MC626 – Verificação, Validação e Testes

(2º semestre 2019)

Prof. Breno Bernard Nicolau de França

breno@ic.unicamp.br

www.ic.unicamp.br/~breno

Dia	Horário
Terça-feira	21h-23h (CB07)
Quinta-feira	19h-21h (CB09)
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 engenharia de requisitos, bem como as transformações de requisitos em artefatos de projeto 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 (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 24. Ian Sommerville. Software Engineering, 9 Ed. 2011. Pearson.

3.4. 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 uma função especificada e os casos de teste para a mesma utilizando a técnica TDD em laboratório. Entrega dos casos de teste e a função codificados (com comentários) **A presença do aluno nesta avaliação é obrigatória.**

3.5. 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 em pelo menos três das 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.

3.6. Testes Não-Funcionais

Objetivo	O aluno deve ser capaz de aplicar testes não funcionais para as características: Desempenho (Estresse 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.

3.7. 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. Teste Dirigido por Modelos

Objetivo	O aluno deve ser capaz de aplicar técnicas de teste dirigidos por modelos para diagramas de estados e especificações em casos de uso.
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.

3.9. Conceitos Básicos de *Model Checking*

Objetivo	O aluno deve ser capaz de compreender os conceitos básicos envolvidos nas técnicas de verificação de modelos.
Procedimento	Aula expositiva dialogada, laboratórios/exercícios práticos.
Avaliação	Avaliação de um modelo de estados, registrando os problemas encontrados (relatório de incidentes em PDF).

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). 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).

A média (M) da disciplina será calculada como:
$M = 0.1P + \sum^{9}_{i=1} 0.1Ui$

Exame:

- Caso no final da disciplina o aluno tenha média $2.5 \leq M < 5.0$, poderá fazer o exame (E), cuja data está no cronograma.
- Neste caso, a média final será calculada como MF = (M +E)/2, com nota máxima igual a 5.0.

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**
01/08	Apresentação da Disciplina + Avaliação Diagnóstica
06/08	SECOMP: Não haverá aula
08/08	
13/08	Conceitos de Verificação e Validação
15/08	Conceitos de Verificação e Validação
20/08	Técnicas de Verificação Estática: Revisão, Inspeção e Análise Estática de Software
22/08	Entrega Avaliação da Unidade 1
	Técnicas de Verificação Estática: Revisão, Inspeção e Análise Estática de Software
27/08	Fundamentos de Teste de Software
29/08	Entrega Avaliação da Unidade 2
	Fundamentos de Teste de Software

03/09	Fundamentos de Teste de Software
05/09	TDD
10/09	Entrega Avaliação da Unidade 3
	TDD (Lab)
12/09	TDD (Lab)
17/09	BDD
19/09	BDD (Lab)
24/09	Avaliação da Unidade 4
26/09	Técnicas de Teste Funcional
01/10	Técnicas de Teste Funcional (Lab)
03/10	Técnicas de Teste Funcional (Lab)
08/10	Técnicas de Teste Estrutural
10/10	Técnicas de Teste Estrutural (Lab)
15/10	Testes Não Funcionais: Desempenho
17/10	Entrega da Avaliação da Unidade 5
	Testes Não Funcionais: Desempenho (Lab)
22/10	Testes Não Funcionais: Robustez
24/10	Testes Não Funcionais: Segurança
29/10	SBQS: Não haverá aula
31/10	Entrega da Avaliação da Unidade 6
	Testes Não Funcionais: Segurança (Lab)
05/11	Teste de Mutação
07/11	Teste de Mutação (Lab)
12/11	Teste Baseado em Modelos
14/11	Entrega da Avaliação da Unidade 7
	Teste Baseado em Modelos
19/11	Teste Baseado em Modelos (Lab)
21/11	Model checking

26/11	Entrega da Avaliação da Unidade 8
	Model checking
28/11	Trabalhos
03/12	Entrega da Avaliação da Unidade 9
05/12	Trabalhos
10/12	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.