MC626 – Verificação, Validação e Testes

(1º semestre 2018)

Prof. Breno Bernard Nicolau de França

breno@ic.unicamp.br

www.ic.unicamp.br/~breno

Dia	Horário
Terça-feira	19h-21h
Quinta-feira	21h-23h
Atendimento (com horário marcado)	Segundas 14h-16h (Breno)

1. Objetivos Terminais

Ao final do curso, o aluno deve ser capaz de:

★ Analisar os requisitos de um sistema de software e decidir sobre as técnicas oportunas para verificá-los e validá-los.

★ Aplicar corretamente técnicas de verificação e validação de software.

2. Conhecimentos Requeridos

➢ Programação modular em linguagem com suporte a definição de interfaces e componentes (módulos);

➢ Análise e Projeto de Software, que compreende a especificação de requisitos, bem como as transformações de requisitos em artefatos de projeto (sobretudo diagramas de transição de estados e classes) e código.

Avaliação Diagnóstica:

Na primeira semana do curso, será aplicada uma avaliação diagnóstica no intuito de identificar o repertório do aluno no que diz respeito ao conjunto de conhecimentos requeridos que se entende como imprescindíveis para um bom desempenho na disciplina.

Nesta oportunidade, o aluno deverá ser capaz de responder corretamente a um questionário de múltipla escolha com dez questões, envolvendo projeto e arquitetura de software, mais especificamente os princípios de abstração, ocultamento de informação, separação de preocupações, e independência funcional. Além disso, o questionário conterá questões de programação que remetem às boas práticas de projeto de software.

3. Unidades

3.1. Conceitos de Verificação e Validação

Objetivo	O aluno deve ser capaz de compreender os conceitos de qualidade de software, verificação e validação, e suas implicações no ciclo de vida do software.
Procedimento	Aula expositiva dialogada[1] e leituras de capítulo[2] do livro.
Avaliação	Lista de questões discursivas. Discussão de possíveis respostas no gabarito. Entrega individual.

3.2. Técnicas de Verificação Estática

Objetivo	O aluno deve ser capaz de analisar os diferentes tipos de técnicas de verificação estática: revisão, inspeção e análise estática de código.
Procedimento	Aula expositiva dialogada e duas leituras/artigos (inspeções e análise estática).
Avaliação	Síntese crítica e comparativa (inspeções vs análise estática) das leituras. Entrega em dupla.

3.3. Fundamentos de Teste de Software

Objetivo	O aluno deve ser capaz de compor uma estratégia de testes a partir de um contexto de projeto, utilizando os conceitos fundamentais de teste de software, dimensões, bem como o processo de testes.
Procedimento	Aula expositiva dialogada, lista de exercícios.
Avaliação	Definir uma estratégia de testes a partir de um cenário hipotético. Mínimo duas páginas explicando a estratégia e justificando as decisões tomadas. Entrega em dupla.

[1] Este modelo de aula expande o conceito da aula expositiva tradicional, trazendo o aluno para uma discussão do conteúdo exposto, podendo incitar questionamentos, análise crítica e confronto com a realidade. Práticas: convidar alunos a responder questões, formar grupos para discutir e apresentar uma visão sobre um tópico; pedir para alunos levantarem questões; discussão de práticas de como o objeto aparece em cenários reais.

[2] Capítulo 22 (Ed. 8) ou Capítulo 24 (Ed. 9). Ian Sommerville. Software Engineering, 8/9 Ed. 2011. Pearson.

3.4. Técnicas de Teste de Software

Objetivo	O aluno deve ser capaz de aplicar as técnicas de testes de software para o projeto de casos de testes.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Adicionar casos de testes relevantes para um software livre existente com base nas seguintes técnicas: Análise de Valor Limite, Particionamento em Classes de Equivalência, Tabela de Decisão, Transição de Estados, e Teste de Caso de Uso. Entrega dos casos de teste codificados e um breve relatório (documento PDF) explicando como os mesmos foram derivados/gerados.

3.5. Testes Não Funcionais

Objetivo	O aluno deve ser capaz de aplicar testes não funcionais para as características: Desempenho (Stress e Carga), Robustez (Injeção de Falhas), e Segurança.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Implementar uma pequena aplicação Web ou Mobile (ou evoluir uma existente) e adicionar casos de testes relevantes com base nas seguintes características: desempenho, robustez, e segurança. Entrega dos casos de teste codificados e um breve relatório (documento PDF) explicando como os mesmos foram derivados/gerados.

3.6. Test-Driven e Behavior-Driven Development

Objetivo	O aluno deve ser capaz de aplicar as práticas de TDD e BDD no desenvolvimento de software.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Desenvolver casos de teste para uma função especificada utilizando a técnica TDD, ou BDD, em laboratório. Entrega dos casos de teste e a função codificados (com comentários) **A presença do aluno nesta avaliação é obrigatória.**

3.7. Teste de Mutação

Objetivo	O aluno deve ser capaz de aplicar técnicas de teste de mutação.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Derivar um conjunto de casos de testes utilizando técnicas de teste de mutação. Entrega dos casos de teste codificados e um relatório (documento PDF) explicando como os mesmos foram derivados/gerados.

