído

---

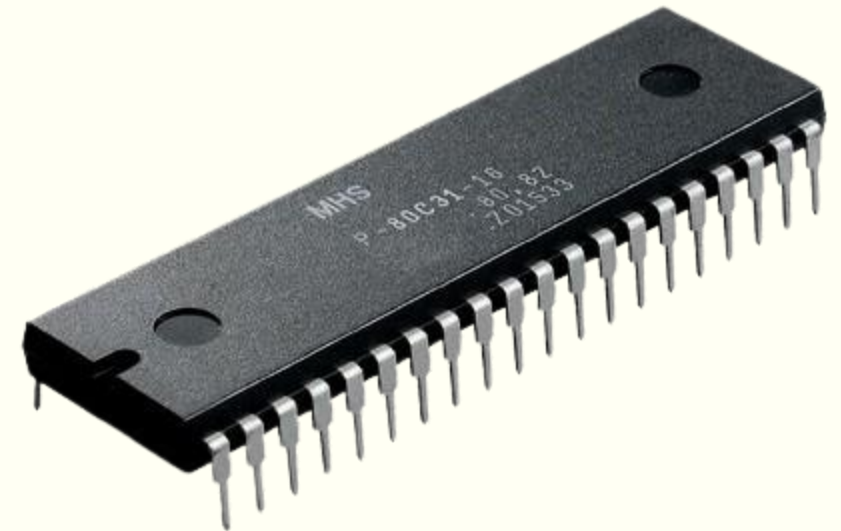
- Coleção de sistemas de computador
- Distribuído x Tradicional
  - **Paralelismo**
  - **Comunicação**
  - **Tolerância a falhas**
- Categorias arquiteturais
  - **Cliente-servidor**
  - **Arquitetura de 3 camadas**
  - **Arquitetura de n camadas**
  - **Objetos distribuídos**
  - **Acoplamento frouxo ou acoplamento rígido**



## 3.4.13 Construindo Sistemas Heterogêneos

---

- Variedade de unidades computacionais
- Independência e comunicação
- Design
  - **Combinação de várias linguagens**
  - **Codesign de hardware/software**
  - **Validação multilíngue**
  - **Co-simulação**
  - **Interfaceamento**



## 3.4.13 Construindo Sistemas Heterogêneos

---

- **Processo de co-design**
  - **Simultaneidade de software e hardware virtual**
  - **Decomposição passo a passo**
  - **Hardware**
    - **Simulação em FPGAs ou ASICs**
  - **Software**
    - **Tradução para linguagem de baixo nível**

## 3.4.14 Análise e Ajuste de Desempenho

---

- **Eficiência do código**
  - **Arquitetura**
  - **Decisões de design, estruturas de dados e algoritmos**
  - **Velocidade e tamanho**
- **Análise de desempenho**
  - **Investigação do comportamento do programa**
  - **Identifica pontos críticos**
- **Ajuste de código**
  - **Melhora o desempenho**
  - **Mudanças em pequena escala**
  - **Uso de Linguagem de baixo nível**



## 3.4.15 Padrões de Plataforma

---

- Conjunto de serviços e APIs padrão
- Plataforma compatíveis seguem esses padrões
- Exemplos
  - **Java 2 Platform Enterprise Edition (J2EE)**
  - **Padrão POSIX para Sistemas Operacionais**
- Benefícios
  - **Desenvolvimento de aplicações multiplataforma**
  - **Reduz ajustes para diferentes ambientes**
  - **Promovem a interoperabilidade**



