

# Modelos e Métodos de Engenharia de Software

Rômulo Souza Fernandes  
18 de Junho de 2024



*Guide to the Software  
Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



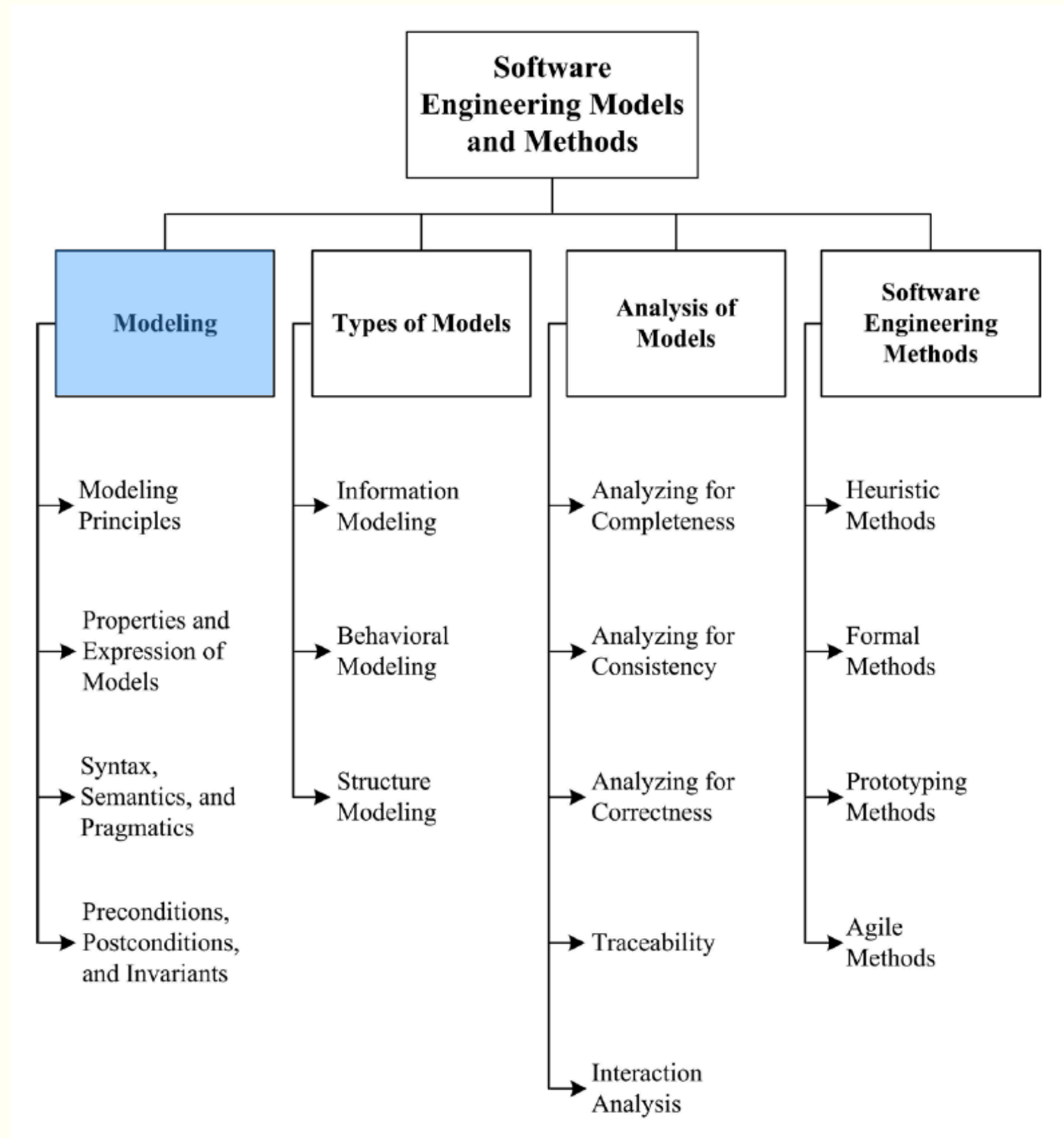
IEEE  computer society

# Chapter 9: Software Engineering Models and Methods

---

- **1. Modeling**
  - **1.1. Modeling Principles**
  - **1.2. Properties and Expression of Models**
  - **1.3. Syntax, Semantics, and Pragmatics**
  - **1.4. Preconditions, Postconditions, and Invariants**
- **2. Types of Models**
  - **2.1. Information Modeling**
  - **2.2. Behavioral Modeling**
  - **2.3. Structure Modeling**
- **3. Analysis of Models**
  - **3.1. Analyzing for Completeness**
  - **3.2. Analyzing for Consistency**
  - **3.3. Analyzing for Correctness**
  - **3.4. Traceability**
  - **3.5. Interaction Analysis**
- **4. Software Engineering Methods**
  - **4.1. Heuristic Methods**
  - **4.2. Formal Methods**
  - **4.3. Prototyping Methods**
  - **4.4. Agile Methods**

# 9.1 Modelagem



*Guide to the Software Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



IEEE  computer society

# 9.1 Modelagem

---

- **9.1.1 Princípios de Modelagem**
- **9.1.2 Propriedades e Expressão de Modelos**
- **9.1.3 Sintaxe, Semântica e Pragmatismo**
- **9.1.4 Precondições, Poscondições e Invariantes**

# Introdução

---

- **Objetivo:** Estruturar a engenharia de software para sistematização, repetibilidade e aumentar as chances de sucesso.
- **Modelos:** Resolução de problemas, fornecendo notações e procedimentos para a construção e análise.
- **Métodos:** Especificação, design, construção, teste, verificação do software final e produtos associados.

