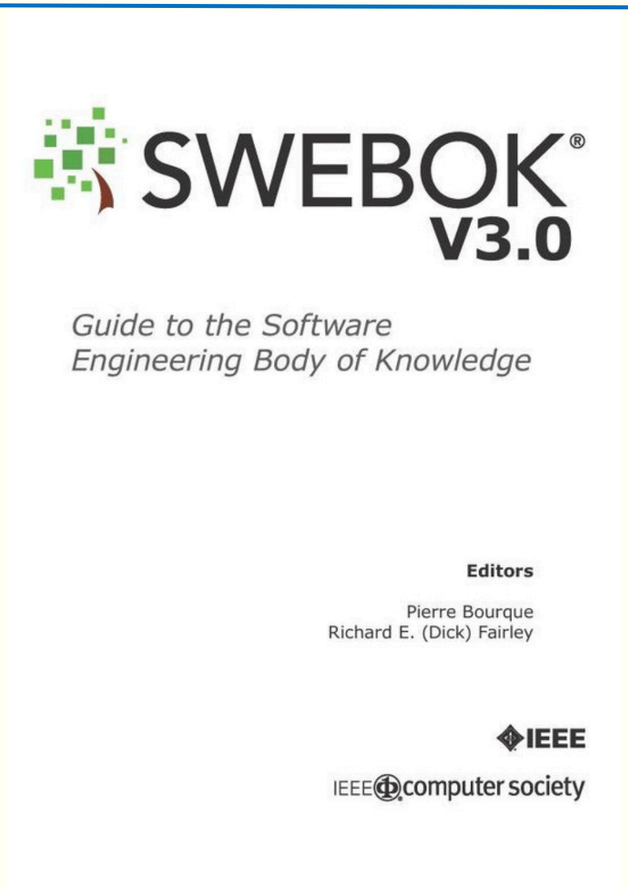
UENF

Modelos e Métodos de Engenharia de Software

Rômulo Souza Fernandes
23 de Abril de 2024



Introdução

- **Objetivo:** Estruturar a engenharia de software para sistematização, repetibilidade e orientação para o sucesso.
- **Uso de modelos:** Resolução de problemas, notação e procedimentos para construção e análise.
- **Uso de métodos:** Especificação, design, construção, teste e verificação do software final e produtos associados.
- **Variedade em escopo:** Única fase até o ciclo de vida completo do software.
- **Foco:** Múltiplas fases do ciclo de vida do software.

9.1 Modelagem

- **Definição:** Entender, projetar e comunicar aspectos do software aos interessados.
- **Interessados:** Usuários, Compradores, Fornecedores, Arquitetos, Autoridades Certificadoras, Desenvolvedores, Engenheiros de Software, entre outros.
- **Conceitos Gerais Unificadores:** Linguagens, notações, técnicas e ferramentas de modelagem.

9.1.1 Princípios de Modelagem

3 princípios gerais que orientam as atividades de modelagem:

- **Modelar o Essencial:** Desenvolver apenas os aspectos essenciais do software para manter os modelos gerenciáveis e úteis.
- **Fornecer Perspectiva:** Utilizar diferentes visões do software para concentrar-se em preocupações específicas relevantes para cada uma delas.
- **Possibilitar Comunicações Efetivas:** Empregar o vocabulário do domínio de aplicação e uma linguagem de modelagem para facilitar a comunicação eficaz das informações do software aos interessados.

9.1.1 Princípios de Modelagem

- **Modelo como uma Abstração:**
  - Simplificação de um componente de software.
  - É uma agregação de abstrações que descrevem aspectos selecionados.
- **Reutilização do Modelo:**
  - Requer validação dos pressupostos do contexto para estabelecer a relevância do modelo reutilizado em um novo contexto.

9.1.2 Propriedades e Expressão de Modelos

- Características distintivas de um modelo específico
- Caracterizar sua completude, consistência e correção dentro da notação de modelagem e ferramentas utilizadas.