# POSIX

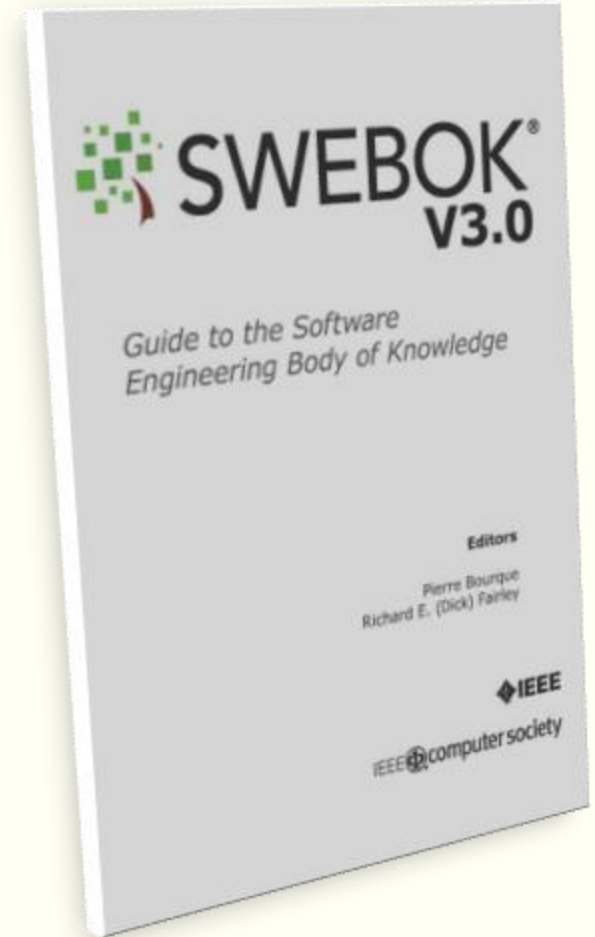
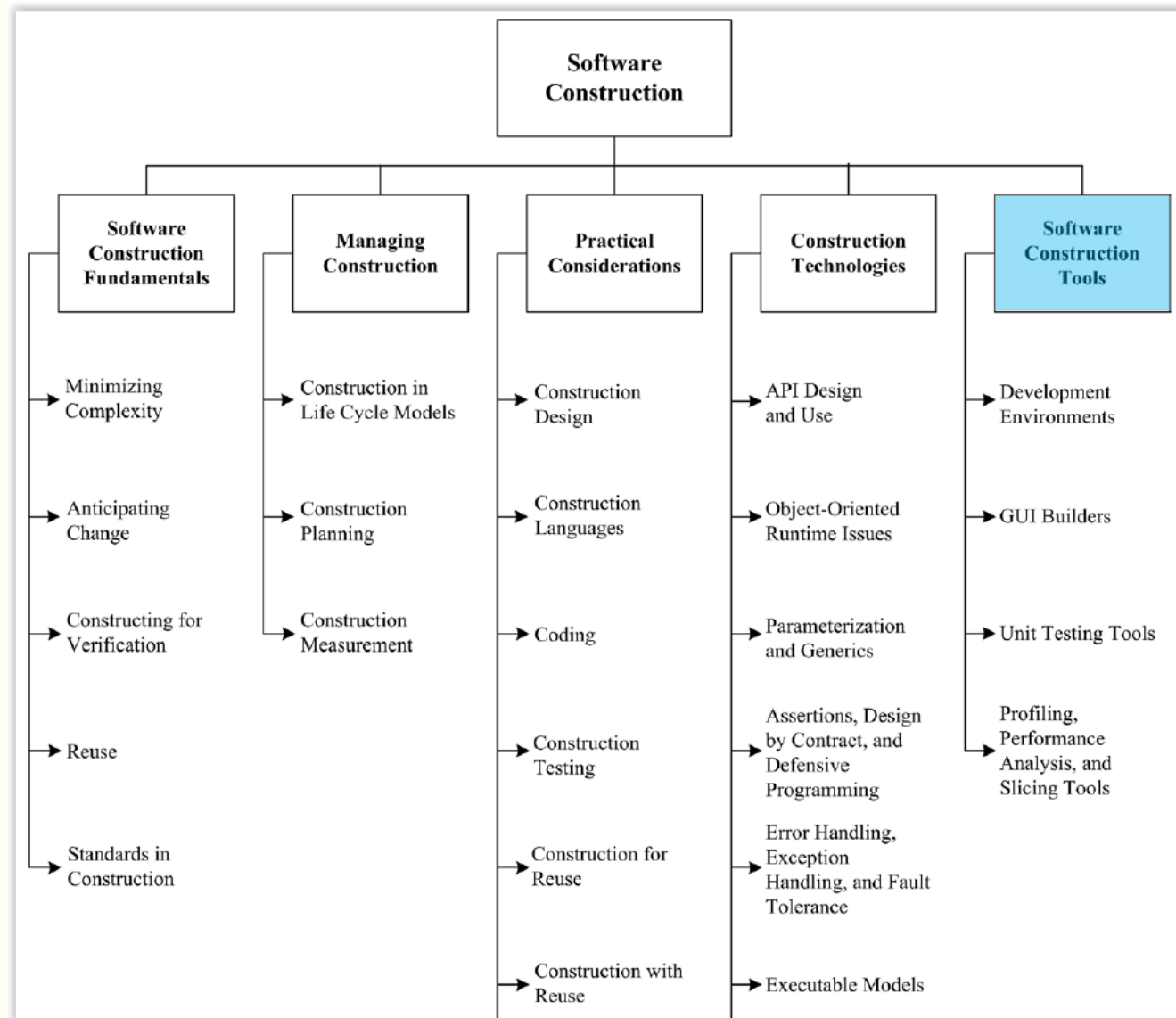
## 3.4.16 Programação Orientada a Testes