# Introdução

---

- **Escopo:** Única fase até o ciclo de vida completo do software.
- **Foco:** Múltiplas fases do ciclo de vida do software.

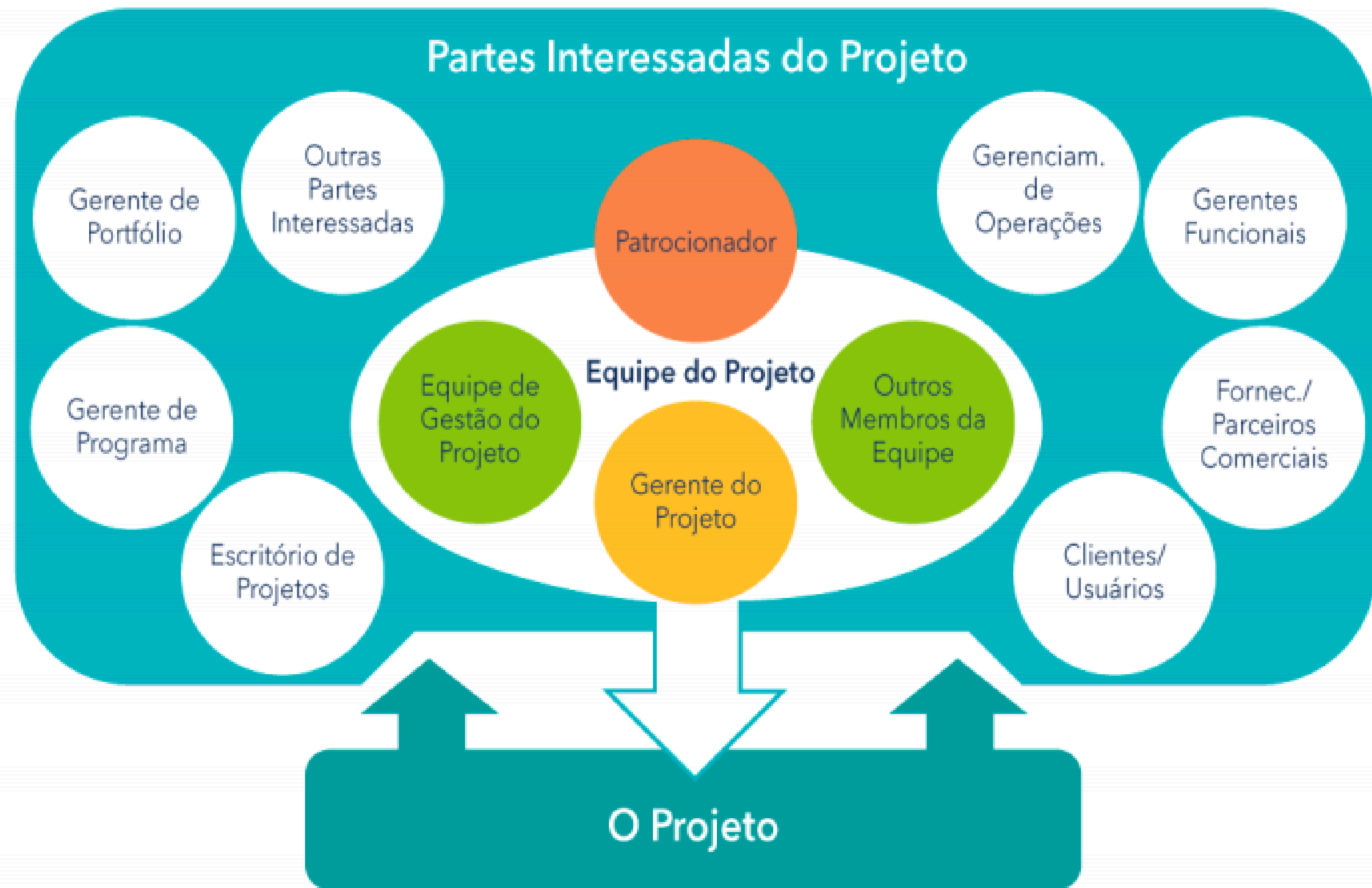


# 9.1 Modelagem

- Técnica utilizada para:
  - Entender
  - Projetar
  - Comunicar
- Interessados



## 9.1 Modelagem





# 9.1 Modelagem

---

- **Conceitos Gerais Unificadores:**
  - Linguagens
  - Notações
  - Técnicas
  - Ferramentas

## 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.

Exemplo: Sistema de reservas de um restaurante

## 9.1.1 Princípios de Modelagem

---

3 princípios gerais que orientam as atividades de modelagem:

- **Fornecer Perspectiva:** Utilizar diferentes perspectivas do software para concentrar-se em preocupações específicas relevantes.

## 9.1.1 Princípios de Modelagem



## 9.1.1 Princípios de Modelagem

---

3 princípios gerais que orientam as atividades de modelagem:

- **Possibilitar Comunicações Efetivas:**

- Vocabulário claro e conciso
- Linguagem de modelagem
- Expressão semântica (significado dentro do contexto)
- Relatórios e documentação

## 9.1.1 Princípios de Modelagem

---

- **Modelo como uma Abstração:**
  - Simplificação de um componente de software, focando nos aspectos essenciais.
  - Nenhuma abstração única descreve completamente um componente de software.
  - Agregação de abstrações que descrevem perspectivas, aspectos ou visões.

## 9.1.2 Propriedades e Expressão de Modelos

---

Características únicas de um modelo, usadas para caracterizar sua **completude**, **consistência** e **correção** dentro da notação de modelagem e ferramentas utilizadas.

