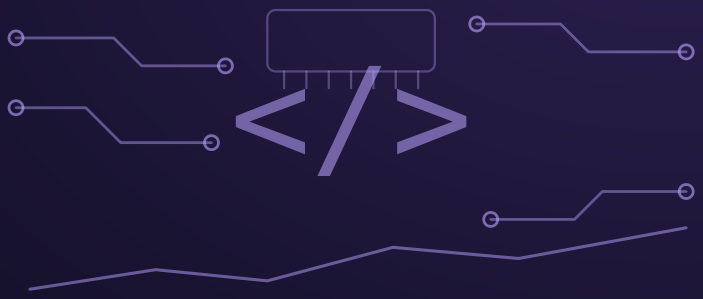
DESENVOLVIMENTO DE SISTEMAS

SÉRIE BANCÁRIA · ANALISTAS

COORDENAÇÃO E REVISÃO

Emerson Braga de Souza



PREPARATÓRIO PARA ANALISTA

Concurso Banestes e Banescor 2027

CURSO LIVRE ICEAD

Material do aluno

Capítulo 1 · Processo e engenharia de software: modelos de maturidade, UML, BPMN, métodos ágeis, requisitos, qualidade e testes

Desenvolvimento de Sistemas · capítulo 1 de 10

Edital nº 01/2027 · prova em 21/02/2027
Atualizado até 20/11/2026

ICEAD
INSTITUTO DE CURSOS EAD

EXPEDIENTE

Curso Preparatório para Analista – Concurso Banestes e Banescor 2027

Capítulo 1 · Processo e engenharia de software: modelos de maturidade, UML, BPMN, métodos ágeis, requisitos, qualidade e testes

OBRA COLETIVA

Organizada e publicada pela ICEAD – Instituto de Cursos EAD (Lei nº 9.610/1998, art. 5º, VIII, h, e art. 17)

COORDENAÇÃO E REVISÃO

Emerson Braga de Souza, administrador (CRA-SP) e contador (CRC-SP)

ELABORAÇÃO

Equipe editorial da ICEAD, com apoio de ferramentas de inteligência artificial

COMO ESTE MATERIAL FOI FEITO

Cada capítulo foi escrito a partir das fontes indicadas abaixo e no fim de cada capítulo, conferido por uma segunda revisão independente (conteúdo, contas e gabaritos) e aprovada pela coordenação antes da publicação. As questões são inéditas, escritas para este curso; nenhuma reproduz questão de prova anterior.

FONTES PRINCIPAIS

Edital nº 01/2027 de Abertura de Inscrições do Banestes (concurso para Analistas, organizado pela Fundação Carlos Chagas), Anexo IV (conteúdo programático); normas técnicas (ABNT, ISO/IEC, NIST, IEEE, IETF, W3C), guias (COBIT, ITIL, PMBOK, TOGAF, Scrum) e documentação oficial das tecnologias citadas em cada capítulo; Lei nº 13.709/2018 e Lei nº 14.133/2021.

EDIÇÃO

Versão de 20/11/2026. Quando houver correção ou atualização, a versão nova fica disponível na Sala de aula.

ENCONTROU UM ERRO?

Escreva para contato@icead.com.br. Toda correção é registrada e entra na próxima versão.

Curso livre da ICEAD, independente, sem vínculo com o Banestes, com a Banescor nem com a Fundação Carlos Chagas. O concurso e as suas regras são definidos só pelo edital oficial e pelas suas retificações.

AVISO LEGAL

Direitos autorais e licença de uso

© 2026 ICEAD, marca de Arizona Marcas e Patentes Ltda. Todos os direitos reservados.

Este material é protegido pela Lei nº 9.610/1998 (Direitos Autorais). A compra dá ao aluno uma **licença de uso pessoal, individual e intransferível**. É proibido reproduzir, distribuir, revender, emprestar ou disponibilizar este arquivo, no todo ou em parte, por qualquer meio.

O compartilhamento não autorizado e a pirataria configuram **crime de violação de direito autoral (Código Penal, art. 184)**, sem prejuízo da responsabilidade civil por perdas e danos.

Exemplar licenciado ao aluno identificado no ato da compra. Cada arquivo pode conter marcação, visível ou oculta, que permite rastrear a origem de cópias irregulares.

Dúvidas: contato@icead.com.br

Sumário

Capítulo 1. Processo e engenharia de software: modelos de maturidade, UML, BPMN, métodos ágeis, requisitos, qualidade e testes

1. Modelos de processo e de maturidade
2. Engenharia de requisitos
3. UML 2.5 e BPMN
4. Métodos ágeis
5. Low-code e no-code
6. Qualidade de software: ISO/IEC 25010 (revisão de 2023, adotada pela ABNT em 2024)
7. Testes de software
8. Gestão da configuração, versionamento semântico e revisão de código
9. Para a prova
10. Treino do capítulo (estilo FCC)

Fontes

Desenvolvimento de Sistemas

Capítulo 1. Processo e engenharia de software: modelos de maturidade, UML, BPMN, métodos ágeis, requisitos, qualidade e testes

> **O que o edital pede (Edital nº 01/2027, Anexo IV, cargo 04):** "Processo de desenvolvimento de software: CMMI-DEV v2.0; ABNT NBR ISO/IEC/IEEE 12207:2021; MR-MPS-SW versão 2023; UML 2.5; BPMN; métodos ágeis (Scrum, Kanban, XP e similares); engenharia de requisitos; engenharia de software; modelagem e especificação de processos e sistemas; desenvolvimento low-code e no-code; [...] qualidade de software segundo o modelo ISO/IEC 25010:2024; tipos de testes de software [...]; gestão da configuração de software; versionamento semântico; revisão de código (code review)."