9.1.2 Propriedades e Expressão de Modelos

- **Completude:** O grau em que todos os requisitos foram implementados e verificados dentro do modelo.
- **Consistência:** O grau em que o modelo não contém requisitos, assertivas, restrições, funções ou descrições de componentes conflitantes.
- **Correção:** O grau em que o modelo satisfaz seus requisitos e especificações de design e está livre de defeitos.

9.1.2 Propriedades e Expressão de Modelos

- **Entidades e Relações:** Representação de artefatos e suas conexões no modelo.
- **Linguagens de Modelagem:** Uso de linguagens textuais ou gráficas para expressar relações.
- **Significado das Entidades:** Definido por sua forma e atributos textuais.
- **Necessidade de Visualizações Múltiplas:** Para capturar todos os aspectos do software.

9.1.3 Sintaxe, Semântica e Pragmatismo

- Podem ser enganosos devido à sua natureza abstrata e informações incompletas.
- Compreender a sintaxe e a semântica dos modelos é crucial para uma interpretação precisa.
  - **Sintaxe:** Define as regras de estruturação dos modelos.
  - **Semântica:** Especifica o significado das entidades e relações no modelo.
- Pragmatismo na comunicação do significado do modelo, mesmo que esteja incompleto.

9.1.3 Sintaxe, Semântica e Pragmatismo

Ao modelar software, é necessário cautela.

- Partes do modelo importadas de outros ambientes devem ser examinadas quanto a suposições semânticas.
- Revisão contínua é fundamental para garantir a precisão e relevância do modelo ao longo do tempo.
- Desafios semânticos podem surgir à medida que o software evolui e várias pessoas contribuem para o modelo.

9.1.4 Precondições, Poscondições e Invariantes

- **Modelagem de Funções/Métodos:**
  - Pressupostos sobre o estado do software antes, durante e após a execução.
  - Essenciais para a operação correta.
- **Agrupamento:**
  - Pré-condições
  - Pós-condições
  - Invariantes

9.1.4 Precondições, Poscondições e Invariantes

- **Pré-condições:**
  - Condições necessárias antes da execução da função ou método para garantir resultados corretos.
- **Pós-condições:**
  - Condições garantidas após a execução bem-sucedida da função ou método, representando mudanças no estado do software.
- **Invariáveis:**
  - Condições constantes antes e após a execução da função ou método, fundamentais para sua operação correta.

9.2 Tipos de Modelos

- **Modelo típico:**
  - Consiste em uma agregação de submodelos.
  - Cada submodelo é uma descrição parcial criada para um propósito específico.
- **Submodelos:**
  - Pode ser composto por um ou mais diagramas.
  - Podem empregar múltiplas linguagens de modelagem ou uma única linguagem, como a Linguagem de Modelagem Unificada (UML).

9.2 Tipos de Modelos

- **Tipos amplos de modelos comumente usados:**
  - Modelos de informação.
  - Modelos comportamentais.
  - Modelos de estrutura.

9.2.1 Modelagem da Informação

- Foco central em dados e informações.
- **Representação abstrata identificando e definindo:**
  - Conjunto de conceitos.
  - Propriedades.
  - Relações.
  - Restrições em entidades de dados.
- **Modelo de informação semântico ou conceitual:**
  - Oferece formalismo e contexto ao software.
  - Perspectiva do problema, sem se preocupar com a implementação.

9.2.1 Modelagem da Informação

- Abstração que inclui apenas conceitos, propriedades, relações e restrições necessários para conceituar a visão do mundo real das informações.
- **Transformações subsequentes:**
  - Elaboração de modelos de dados lógicos e físicos conforme implementados no software.

9.2.2 Modelagem Comportamental

- Identificação e definição das funções do software modelado.
- **Três formas básicas de modelos comportamentais:**
  - Máquinas de estados.
  - Modelos de fluxo de controle.
  - Modelos de fluxo de dados.