## 9.1.2 Propriedades e Expressão de Modelos

---

- **Completeness:** 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

---

- **Incompleto:** O sistema de reservas permite que os clientes façam reservas e modifiquem reservas existentes, mas não inclui a funcionalidade de cancelamento de reservas.
- **Completo:** Um modelo completo mapeará todo o fluxo de reservas, desde a criação da reserva, modificação e cancelamento, até a confirmação da reserva, garantindo que todas as etapas sejam implementadas e verificadas.

## 9.1.2 Propriedades e Expressão de Modelos

---

- **Inconsistente:** O modelo especifica que uma mesa pode ser reservada por um máximo de 2 horas, mas em outra parte do modelo, é mencionado que a duração máxima pode ser de 3 horas.
- **Consistente:** O modelo especifica de forma consistente que a duração máxima de uma reserva é de 2 horas em todas as partes do sistema.

## 9.1.2 Propriedades e Expressão de Modelos

---

- **Incorreto:** O modelo exige que as reservas sejam permitidas apenas durante o horário de funcionamento do restaurante (11h às 23h), mas permite reservas fora desse horário.
- **Correto:** O modelo implementa corretamente a regra de que as reservas só podem ser feitas durante o horário de funcionamento do restaurante (11h às 23h), bloqueando qualquer tentativa de reserva fora desse horário.

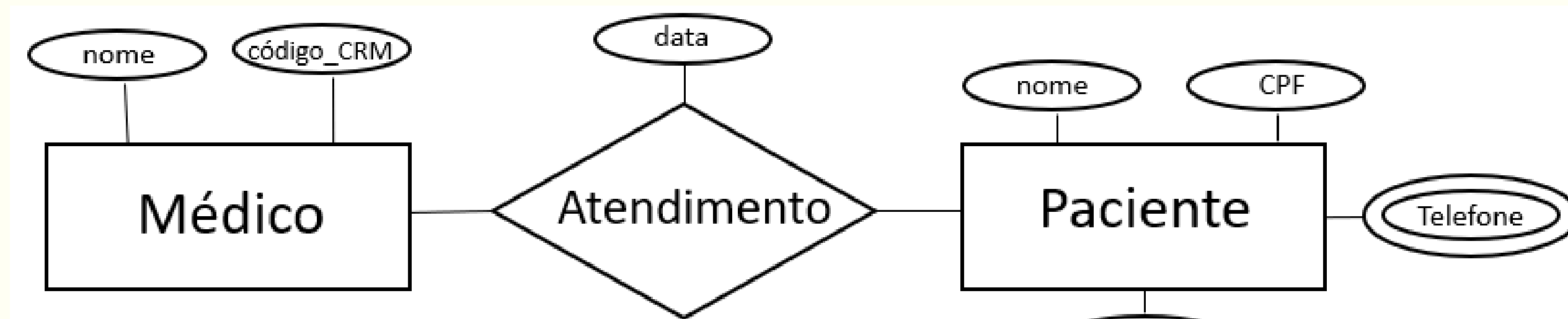
## 9.1.2 Propriedades e Expressão de Modelos

---

- **Representação e Comportamento:** Modelos são criados para representar objetos do mundo real e seus comportamentos, ajudando a responder questões sobre a operação do software.
- **Identificação de Incertezas:** Através da exploração, simulação ou revisão dos modelos, é possível identificar e tratar áreas de incerteza ou perguntas não respondidas sobre os requisitos, design e implementação do software.

## 9.1.2 Propriedades e Expressão de Modelos

- **Entidades podem representar artefatos, como:**
  - **Artefatos concretos:** Processadores, sensores ou robôs.
  - **Artefatos abstratos:** Módulos de software ou protocolos de comunicação.
- Entidades são conectadas a outras usando relações



## 9.1.2 Propriedades e Expressão de Modelos

---

- **Linguagens de Modelagem:** Uso de linguagens **textuais ou gráficas** para expressar as entidades e relações.
- **Significado de uma entidade e relações depende da:**
  - Linguagem de modelagem usada
  - Rigor no design
  - Visualização específica
  - Entidade em questão

## 9.1.2 Propriedades e Expressão de Modelos

---

- **Múltiplas Visualizações:** Para capturar todas as informações sobre o software.
- **Modelos com suporte de automação:** Podem ser verificados quanto à **completude e consistência**.
- Enquanto a **correção** dos modelos geralmente é verificada por meio de simulação e/ou revisão.

## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

- Modelos podem ser enganosos devido à sua natureza abstrata e informações incompletas.
- Um modelo completo pode ser a união de múltiplos submodelos.
- Exemplo:
  - Aumentar o número de reservas disponíveis para cada hora de pico.
  - Capacidade da cozinha, a disponibilidade de funcionários, eventos especiais que possam afetar a demanda.



## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

- 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.

## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

- **Sintaxe**

- **Linguagem textual:** Utiliza uma gramática de notação que define construções válidas, como por exemplo a forma de Backus-Naur.
- **Linguagem Visual:** Definida por modelos gráficos chamados metamodelos, que determinam como as construções sintáticas podem ser combinadas para formar modelos coerentes.

## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

- **Semântica**

- **Linguagem textual:** Especificada por meio de regras e convenções que determinam o significado de cada elemento expresso textualmente.
- **Linguagem Visual:** É transmitida pela interpretação de elementos e da forma como se interconectam.

## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

- O significado é transmitido pelo modelo, mesmo quando as informações estão incompletas, através da abstração.
- A pragmática detalha como esse significado é integrado ao modelo e ao seu contexto.
- Garante uma comunicação eficaz com outros engenheiros de software.