---

- Detecção precoce de defeitos
- Facilidade na correção de defeitos
- Requisitos e design



# Chapter 3: Software Construction





## 3.5 Ferramentas de Construção de Software

---

- **3.5.1 Ambientes de Desenvolvimento**
- **3.5.2 Construtores GUI**
- **3.5.3 Ferramentas de Teste de Unidades**
- **3.5.4 Ferramentas de criação de perfil, análise de desempenho e fatiamento**

## 3.5.1 Ambientes de Desenvolvimento

---

- Eficiência e qualidade do desenvolvimento
- Oferecem:
  - Edição de código
  - Compilação e detecção de erros
  - Integração com controle de código-fonte
  - Ferramentas de Build/Teste/Depuração
  - Visões comprimidas ou em formato de esboço
  - Transformações automáticas de código
  - Suporte para refatoração



## 3.5.2 Construtores GUI

---

- **Modo WYSIWYG (o que você vê é o que você obtém)**
- **Funcionalidades principais**
  - **Editor visual**
  - **Gerenciamento de layout**
  - **Geração automática de código**
- **Estilo orientado a eventos**
  - **Fluxo do programa determinado por eventos**
  - **Manipulação de eventos**
  - **Automatizam as tarefas mais repetitivas**



L<sup>A</sup>T<sub>E</sub>X



## 3.5.2 Construtores GUI

---

- **Integração com IDEs**
  - **Construtores de GUI integrados**
  - **Plug-ins de construtores de GUI**
- **Construtores de GUI independentes**

## 3.5.3 Ferramentas de Teste de Unidades

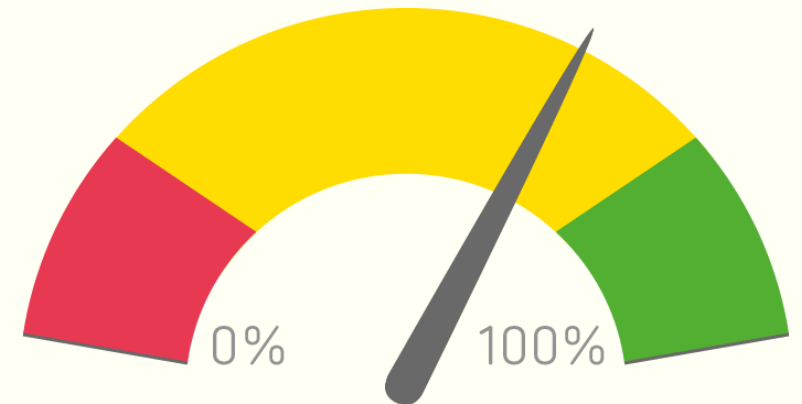
---

- **Ambiente automatizado**
- **Codificação de critérios**
  - **Verificar a correção da unidade**
  - **Diferentes conjuntos de dados**
- **Execução e relatórios**
  - **Implementação de testes individuais**
  - **Utilização de um executador de teste**
  - **Identificação automática e relatórios**

## 3.5.4 Criação de perfil, ferramentas de análise de desempenho e fatiamento

---

- Ferramentas de perfilamento
  - **Monitora do código em execução**
  - **Registro de frequência de execução**
  - **Registro de tempo de execução por declaração ou caminho**
  - **Benefícios**
    - **Identificação de pontos críticos**
    - **Orientação dos esforços de otimização**