9.2.2 Modelagem Comportamental

- **Máquinas de Estados:**
  - Modelo do software como uma coleção de estados, eventos e transições.
  - Transição entre estados por meio de eventos de acionamento.
- **Modelos de Fluxo de Controle:**
  - Demonstração de como sequências de eventos ativam e desativam processos.
- **Modelos de Fluxo de Dados:**
  - Sequências de etapas em que os dados se movem por processos

9.2.3 Modelagem Estrutural

- Ilustra a composição física ou lógica do software.
- **Elementos estruturais comuns:**
  - Composição e decomposição de entidades.
  - Generalização e especialização de entidades.
  - Identificação de relações e cardinalidade entre entidades.
  - Definição de interfaces de processo ou funcional.
  - Diagramas de estrutura da UML:

9.2.3 Modelagem Estrutural

- **Diagramas de classes**
  - Diagramas de componentes
  - Diagramas de objetos
  - Diagramas de implantação
  - Diagramas de empacotamento

9.3 Análise de Modelos

- Estudar, raciocinar e entender a estrutura, função, uso operacional e considerações de montagem associadas ao software.
- Modelos completos, consistentes e corretos para servir ao seu propósito pretendido.

9.3.1 Análise de Completude

- Fundamental desde o processo de elicitação de requisitos até a implementação do código.
- Grau em que todos os requisitos especificados foram implementados e verificados.
- Verificação por uma ferramenta
  - Análise estrutural
  - Análise de alcançabilidade do espaço de estados
- Verificação manual
  - Inspeções ou outras técnicas de revisão

9.3.2 Análise de Consistência

- Grau em que os modelos não contêm requisitos, assertivas, restrições, funções ou descrições de componentes conflitantes.
- **Ferramentas de Modelagem**
  - Função de análise automatizada.
- **Verificação Manual**
  - Inspeções ou outras técnicas de revisão.
- Erros e Aviso indicam necessidade de ação corretiva.

9.3.3 Análise de Exatidão (Correção)

- Grau em que um modelo satisfaz seus requisitos e especificações de design, é livre de defeitos e atende às necessidades das partes interessadas.
- **Verificação Sintática**
  - Gramática e construções da linguagem de modelagem.
- **Verificação Semântica**
  - Linguagem de modelagem para expressar com precisão o que está sendo representado.

9.3.3 Análise de Exatidão (Correção)

- Verificação automática
  - Modelagem para verificar a correção sintática do modelo
- Verificação manual
  - Inspeções ou outras técnicas de revisão

9.3.4 Rastreabilidade

- Uso, criação e modificação
  - Documentos, especificações, diagramas e códigos.
- **Importância:** Consistência dos requisitos com o modelo de software, gerenciamento e qualidade do processo.
- **Benefícios:** Gerenciamento e qualidade, garantias às partes interessadas, análise de mudanças.
- **Ferramentas de Modelagem:** Meios automatizados ou manuais para especificar e gerenciar links de traçabilidade.

9.3.5 Análise de Interação

Relações de comunicação ou fluxo de controle entre entidades dentro do modelo de software.

- **Comportamento Dinâmico:** Interações entre diferentes partes do modelo de software, como SO, middleware e aplicativos.
- **Simulação:** Estudo do comportamento dinâmico do software modelado, revisar o design de interação e verificar a funcionalidade do software.

9.4 Métodos de Engenharia de Software

- Abordagem organizada e sistemática para desenvolver um software para um computador-alvo.
- Inúmeros métodos
- Uso por pessoas com habilidade adequada
  - Visualizar os detalhes do software
  - Transformar a representação em um conjunto de código e dados funcionais.

9.4.1 Métodos Heurísticos

- Métodos de engenharia de software baseados em experiência que foram e são amplamente praticados na indústria de software.

Contém 3 categorias de discussão:

- Métodos de análise e design estruturados
- Métodos de modelagem de dados
- Métodos de análise e design orientados a objetos.