## 9.1.3 Sintaxe, Semântica e Pragmatismo

---

Exemplo: Como os clientes fazem reservas, como as informações são registradas, como os assentos são alocados.

- **Abstração:** Omitir detalhes como a decoração ou música ambiente.
- **Pragmática:** Utilizar diagramas e símbolos, descrições detalhadas, treinamento e ferramentas para comunicação, como controle de versão.

## 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 ou Métodos:**
  - Inicia com 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.

### Exemplo:

- O cliente deve ter uma conta no sistema de reservas e estar logado.
- Deve haver pelo menos uma mesa disponível no horário solicitado.
- O cliente deve fornecer todas as informações necessárias.



## 9.1.4 Precondições, Poscondições e Invariantes

---

- **Pós-condições:**

- Condições garantidas após a execução bem-sucedida da função ou método, representando mudanças no estado do software.

### Exemplo:

- Uma confirmação de reserva é enviada ao cliente por e-mail ou mensagem de texto.
- O sistema de reservas atualiza o banco de dados
- A tabela de disponibilidade do restaurante é atualizada

## 9.1.4 Precondições, Poscondições e Invariantes

---

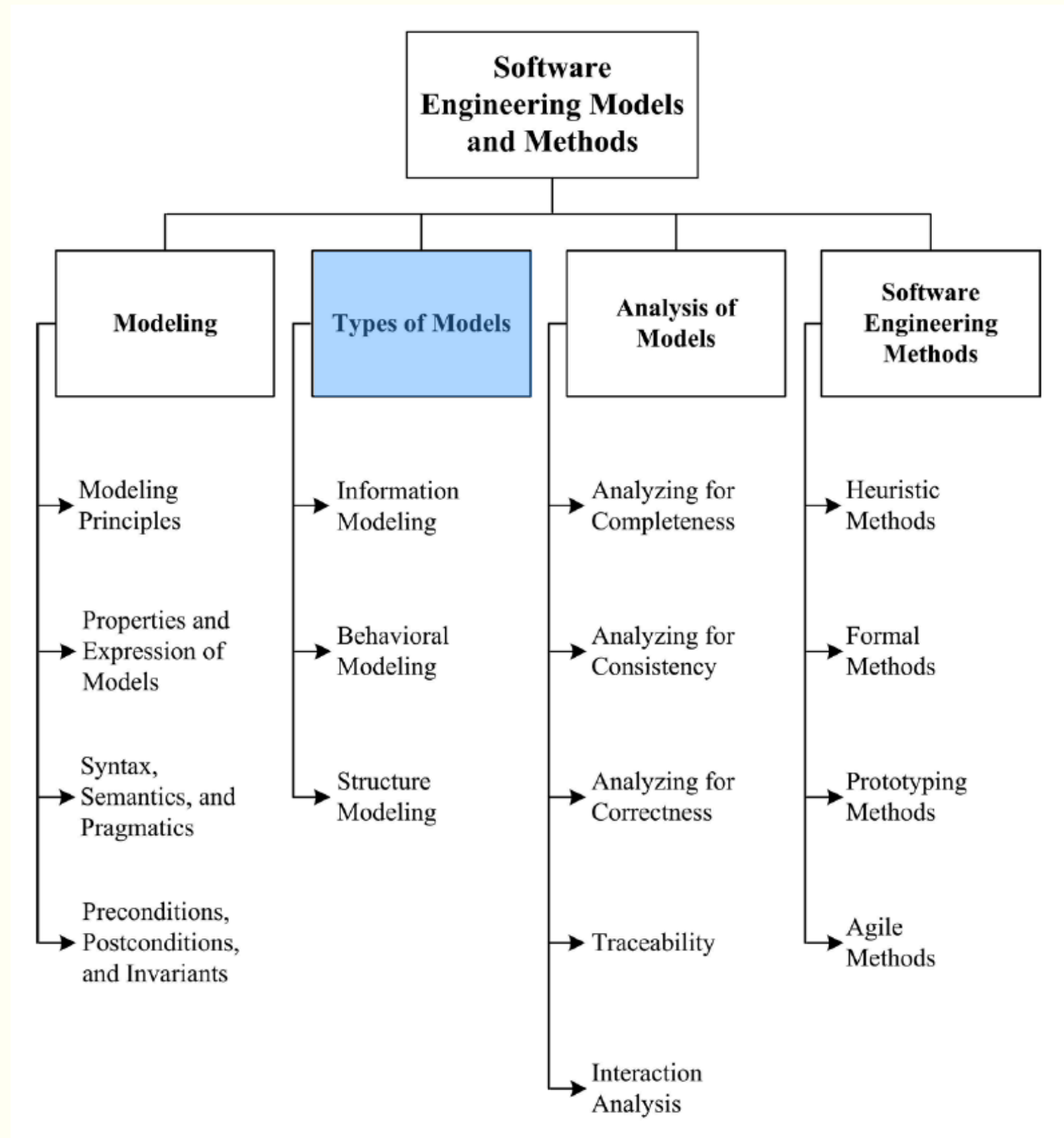
- **Invariáveis:**

- Condições constantes antes e após a execução da função ou método, fundamentais para sua operação correta.

### Exemplo:

- O horário de funcionamento do restaurante.
- A capacidade máxima do restaurante.
- Tempo mínimo e máximo para fazer uma reserva.
- Políticas de cancelamento.

## 9.2 Tipos de Modelos



*Guide to the Software Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



IEEE  computer society

## 9.2 Tipos de Modelos

---

- **9.2.1 Modelagem da Informação**
- **9.2.2 Modelagem Comportamental**
- **9.2.3 Modelagem Estrutural**

## 9.2 Tipos de Modelos

---

- **Modelo típico:**
  - Agregação de submodelos.
    - Descrição parcial criada para um propósito específico.
    - Pode ser composto por um ou mais diagramas.
    - Podem empregar múltiplas linguagens de modelagem ou uma única.