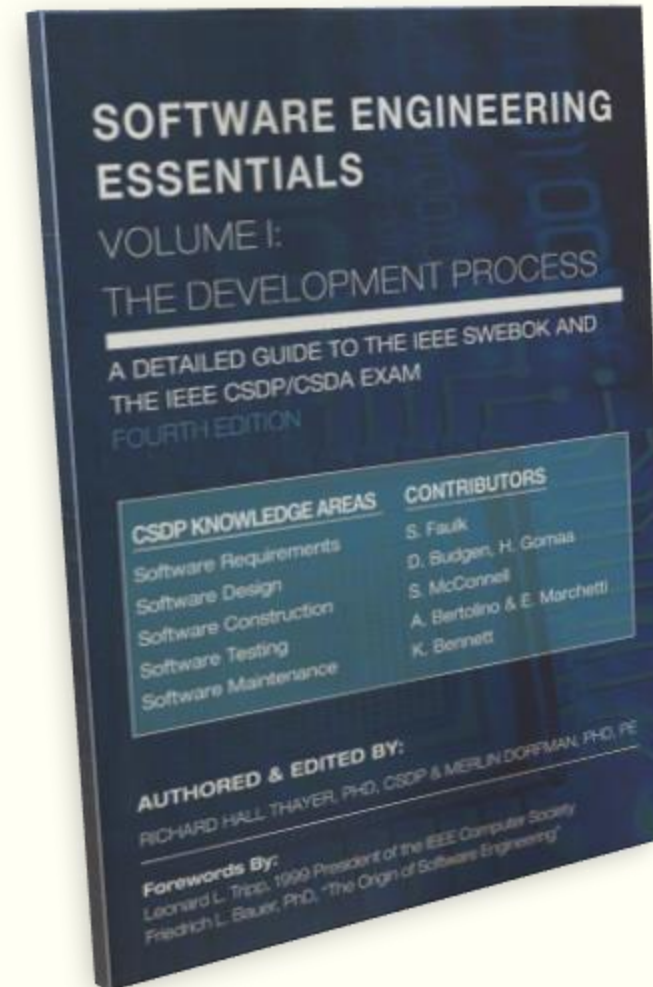
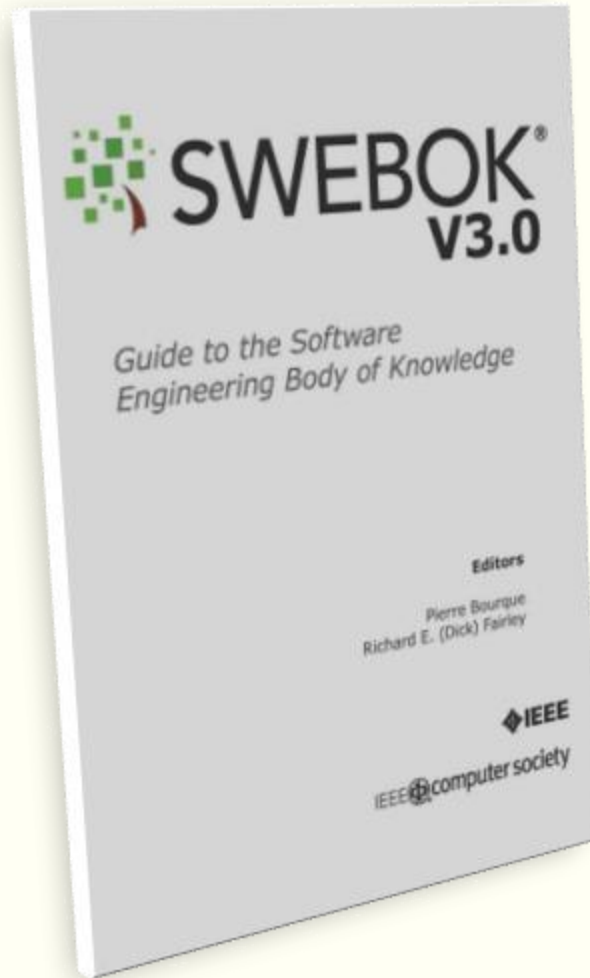