A prova de Conhecimentos Específicos tem **50 questões com peso 2** (item 9.1 do edital): cada acerto aqui vale o dobro de um acerto em Conhecimentos Gerais. A FCC cobra engenharia de software com **definições precisas** (qual diagrama UML mostra o quê, qual cerimônia do Scrum tem qual objetivo, qual característica de qualidade é qual) e com pequenos casos ("a equipe fez X; trata-se de...").

1. Modelos de processo e de maturidade

1.1 Ciclos de vida

- **Cascata:** fases sequenciais (requisitos, projeto, implementação, testes, implantação). Previsível, mas a mudança custa caro e o cliente vê o produto tarde.
- **Incremental:** o produto é entregue em partes funcionais; cada incremento soma funcionalidades.
- **Iterativo e espiral:** ciclos repetidos de planejamento, análise de **riscos**, engenharia e avaliação (o modelo espiral, de Boehm, é guiado por riscos).
- **Ágil:** iterações curtas, entrega frequente de software funcionando e adaptação contínua.

1.2 CMMI-DEV v2.0

O CMMI (Capability Maturity Model Integration), mantido hoje pelo ISACA, organiza boas práticas em **áreas de prática**. Na versão 2.0:

- **Níveis de maturidade** (representação por estágios) de 0 a 5: 0 incompleto, 1 inicial, 2 gerenciado, 3 definido, 4 gerenciado quantitativamente, 5 em otimização.
- **Níveis de capacidade** (por área de prática) de 0 a 3.
- A avaliação pelo método oficial (SCAMPI na versão 1.3; método de avaliação próprio na 2.0) atribui o nível.

Pegadinha frequente: no nível **2**, os projetos são gerenciados (planejados, monitorados); no nível **3**, a organização tem **processos padrão** que os projetos adaptam; nos níveis **4 e 5**, entram o controle **estatístico** e a melhoria contínua com base em dados.

1.3 ABNT NBR ISO/IEC/IEEE 12207:2021

Norma de **processos do ciclo de vida de software**. Não impõe um modelo de ciclo de vida; descreve processos agrupados em quatro grupos:

1. **Processos de acordo** (aquisição e fornecimento);
2. **Processos organizacionais de habilitação de projetos** (gestão do modelo de ciclo de vida, de infraestrutura, de portfólio, de recursos humanos, de qualidade e de conhecimento);
3. **Processos de gestão técnica** (planejamento, avaliação e controle de projeto, decisão, riscos, configuração, informação, medição, garantia da qualidade);
4. **Processos técnicos** (definição do negócio ou missão, necessidades e requisitos das partes interessadas, requisitos do sistema/software, arquitetura, projeto, implementação, integração, verificação, transição, validação, operação, manutenção e descarte).

Diferença clássica: **verificação** responde "estamos construindo o produto **corretamente**?" (conforme a especificação); **validação** responde "estamos construindo o produto **certo**?" (atende ao uso pretendido).

1.4 MR-MPS-SW (versão 2023)

Modelo de Referência do **MPS.BR** (Softex) para software, compatível com as normas ISO e com o CMMI, pensado para empresas brasileiras, inclusive pequenas. Tem **sete níveis de maturidade**, do **G** (parcialmente gerenciado, o mais baixo) ao **A** (em otimização, o mais alto): G, F, E, D, C, B, A. A avaliação usa o MA-MPS (método de avaliação).

2. Engenharia de requisitos

- **Requisitos funcionais:** o que o sistema faz ("emitir extrato do mês").
- **Requisitos não funcionais:** restrições e qualidades (desempenho, segurança, usabilidade, disponibilidade: "responder em até 2 segundos").
- **Regras de negócio:** políticas do domínio ("cliente menor de idade não contrata cartão de crédito").
- **Atividades:** elicitação (entrevistas, workshops, observação, protótipos, *brainstorming*), análise e negociação, especificação, **validação** (revisões, protótipos) e **gerência de requisitos** (mudanças e **rastreabilidade**).
- **Rastreabilidade:** ligar cada requisito à sua origem, ao projeto, ao código e aos testes, nos dois sentidos (para frente e para trás). A **matriz de rastreabilidade** mostra o impacto de uma mudança.
- **Histórias de usuário** (ágil): "Como **cliente**, quero **consultar o saldo pelo aplicativo** para **evitar ir à agência**", com **critérios de aceitação**. O acrônimo **INVEST** resume a boa história: Independente, Negociável, Valiosa, Estimável, Pequena (*Small*) e Testável.