## 9.2 Tipos de Modelos

---

- **Modelo típico:**
  - Linguagem de Modelagem Unificada (UML).
  - Diagramas junto com construções da linguagem.



## 9.2 Tipos de Modelos

---

- **3 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 que identifica e define:**
  - Conjunto de conceitos.
  - Propriedades.
  - Relações.
  - Restrições em entidades de dados.
- **Frequentemente utilizado para:**
  - 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.

## 9.2.1 Modelagem da Informação

---

### Transformações Subsequentes: Refinamento

- **Modelo de Dados Lógico:**
  - Definição de tabelas de banco de dados relacionais, com chaves primárias e estrangeiras.
- **Modelo de Dados Físico:**
  - Seleção de tipos de dados, configuração de índices, particionamento ou fragmentação de tabelas, alocação de espaço em disco e backup.

## 9.2.2 Modelagem Comportamental

---

Identificação e definição das funções do software que está sendo modelado.

- **Três formas básicas:**
  - 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.

### Exemplo: Carrinho de compras

- Estados
  - Carrinho vazio e carrinho com produtos
- Eventos
  - Adicionar produtos e remover produtos
- Transição:
  - Carrinho vazio para carrinho com produtos

## 9.2.2 Modelagem Comportamental

---

- **Modelos de Fluxo de Controle:**
  - Demonstração de como sequências de eventos ativam e desativam processos.

### Exemplo: Pagamento

1. O cliente seleciona a forma de pagamento
2. O sistema verifica a validade dos dados de pagamento.
3. Se os dados forem válidos:
  - a. O sistema processa o pagamento e confirma a compra.
4. Se os dados forem inválidos:
  - a. O sistema exibe uma mensagem de erro informando o problema.

## 9.2.2 Modelagem Comportamental

---

- **Modelos de Fluxo de Dados:**
  - Sequências de etapas em que os dados se movem por processos.

### Exemplo: Processamento do pedido

- **Dados:**
  - Dados do cliente, pedido e pagamento
- **Processamento:**
  - Validar os dados do cliente e pedido
  - Verificar a disponibilidade dos produtos no estoque
  - Processar o pagamento e registrar a transação

## 9.2.3 Modelagem Estrutural

---

- Ilustram como o software é organizado, mostrando suas diferentes partes e como elas se encaixam para formar o todo
- Estabelece a fronteira definida entre o software que está sendo modelado e o ambiente no qual ele deve operar.
- **Elementos estruturais comuns:**
  - Composição, decomposição, generalização e especialização de entidades

