





UNIP
UNIVERSIDADE PAULISTA

Roteiros

Análise e Projeto de Sistemas

  <p>Instituto de Ciências Exatas e Tecnologia</p>	<p>Disciplina: Análise e Projeto de Sistemas</p> <p>Título da Aula: Introdução à Análise e Projeto de Sistemas</p>	<p>ROTEIRO 1</p>
---	--	-------------------------

1. Objetivos da Aula

- Compreender o ciclo de vida do desenvolvimento de software e suas fases principais.
- Diferenciar metodologias tradicionais (cascata) e ágeis (Scrum/XP).
- Desenvolver um fluxograma representando o ciclo de vida de um sistema real.
- Relacionar conceitos teóricos com um exemplo prático.

2. Recursos Necessários

- Computadores com acesso à internet.
- Ferramentas gratuitas para diagramas (Draw.io, Lucidchart) ou Astah.
- Material de apoio: Capítulo 1 do livro-texto.
- Editor de texto para o relatório final.

3. Estrutura da Aula

1. Abertura (10 minutos): Apresentar conceitos básicos e discutir exemplos.
2. Revisão Conceitual (20 minutos): Explicar ciclo de vida e metodologias.
3. Demonstração (20 minutos): Criar um fluxograma do ciclo de vida.
4. Atividade Prática (40 minutos): Cada aluno escolhe um sistema real, descreve o ciclo de vida e cria um fluxograma.
5. Encerramento (20 minutos): Discussão das dificuldades e orientações sobre o relatório.

4. Relatório Final

O relatório deve conter:

- Resumo teórico (definição de ciclo de vida, diferenças entre cascata e ágil).
- Estudo de caso do sistema escolhido.
- Diagrama do ciclo de vida (fluxograma).
- Reflexões finais e referências.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade do diagrama (3,0).
- Conexão entre teoria e prática (3,0).
- Criatividade e melhorias (2,0).

6. Conclusão

Ao final desta prática, o estudante deverá ser capaz de diferenciar abordagens de desenvolvimento e representar o ciclo de vida de software em um diagrama, além de produzir um relatório objetivo.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas
Título da Aula: Levantamento de Requisitos

ROTEIRO 2

1. Objetivos da Aula

- Compreender a importância do levantamento de requisitos em projetos de software.
- Aplicar técnicas de elicitação, como entrevistas, prototipação e workshops.
- Desenvolver um conjunto de user stories ou casos de uso para um sistema exemplo.
- Criar um diagrama simples de caso de uso utilizando ferramenta gratuita (Draw.io ou Astah).

2. Recursos Necessários

- Computadores com acesso à internet.
- Ferramentas para diagramas (Draw.io, Lucidchart ou Astah).
- Material de apoio: Capítulo 2 do livro-texto.
- Editor de texto para o relatório final.

3. Estrutura da Aula

1. Abertura (10 minutos): Apresentar conceitos de requisitos funcionais e não funcionais.
2. Revisão Conceitual (20 minutos): Técnicas de elicitação e documentação de requisitos.
3. Demonstração (20 minutos): Exemplo de user stories e criação de diagrama de caso de uso.
4. Atividade Prática (40 minutos): Cada aluno deve definir 5 requisitos funcionais e não funcionais para um sistema fictício e construir um diagrama de caso de uso.
5. Encerramento (20 minutos): Discussão dos resultados e orientações sobre o relatório.

4. Relatório Final

O relatório deve conter:

- Resumo teórico sobre requisitos e técnicas de elicitação.
- Lista de requisitos funcionais e não funcionais levantados.
- Diagrama de caso de uso.
- Reflexões sobre o processo e dificuldades encontradas.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade da lista de requisitos (2,0).
- Coerência do diagrama de caso de uso (3,0).
- Criatividade e melhorias (3,0).

6. Conclusão

Ao final desta prática, o estudante deverá ser capaz de levantar e documentar requisitos básicos, criando artefatos como user stories e diagramas de caso de uso.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas

Título da Aula: Modelagem de Sistemas com UML

ROTEIRO 3

1. Objetivos da Aula

- Introduzir os conceitos fundamentais da UML e seus principais diagramas.
- Exercitar a criação de diagramas de caso de uso, classes e sequência.
- Utilizar ferramentas gratuitas (Draw.io, Lucidchart ou Astah) para a modelagem de um sistema exemplo.
- Relacionar os modelos criados aos requisitos levantados na aula anterior.

2. Recursos Necessários

- Computadores com acesso à internet.
- Ferramentas para diagramas (Draw.io, Lucidchart ou Astah).
- Material de apoio: Capítulo 3 do livro-texto.
- Editor de texto para o relatório final.

3. Estrutura da Aula

1. Abertura (10 minutos): Apresentação da UML e seus principais diagramas.
2. Revisão Conceitual (20 minutos): Explicação dos diagramas de caso de uso, classes e sequência.
3. Demonstração (20 minutos): Exemplo de modelagem de um sistema simples.