Dicionário de dados e documentação técnica: descrevem cada dado (nome, tipo, tamanho, domínio, obrigatoriedade, origem), mantendo a documentação junto do código (por exemplo, em arquivos de texto versionados) e atualizada a cada mudança.

3. UML 2.5 e BPMN

3.1 Diagramas UML

A UML 2.5 tem **14 diagramas**, em dois grupos:

Estruturais (o que o sistema)	Comportamentais (o que o sistema)
Classes, Objetos, Componentes, Implantação, Pacotes, Estrutura composta, Perfil	Casos de uso, Atividades, Máquina de estados; e os de **interação**: Sequência, Comunicação, Visão geral de interação, Tempo

Pontos que caem:

- **Casos de uso:** relacionamento **«include»** é obrigatório (o caso base sempre executa o incluído); **«extend»** é opcional/condicional (o caso que estende só acontece em certa condição). **Generalização** entre atores ou casos de uso.
- **Classes:** associação, **agregação** (losango vazio: a parte existe sem o todo) e **composição** (losango cheio: a parte morre com o todo), generalização (herança) e dependência. Visibilidade: **+** pública, **–** privada, **#** protegida, **~** de pacote.
- **Sequência:** troca de mensagens ao longo do tempo, com linhas de vida e fragmentos combinados (**alt**, **opt**, **loop**, **par**).
- **Máquina de estados:** estados de um objeto e transições disparadas por eventos (ex.: conta aberta, bloqueada, encerrada).
- **Implantação:** nós físicos (servidores, dispositivos) e os artefatos instalados neles.

3.2 BPMN

O BPMN (Business Process Model and Notation, OMG, versão 2.0) modela processos de negócio:

- **Eventos** (círculos): início (borda fina), intermediário (borda dupla), fim (borda grossa).
- **Atividades** (retângulos arredondados): tarefa ou subprocesso.
- **Gateways** (losangos): **exclusivo** (XOR, um caminho), **paralelo** (AND, todos os caminhos), **inclusivo** (OR, um ou mais), baseado em eventos.
- **Piscinas** (*pools*) representam participantes; **raias** (*lanes*) dividem responsabilidades dentro de um participante.
- **Fluxo de sequência** (linha cheia) liga elementos dentro da mesma piscina; **fluxo de mensagem** (linha tracejada) liga piscinas diferentes.

4. Métodos ágeis

Manifesto Ágil (2001): indivíduos e interações mais que processos e ferramentas; software em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; responder a mudanças mais que seguir um plano. Os itens da direita têm valor, mas os da esquerda valem mais.

4.1 Scrum (Guia do Scrum de 2020)

- **Responsabilidades (papéis): Product Owner** (maximiza o valor do produto e gerencia o Product Backlog), **Scrum Master** (responsável pela eficácia do time e pela adoção do Scrum) e **Developers** (criam o incremento). O time é pequeno, em geral **10 pessoas ou menos**, sem sub-times nem hierarquia.
- **Eventos:** a **Sprint** (até **um mês**, contém todos os outros), **Sprint Planning** (até 8 horas para Sprint de um mês: por quê, o quê e como), **Daily Scrum** (**15 minutos**, para os Developers inspecionarem o progresso rumo à Meta da Sprint), **Sprint Review** (até 4 horas: inspeciona o resultado com as partes interessadas) e **Sprint Retrospective** (até 3 horas: melhora a qualidade e a eficácia).
- **Artefatos e compromissos:** Product Backlog → **Meta do Produto**; Sprint Backlog → **Meta da Sprint**; Incremento → **Definição de Pronto** (*Definition of Done*).

4.2 Kanban e XP

- **Kanban:** visualizar o fluxo (quadro), **limitar o trabalho em progresso** (WIP), gerenciar o fluxo e medir *lead time* e *cycle time*; não prescreve iterações nem papéis.
- **XP (Extreme Programming):** valores (comunicação, simplicidade, *feedback*, coragem e respeito) e práticas como **programação em par**, **TDD**, **integração contínua**, **refatoração**, *design* simples, propriedade coletiva do código, ritmo sustentável e pequenas entregas.

5. Low-code e no-code

Plataformas que permitem criar aplicações com componentes visuais e pouca ou nenhuma programação. Ganham **velocidade** e aproximam as áreas de negócio (*citizen developers*); os riscos são **dependência do fornecedor** (*lock-in*), limites de personalização e desempenho e **governança** (aplicações "fantasmas", dados espalhados, falhas de segurança). Por isso exigem catálogo, padrões e controle de acesso.

6. Qualidade de software: ISO/IEC 25010 (revisão de 2023, adotada pela ABNT em 2024)

O modelo de qualidade do produto tem **nove características**:

Característica	Ideia
Adequação funcional	faz o que deve, completo, correto e apropriado
Eficiência de desempenho	tempo, uso de recursos, capacidade
Compatibilidade	coexistência e interoperabilidade