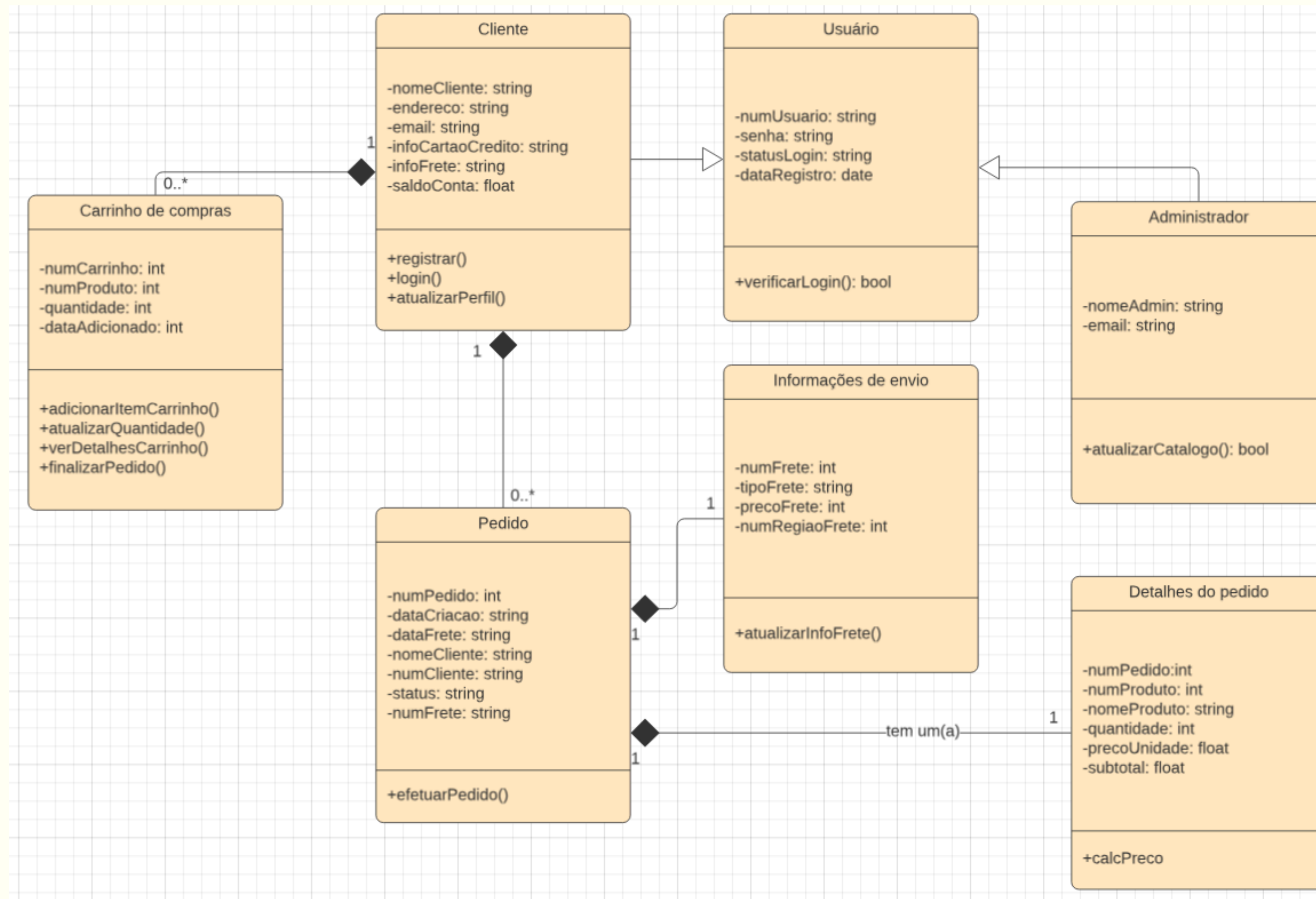
## 9.2.3 Modelagem Estrutural

---

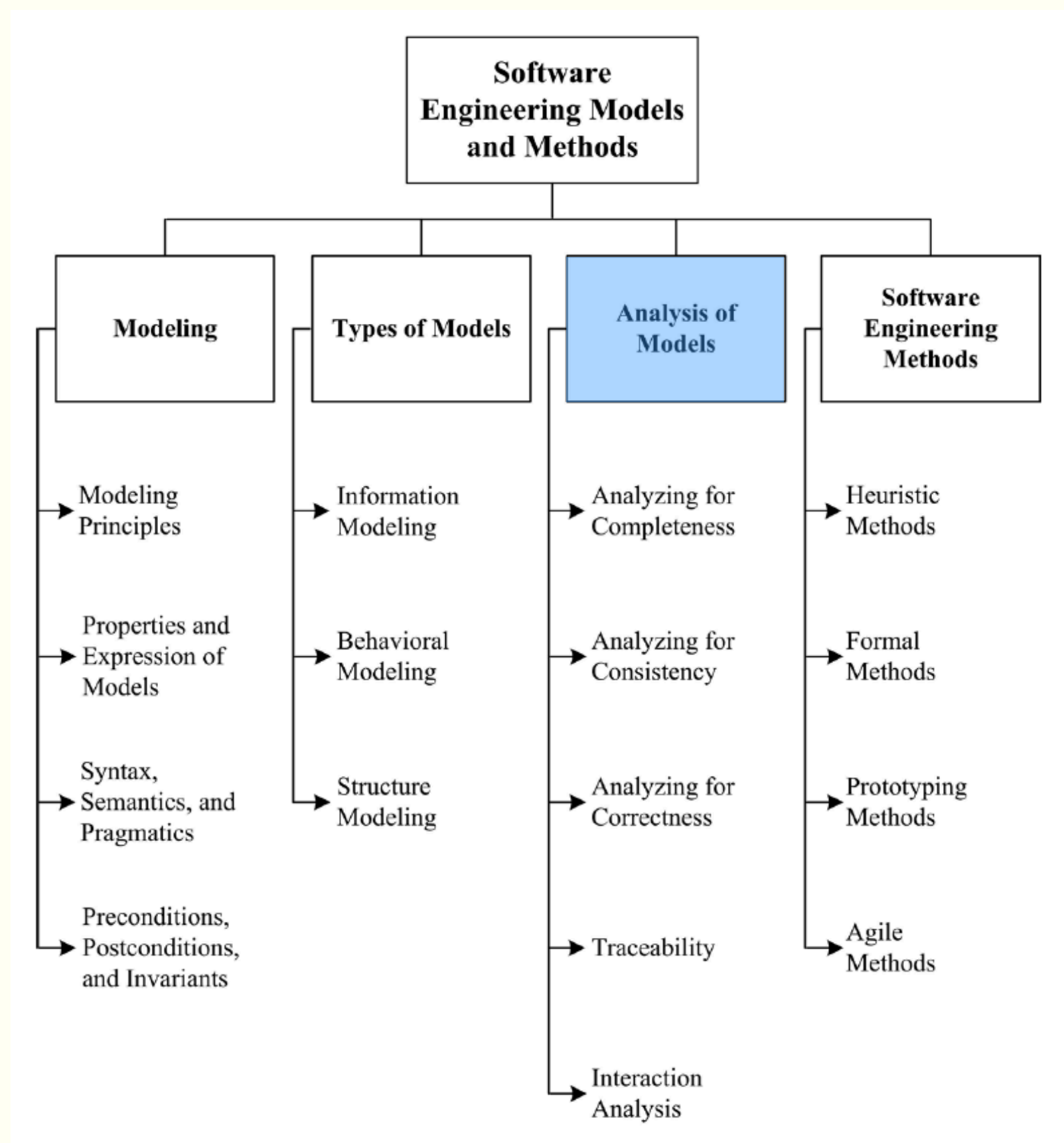
- **Elementos estruturais comuns:**
  - Identificação de relações e cardinalidade entre entidades.
  - Definição de interfaces de processo ou funcional.
- **Diagramas UML incluem:**
  - Diagramas de classes, componentes, objetos, implantação e empacotamento.



## 9.2.3 Modelagem Estrutural



## 9.3 Análise de Modelos




*Guide to the Software Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



IEEE  computer society

## 9.3 Análise de Modelos

---

- **9.3.1 Análise de Completeza**
- **9.3.2 Análise de Consistência**
- **9.3.3 Análise de Exatidão (Correção)**
- **9.3.4 Rastreabilidade**
- **9.3.5 Análise de Interação**

## 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

---

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

## 9.3.1 Análise de Completude

---

Exemplo: Sistema de gestão hospitalar

Verifica se o sistema permite todas as transições necessárias para o fluxo completo de uma consulta médica, desde o agendamento até o registro da consulta no prontuário do paciente.