## 3.5.4 Criação de perfil, ferramentas de análise de desempenho e fatiamento

---

- **Fatiamento de programas**
  - **Cálculo do conjunto de declarações**
  - **Aplicações**
    - **Localização de erros**
    - **Compreensão do programa**
    - **Análise de otimização**
  - **Ferramentas de fatiamento de programas**
    - **Métodos estáticos e dinâmicos de análise**

# Bibliografia Básica





# Bibliografia Básica



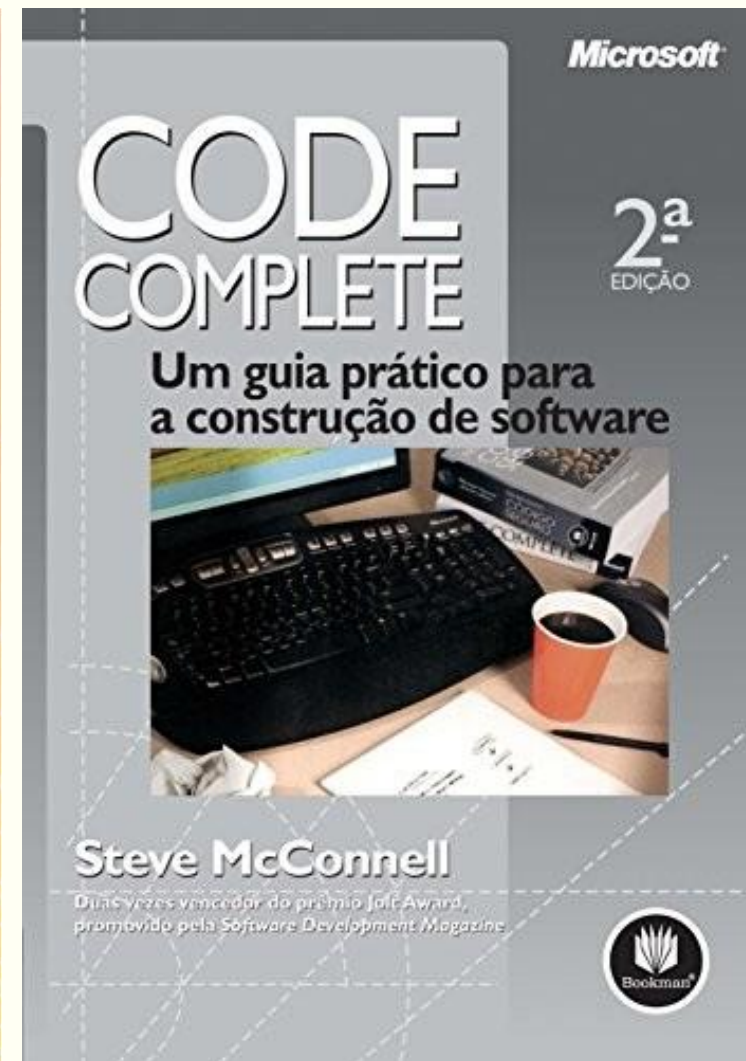