4. Atividade Prática (40 minutos): Cada aluno modelará um sistema fictício com pelo menos:
 - 1 diagrama de caso de uso,
 - 1 diagrama de classes,
 - 1 diagrama de sequência.
5. Encerramento (20 minutos): Apresentação dos trabalhos e discussão.

4. Relatório Final

O relatório deve conter:

- Breve resumo sobre a importância da UML e seus diagramas.
- Diagramas criados (caso de uso, classes e sequência).
- Descrição do sistema modelado.
- Reflexões sobre o processo de modelagem.

5. Critérios de Avaliação

- Clareza e completude dos diagramas (4,0).
- Conexão com requisitos levantados anteriormente (2,0).
- Qualidade da descrição do sistema (2,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante deverá compreender os conceitos básicos de modelagem com UML e ser capaz de criar diagramas para representar diferentes aspectos de um sistema.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas

Título da Aula: Design de Software

ROTEIRO 4

1. Objetivos da Aula

- Compreender os principais padrões de design de software (Factory, Singleton, Observer, Strategy etc.).
- Desenvolver exemplos práticos desses padrões nas linguagens Python ou C.
- Explorar conceitos de arquitetura multicamadas.
- Criar diagramas UML que representem a aplicação dos padrões em um sistema exemplo.

2. Recursos Necessários

- Computadores com ambiente configurado para Python e/ou C.
- Ferramentas de modelagem (Draw.io, Astah ou Lucidchart).
- Editor de código (VS Code, Visual Studio, Code::Blocks ou similar).
- Material de apoio: Capítulo 4 do livro-texto.

3. Estrutura da Aula

1. Abertura (10 minutos): Revisão dos conceitos de padrões de projeto.
2. Revisão Conceitual (20 minutos): Apresentação de exemplos práticos (Factory, Singleton, Observer).
3. Demonstração (20 minutos): Criação de um pequeno projeto aplicando um padrão.

4. Atividade Prática (40 minutos): Implementar um padrão em Python ou C e criar o diagrama UML correspondente.
5. Encerramento (20 minutos): Apresentação das soluções e discussão em grupo.

4. Relatório Final

O relatório deve conter:

- Resumo teórico dos padrões abordados.
- Código-fonte desenvolvido.
- Diagrama UML representando o padrão aplicado.
- Reflexões sobre o uso dos padrões no desenvolvimento.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade e funcionalidade do código desenvolvido (3,0).
- Correção e clareza do diagrama UML (3,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante será capaz de identificar e aplicar padrões de design em sistemas reais, bem como representar essas soluções em diagramas UML.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas

Título da Aula: Arquitetura de Software

ROTEIRO 5

1. Objetivos da Aula

- Compreender os principais modelos arquiteturais: Monolítico, SOA e Microserviços.
- Desenvolver diagramas UML (componentes e implantação) representando cada modelo.
- Explorar conceitos de escalabilidade, acoplamento e modularização.
- Analisar um caso real e propor uma arquitetura adequada.

2. Recursos Necessários

- Computadores com acesso à internet.
- Ferramentas de modelagem (Draw.io, Astah ou Lucidchart).
- Material de apoio: Capítulo 5 do livro-texto.
- Editor de texto para o relatório final.

3. Estrutura da Aula

1. Abertura (10 minutos): Introdução aos conceitos de arquitetura de software.
2. Revisão Conceitual (20 minutos): Explicação dos modelos arquiteturais e suas características.
3. Demonstração (20 minutos): Exemplo de diagrama de componentes para um sistema web.

4. Atividade Prática (40 minutos): Cada aluno deve modelar uma arquitetura para um sistema fictício, incluindo:
 - Diagrama de componentes,
 - Diagrama de implantação.
5. Encerramento (20 minutos): Discussão sobre vantagens e desvantagens dos modelos.

4. Relatório Final

O relatório deve conter:

- Resumo teórico dos modelos arquiteturais.
- Diagramas criados (componentes e implantação).
- Descrição do sistema e justificativa da arquitetura escolhida.
- Reflexões sobre a escalabilidade e manutenção do sistema.

5. Critérios de Avaliação

- Clareza e detalhamento do resumo teórico (2,0).
- Correção e clareza dos diagramas (4,0).
- Justificativa da arquitetura escolhida (2,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante será capaz de identificar e propor diferentes arquiteturas de software para atender requisitos de sistemas reais.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas
Título da Aula: Ferramentas de Modelagem
e Case

ROTEIRO 6

1. Objetivos da Aula

- Introduzir o conceito de ferramentas CASE (Computer-Aided Software Engineering).
- Explorar recursos básicos de ferramentas como Astah, Draw.io e Visual Paradigm.
- Criar um projeto-modelo com diagramas UML (caso de uso, classes e sequência).
- Demonstrar como gerar código ou documentação automática a partir de modelos.

2. Recursos Necessários

- Computadores com acesso à internet.
- Ferramentas CASE instaladas ou versões online (Astah Community, Draw.io).
- Material de apoio: Capítulo 6 do livro-texto.
- Editor de texto para o relatório final.