## 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.

## 9.3.2 Análise de Consistência

---

Exemplo:

Podem surgir conflitos entre os requisitos de acesso ao prontuário do paciente: um requisito diz que apenas médicos podem editar o prontuário, enquanto outro permite que enfermeiros façam alterações.



## 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

## 9.3.3 Análise de Exatidão (Correção)

---

Exemplo de correção sintática:

Durante a verificação de um diagrama de sequência para o agendamento de consultas, a ferramenta detecta que a mensagem de retorno "confirmação de agendamento" está usando uma linha sólida em vez de uma linha pontilhada, o que é incorreto na notação UML.

## 9.3.3 Análise de Exatidão (Correção)

---

Exemplo de correção semântica:

O sistema deve permitir que pacientes agendem consultas com médicos disponíveis, evite agendamentos duplicados e envie notificações após a confirmação da consulta.

## 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 rastreabilidade.

## 9.3.4 Rastreabilidade

---

### Exemplo:

Cada requisito de agendamento de consultas é vinculado aos seus diagramas UML correspondentes, ao código-fonte implementado e aos casos de teste associados. Se uma mudança no requisito de agendamento de consultas for necessária, a ferramenta de rastreabilidade ajuda a identificar todas as áreas impactadas

## 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.

## 9.3.5 Análise de Interação

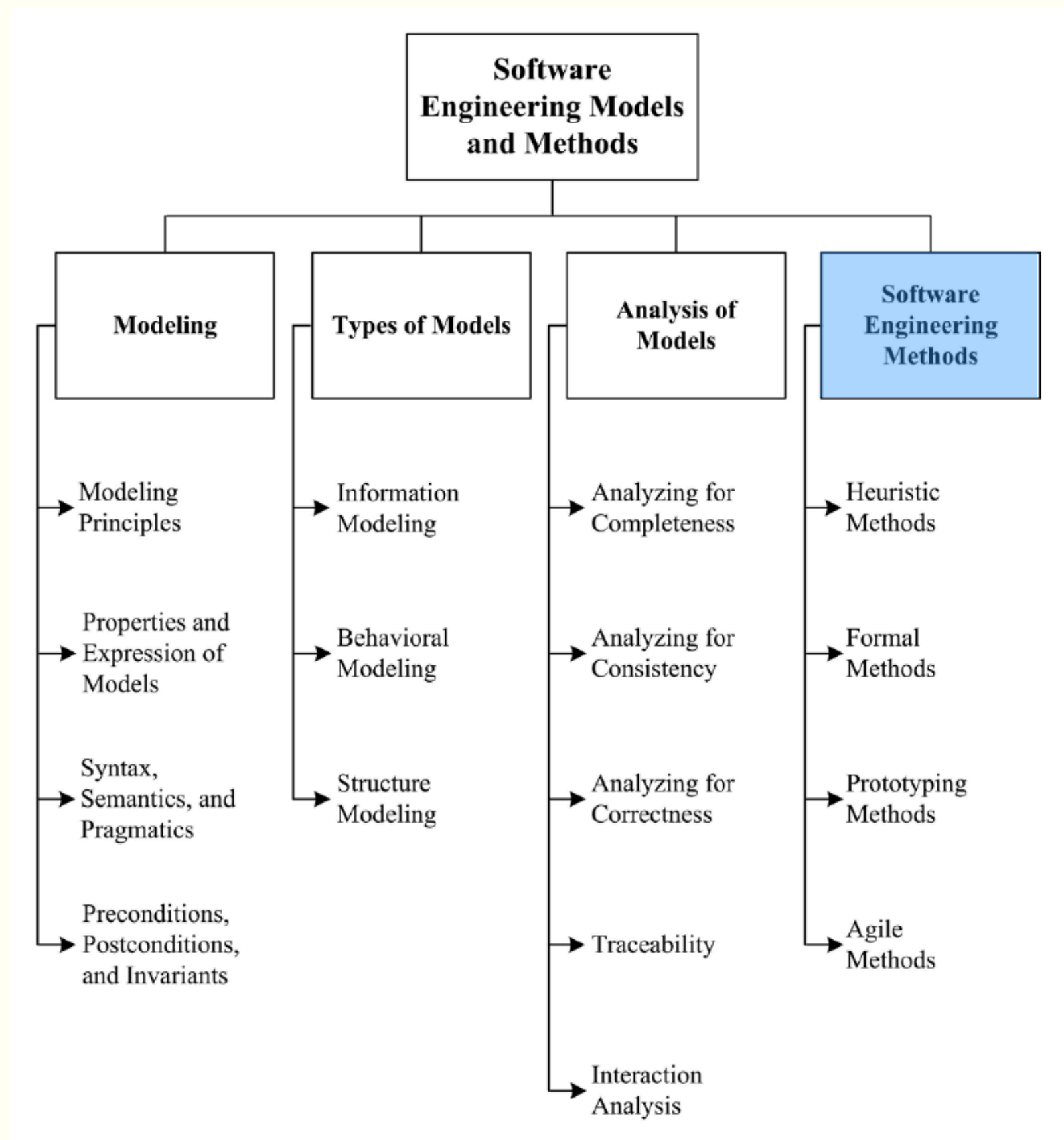
---

### **Exemplo:**

- O sistema deve permitir que o usuário selecione um médico e um horário disponível, enviando uma solicitação ao servidor.
- O servidor verifica a disponibilidade e confirma o agendamento, atualizando o banco de dados.
- Uma notificação é enviada ao paciente e ao médico.