3.8. Model-Based Testing

Objetivo	O aluno deve ser capaz de aplicar técnicas de teste dirigidos por modelos.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Derivar um conjunto de casos de testes a partir de modelos de transição de estados e/ou casos de uso. Entrega dos casos de teste codificados e um relatório (documento PDF) explicando como os mesmos foram derivados/gerados.

4. Critérios de Avaliação

A avaliação da disciplina realizada com base em três critérios:

1. **Participação**: este critério é individualizado e representa até 10% da nota final. A atribuição da nota de participação é proporcional e considerada a frequência, envolvimento nas atividades em sala de aula e laboratório, cumprimento de prazos relativos às entregas (exercícios, projetos, etc.), e cumprimento das leituras e vídeos recomendados.
2. **Avaliação de Unidade**: este critério representa até 90% da nota final. As notas serão atribuídas com base no desempenho do aluno para as atividades de avaliação previstas para cada unidade (seção 3). Com exceção da unidade quatro (seção 3.4), que terá peso 2 (20% da nota final), todas as outras unidades terão peso 1 (10% da nota final). As datas das avaliações e prazos de entrega estão definidos no cronograma (seção 5).

4.1. Informações Importantes:

- As datas referentes às entregas, tanto dos projetos quanto dos exercícios, estão disponíveis no cronograma da disciplina.
- A presença é **obrigatória** em todas as aulas (incluindo laboratório). Frequência inferior a 75% causa reprovação.

- Casos de plágio (cópia de texto, imagem ou ideia) entre os trabalhos ou de conteúdos externos serão tratados com rigor. A nota da avaliação em questão será anulada sem possibilidade de reposição e o caso será encaminhado à coordenação do curso.

5. Cronograma

As datas definidas a seguir podem sofrer alterações devido a imprevistos e/ou situações adversas.

Data	**Tópico**
27/02	Apresentação da Disciplina + Avaliação Diagnóstica
01/03	Conceitos de Verificação e Validação
06/03	Conceitos de Verificação e Validação
08/03	Técnicas de Verificação Estática: Revisão e Inspeção de Software
13/03	Entrega Avaliação da Unidade 1
	Técnicas de Verificação Estática: Revisão e Inspeção de Software
15/03	Fundamentos de Teste de Software
20/03	Entrega Avaliação da Unidade 2
	Fundamentos de Teste de Software
22/03	Fundamentos de Teste de Software
27/03	Técnicas de Teste Funcional
29/03	Feriado: Não haverá atividades
03/04	Entrega Avaliação da Unidade 3
	Técnicas de Teste Funcional (Lab)
05/04	Técnicas de Teste Funcional
10/04	Técnicas de Teste Funcional (Lab)
12/04	Técnicas de Teste Estrutural
17/04	Técnicas de Teste Estrutural (Lab)
19/04	Técnicas de Teste Estrutural (Lab)
24/04	Entrega Avaliação da Unidade 4
	Testes Não Funcionais: Desempenho
26/04	Testes Não Funcionais: Desempenho (Lab)

01/05	Feriado: Não haverá atividades
03/05	Testes Não Funcionais: Robustez
08/05	Testes Não Funcionais: Robustez (Lab)
10/05	Testes Não Funcionais: Segurança
15/05	Testes Não Funcionais: Segurança (Lab)
17/05	TDD
22/05	Entrega Avaliação da Unidade 5
	TDD (Lab)
24/05	BDD
29/05	BDD (Lab)
31/05	Feriado: Não haverá atividades
05/06	Avaliação da Unidade 6 (Lab)
07/06	Teste de Mutação
12/06	Teste de Mutação (Lab)
14/06	Teste Baseado em Modelos
19/06	Entrega Avaliação da Unidade 7
	Teste Baseado em Modelos
21/06	Teste Baseado em Modelos (Lab)
26/06	Entrega Avaliação da Unidade 8
28/06	Apresentação de Trabalhos
10/07	Exame

6. Bibliografia

O curso é baseado nos seguintes livros texto, ou edições mais novas dos mesmos. Qualquer material adicional de leitura será anunciado, em sala, quando necessário.

- Sommerville, I. (2007). *Engenharia de Software*, 8ª edição. São Paulo: Pearson Addison-Wesley, 22, 103.
- Delamaro, M., Jino, M., & Maldonado, J. (2017). *Introdução ao teste de software*. Elsevier Brasil.

- Ammann, P., & Offutt, J. (2016). *Introduction to software testing*. Cambridge University Press.