3. Estrutura da Aula

1. Abertura (10 minutos): Conceitos e benefícios das ferramentas CASE.
2. Revisão Conceitual (20 minutos): Recursos comuns das ferramentas (modelagem, geração de código).
3. Demonstração (20 minutos): Criar diagramas em uma ferramenta escolhida (Astah ou Draw.io).

4. Atividade Prática (40 minutos): Criar um projeto-modelo com pelo menos:

- Diagrama de caso de uso,
- Diagrama de classes,
- Diagrama de sequência.

5. Encerramento (20 minutos): Apresentação dos projetos e discussão.

4. Relatório Final

O relatório deve conter:



- Resumo teórico sobre ferramentas CASE.
- Diagramas criados (capturas de tela).
- Breve descrição do processo de modelagem.
- Reflexões sobre os benefícios do uso de ferramentas CASE.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade e completude dos diagramas (4,0).
- Coerência da documentação gerada (2,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante será capaz de utilizar ferramentas CASE para modelar sistemas e gerar documentação técnica.

  <p>Instituto de Ciências Exatas e Tecnologia</p>	<p>Disciplina: Análise e Projeto de Sistemas</p> <p>Título da Aula: Avaliação e Validação de Projetos de Sistemas</p>	<p>ROTEIRO 7</p>
---	---	-------------------------

1. Objetivos da Aula

- Compreender técnicas de avaliação colaborativa de projetos de sistemas.
- Aplicar métodos de validação e verificação de requisitos e modelos.
- Desenvolver checklists de avaliação de qualidade para diagramas e documentação.
- Simular uma revisão em grupo para identificar falhas e melhorias.

2. Recursos Necessários

- Computadores com acesso à internet.
- Exemplos de diagramas UML (criados em aulas anteriores por exemplo).
- Ferramentas de colaboração (Google Docs, Trello, Miro ou similar).
- Material de apoio: Capítulo 7 do livro-texto.

3. Estrutura da Aula

1. Abertura (10 minutos): Introdução à importância da avaliação e validação.
2. Revisão Conceitual (20 minutos): Métodos de validação, revisões e inspeções.
3. Demonstração (20 minutos): Exemplo de checklist para avaliação de diagramas UML.
4. Atividade Prática (40 minutos): Cada grupo avalia os diagramas de outro grupo, utilizando o checklist.
5. Encerramento (20 minutos): Apresentação dos pontos fortes e sugestões de melhoria.

4. Relatório Final

O relatório deve conter:

- Resumo teórico sobre avaliação e validação de sistemas.
- Checklist utilizado para a revisão.
- Principais problemas encontrados e sugestões de melhorias.
- Reflexões sobre o processo de avaliação em grupo.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade e aplicabilidade do checklist (3,0).
- Identificação e justificativa das falhas encontradas (3,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante será capaz de avaliar criticamente um projeto de software e propor melhorias baseadas em critérios de qualidade.



**Instituto de Ciências
Exatas e Tecnologia**

Disciplina: Análise e Projeto de Sistemas

Título da Aula: Manutenção e Evolução
de Sistemas

ROTEIRO 8

1 Objetivos da Aula

- Compreender os tipos de manutenção de software (corretiva, adaptativa, perfectiva e preventiva).
- Aplicar técnicas de refatoração para melhorar a qualidade do código.
- Analisar um sistema simples e propor melhorias evolutivas.
- Criar um pequeno exemplo prático de reengenharia.

2. Recursos Necessários

- Computadores com ambiente configurado para Python e/ou C.
- Editor de código (VS Code, Visual Studio, Code::Blocks ou similar).
- Material de apoio: Capítulo 8 do livro-texto.
- Ferramentas de versionamento (GitHub, GitLab ou similar).

3. Estrutura da Aula

1. Abertura (10 minutos): Introdução aos conceitos de manutenção e evolução.
2. Revisão Conceitual (20 minutos): Exemplos reais de manutenção de sistemas.
3. Demonstração (20 minutos): Refatorar um código existente para melhorar legibilidade e desempenho.
4. Atividade Prática (40 minutos): Refatorar e documentar um código legado, propondo melhorias.
5. Encerramento (20 minutos): Discussão das melhorias aplicadas e lições aprendidas.

4. Relatório Final

O relatório deve conter:

- Resumo teórico sobre manutenção e evolução de software.
- Código original (antes da refatoração) e código atualizado.
- Descrição das melhorias aplicadas.
- Reflexões sobre o impacto da manutenção no ciclo de vida do software.

5. Critérios de Avaliação

- Clareza do resumo teórico (2,0).
- Qualidade da refatoração e código atualizado (4,0).
- Justificativa das melhorias aplicadas (2,0).
- Organização e apresentação do relatório (2,0).

6. Conclusão

Ao final desta prática, o estudante será capaz de identificar problemas em códigos existentes e aplicar técnicas de manutenção e evolução para melhorar a qualidade do sistema.