# 9.4 Métodos de Engenharia de Software



*Guide to the Software Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



IEEE  computer society

## 9.4 Métodos de Engenharia de Software

---

- **9.4.1 Métodos Heurísticos**
- **9.4.2 Métodos Formais**
- **9.4.3 Métodos de Prototipação**
- **9.4.4 Métodos Ágeis**

## 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

---

- Baseados em experiências que foram e são amplamente praticadas 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:
  - Desenvolvido com ponto de vista funcional ou comportamental.
  - Decompõe ou refina progressivamente os componentes
  - Design detalhado
  - Convergência para detalhes específicos do software.

## 9.4.1 Métodos Heurísticos

---

- Modelagem de Dados:
  - Construída a partir do ponto de vista dos dados.
  - Definir e analisar os requisitos de dados que suportam design de banco de dados ou repositório de dados
  - Empresarial
  - Recurso ou ativo de sistemas de negócios

## 9.4.1 Métodos Heurísticos

---

- Análise e Design Orientados a Objetos:
  - Representado como uma coleção de objetos.
    - Encapsulam dados
    - Interagem com outros objetos por métodos.
  - Construído com diagramas e refinamento progressivo.
    - Real ou virtual

## 9.4.2 Métodos Formais

---

- Usados para especificar, desenvolver e verificar software.
  - Notação e linguagem baseadas em matemática
- **Linguagens de Especificação:**
  - Fornece a base matemática
  - Não é uma linguagem clássica de 3º geração
  - Utilizada para:
    - Especificação de software
    - Análise de requisitos
    - Estágios de design
    - Descrever comportamentos de entrada e saída



## 9.4.2 Métodos Formais

---

- **Refinamento e Derivação:**

- Cria uma especificação de nível inferior com transformações
- Especificações podem ser refinadas, adicionando detalhes
- Devem definir também os significados exatos em tempode de execução das relações entre entidades

## 9.4.2 Métodos Formais

---

- **Verificação Formal:** Exploração do espaço de estados para demonstrar que o design representado possui ou preserva as propriedades do modelo de interesse

Exemplo: Análise que verifica o comportamento correto do programa sob todas as possíveis intercalações de eventos ou chegadas de mensagens

Assume a forma de uma máquina de estado finito ou outro autômato formalmente definido

## 9.4.2 Métodos Formais

---

- **Inferência Lógica:** Método de design de software que usa princípios da lógica matemática para garantir que certas condições sejam mantidas antes e depois da execução de determinadas partes do código

## 9.4.3 Métodos de Prototipação

---

Cria versões incompletas ou minimamente funcionais de um software, para experimentar novos recursos específicos, obter feedback sobre requisitos, explorar mais requisitos, design de software ou opções de implementação.

Seleciona um método para entender os aspectos ou componente menos compreendidos primeiro, em contraste dos outros métodos que geralmente iniciam com os mais compreendidos.

## 9.4.3 Métodos de Prototipação

---

- **Estilo de prototipagem:**
  - Refere-se as várias abordagens para desenvolver protótipos, que podem ser descartados
  - Utilizados com objetivo de explorar, refinar e validar requisitos e design.
  - Cada estilo utiliza diferentes processos para atingir resultados específicos.

## 9.4.3 Métodos de Prototipação

---

- **Alvo de Prototipagem:**
  - Produto específico atendido pelo esforço de prototipagem
  - Exemplos de alvos incluem uma especificação de requisitos, componente de design, algoritmo ou interface.

## 9.4.3 Métodos de Prototipação

---

- **Técnicas de Avaliação:**
  - Engenheiro de software ou outros stakeholders
  - Protótipos podem ser avaliados em relação ao software implementado ou conjunto de requisitos

## 9.4.4 Métodos Ágeis

---

- Surgiram em 1990
- Reduzir o grande overhead
- Considerados métodos leves
- Com ciclos de desenvolvimento curtos e iterativos



## 9.4.4 Métodos Ágeis

---

- **RAD (Desenvolvimento Rápido de Aplicações):**
  - Usado em desenvolvimento de sistemas de informação intensivos de dados, por engenheiros.
  - Desenvolver, testar e implantar rapidamente aplicativos empresariais.

## 9.4.4 Métodos Ágeis

---

### **XP (Programação Extrema):**

- Abordagem de desenvolvimento que foca em colaboração direta com o cliente, desenvolvimento iterativo e alta qualidade.
- Requisitos são definidos através de histórias
- Testes são escritos antes do código
- O código é continuamente melhorado

## 9.4.4 Métodos Ágeis

---

- **Scrum:**

- Focado no gerenciamento de projetos
- Gerencia atividades dentro do incremento do projeto
- Chamado de sprint e dura no máximo 30 dias
- Uma lista de itens é desenvolvida a partir das tarefas identificadas, definidas e priorizadas
- Uma versão funcional é testada e lançada em cada incremento

## 9.4.4 Métodos Ágeis

---

- **FDD (Desenvolvimento Dirigido por Funcionalidades):**
  - Curto, iterativo e orientado a modelos

Contém 5 fases:

1. Desenvolver um modelo de produto para delimitar a amplitude do domínio
2. Criar a lista de necessidades ou funcionalidades,
3. Construir o plano de desenvolvimento de funcionalidades,
4. Desenvolver projetos para funcionalidades específicas da iteração
5. Codificar, testar e depois integrar as funcionalidades.



**Rômulo Souza Fernandes**

**Ciência da Computação**

**UENF-CCT-LCMAT**

**Campos, RJ**

**00119110559@pq.uenf.br**