9.4.1 Métodos Heurísticos

- Análise e Design Estruturados:
  - Desenvolvimento funcional ou comportamental.
  - Progressiva decomposição e refinamento dos componentes.
  - Convergência para detalhes específicos do software.
- Modelagem de Dados:
  - Construída a partir do ponto de vista dos dados.
  - Define tabelas e relacionamentos.
  - Usada em software empresarial para gerenciar dados.

9.4.1 Métodos Heurísticos

- Análise e Design Orientados a Objetos:
  - Representado como uma coleção de objetos.
  - Objetos encapsulam dados e interagem por métodos.
  - Construído com diagramas e refinamento progressivo.

9.4.2 Métodos Formais

- Abordagens matemáticas para especificar, desenvolver e verificar software.
- **Linguagens de Especificação:** Notações formais para descrever comportamentos de entrada/saída e requisitos.
- **Refinamento e Derivação:** Processo de criar uma representação executável do programa a partir de uma especificação de nível superior.
- **Verificação Formal:** Exploração do espaço de estados para demonstrar propriedades do modelo de interesse.
- **Inferência Lógica:** Especifica pré-condições e pós-condições para prever o comportamento do software sem executá-lo.

9.4.3 Métodos de Prototipação

- Cria versões preliminares para experimentar novos recursos, obter feedback e explorar requisitos.
- **Estilo de Prototipagem:** Aborda diferentes abordagens, como código descartável ou produtos de papel, dependendo dos resultados necessários e da urgência.
- **Alvo de Prototipagem:** Varia de uma especificação de requisitos a elementos de design, algoritmos ou interfaces usuário-máquina.
- **Técnicas de Avaliação:** Podem incluir testes em relação ao software final, avaliação em relação a requisitos ou uso como modelo para desenvolvimento futuro.

9.4.4 Métodos Ágeis

- Surgiu nos anos 90 para reduzir o overhead associado a métodos pesados.
- Ciclos de desenvolvimento curtos, equipes auto-organizadas e ênfase na entrega contínua de produtos funcionais.
- **RAD:** Usado em sistemas intensivos em dados, o RAD permite o desenvolvimento rápido e implantação de aplicativos empresariais.
- **XP:** Baseado em histórias de requisitos, desenvolve testes primeiro, envolve o cliente na definição de testes de aceitação e promove refatoração e integração contínuas.

9.4.4 Métodos Ágeis

- **Scrum:** Mais amigável ao gerenciamento de projetos, Scrum divide o trabalho em sprints de 30 dias, com entregas contínuas de incrementos testáveis.
- **FDD**: Iterativo e orientado a modelo, o FDD é baseado em um processo de cinco fases para desenvolver e integrar funcionalidades.
- **Variações e Tendências:** Métodos ágeis continuam evoluindo, combinando características de métodos baseados em planos e ágeis para atender às necessidades organizacionais.
- **Escolha de Método:** A escolha do método de engenharia de software é influenciada pelas necessidades do negócio e stakeholders do projeto.

Conclusão

- **Variedade de Metodologias:** Exploramos uma ampla gama de metodologias, desde as tradicionais até as mais ágeis, oferecendo uma visão abrangente da engenharia de software.
- **Ferramentas para Eficiência:** Analisamos modelos, métodos de análise e design, prototipagem e métodos ágeis, descobrindo ferramentas que promovem a eficiência e a colaboração no desenvolvimento de software.

Conclusão

- **Adaptação Contínua:** Reconhecemos a importância de adaptar as metodologias às necessidades específicas de cada projeto, equipe e cliente, para garantir o sucesso em um ambiente em constante mudança.
- **Inovação Sustentada:** Compreendemos que a compreensão profunda das práticas de engenharia de software nos capacita a impulsionar a inovação de forma sustentada em todas as áreas.

Rômulo Souza Fernandes

Ciência da Computação

UENF-CCT-LCMAT

Campos, RJ

00119110559@pq.uenf.br