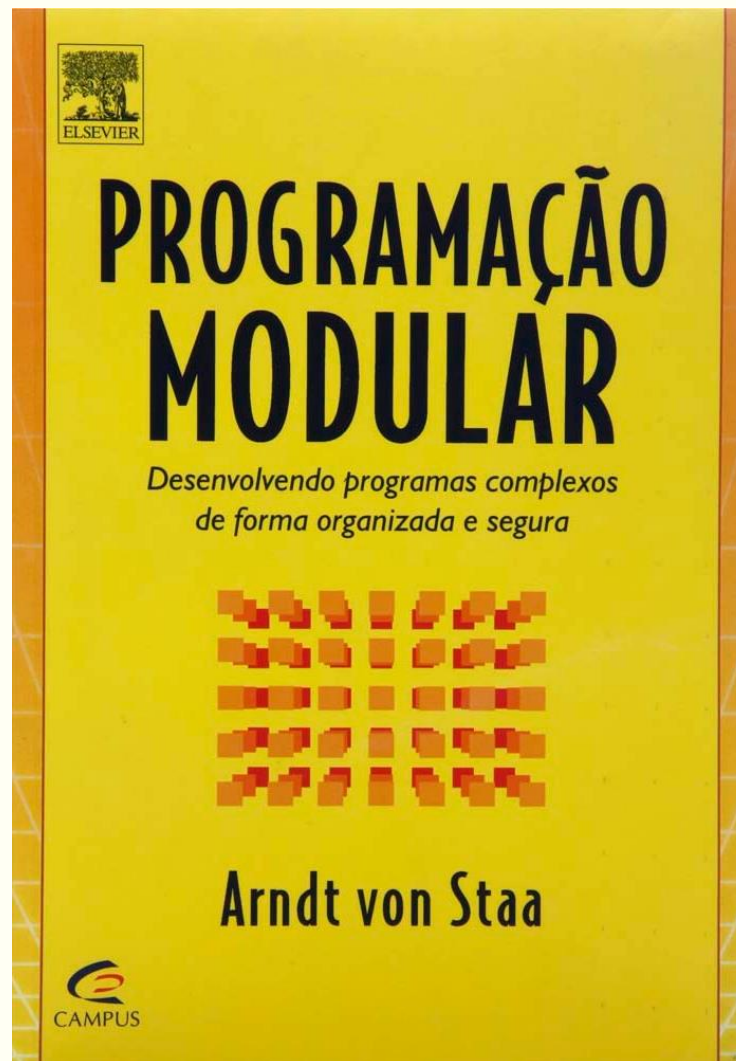
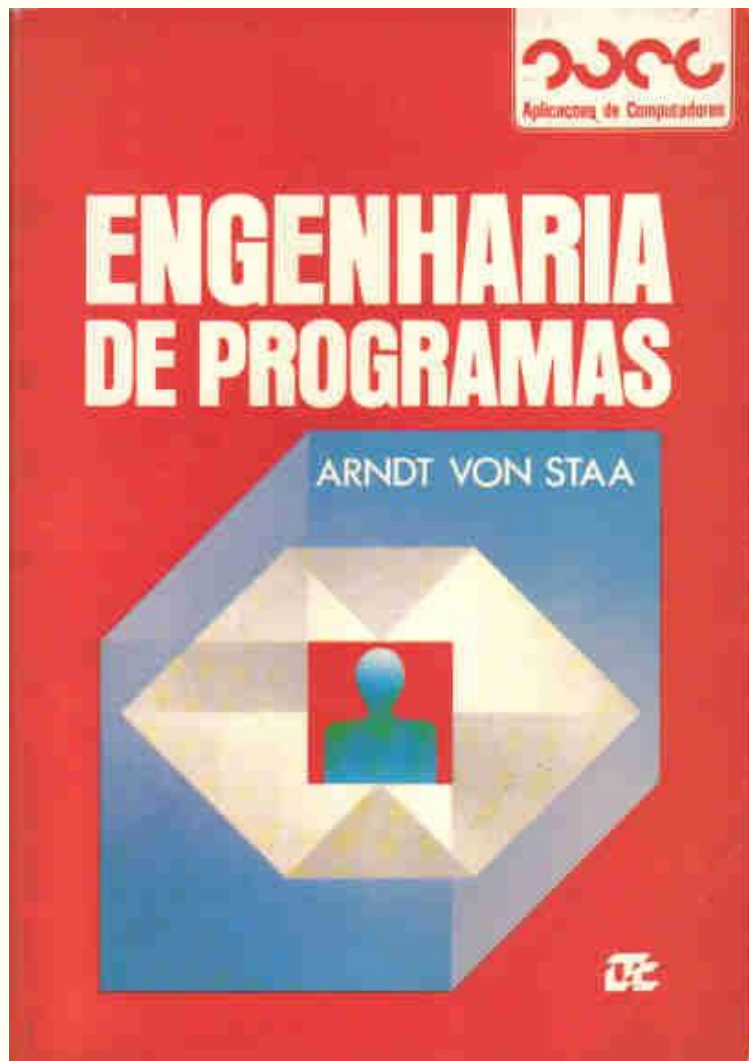
I. Sommerville  
Pearson  
2018



R. Pressman,  
B. R. Maxim  
AMGH Editora  
2016



# Bibliografia complementar





**Mariana Cossetti Dalfior**

**Ciência da Computação**

**UENF-CCT-LCMAT**

**Campos, RJ**

**[marianacossetti@hotmail.com](mailto:marianacossetti@hotmail.com)**

**[20211100064@pq.uenf.br](mailto:20211100064@pq.uenf.br)**