Universidade Federal de Goiás
Instituto de Informática
Engenharia de Software

Matriz Curricular: ENGSO-BN-2 - 2017.1
Plano de Disciplina
Ano Letivo: 2025 - 1º Semestre

Dados da Disciplina

Código	Nome	Carga Horária	
		Teórica	Prática
10000129	Construção de Software	48	80

Prof(a): Fabio Nogueira de Lucena

Turma: A

Ementa

1. Visão geral (4h): construção (minimizar complexidade, antecipação de mudança, verificação, padrões), projeto de software (software design), qualidade de produto. 2. Planejamento (8h): linguagens de programação e processos de construção. 3. Gerência de construção (16h): controle de versão, inspeção e revisão de código. 4. Fundamentos de codificação (32h): estratégias recomendadas para criar código, variáveis, classes, interfaces, polimorfismo, rotinas, recursão, condições, laços, tratamento de exceção, reflexão, programação defensiva, padrão de codificação (leiaute e estilo), documentação, ferramentas de programação. 5. Projeto (design) detalhado e codificação (32h): noções de projeto detalhado, especificação de projeto, análise sintática (parsing), expressões regulares, parametrização (generics), closure, logging, configuração de software em tempo de execução. Internacionalização. Técnicas de construção baseadas em estado e tabelas. 6. Refatoração (8h). 7. Testes de unidade (16h). 8. Detecção e remoção de defeitos (debugging) (8h). 9. Integração (4h): integração contínua.

Objetivo Geral

Aprimorar técnicas de programação, desenvolver habilidades para escrita de código legível, manutenível, além de percepção de software design (arquitetura de software).

Objetivos Específicos

O estudante deverá ser capaz ao final da disciplina de:
- conhecer os pré-requisitos necessários para construção de software de qualidade;
- elaborar códigos-fontes seguindo boas práticas de programação;
- dividir os códigos-fontes em pacotes conforme sua especialidade;
- utilizar ferramentas de controle de versão durante o processo de construção de software;
- conhecer e fazer uso de ferramentas de desenvolvimento de software;
- construir software bem documentado.

Relação com Outras Disciplinas

Para que o aluno tenha um bom desempenho na disciplina é necessário que tenha sólidos conhecimentos do conteúdo das disciplinas precursoras: Introdução à programação, Algoritmos e Estruturas de Dados (1 e 2), Programação orientada a objetos, Linguagens e Paradigmas de Programação, Engenharia de Software, Análise e Projeto de Algoritmos, Banco de Dados e Projeto de Software. E a consolidação do Conhecimento do conteúdo desta disciplina é condição necessária para um bom desempenho nas disciplinas do 5º período, bom como naquelas componentes do períodos seguintes. Esta relação está associada ao fato que esta disciplina utiliza do conhecimento das precursoras e introduz, de forma genérica, conceitos que serão especializados nas sucessoras.

Programa

TODAS AS ATIVIDADES DO PROGRAMA serão apresentadas no contexto da realização do trabalho prático do grupo. Isso significa que não haverá, exceto se detectado necessário, uma exposição dirigida

para testes de unidade, por exemplo. Em vez de definir uma agenda rígida dos tópicos, todos eles serão tratados no contexto da construção de uma aplicação prática realizada pelo grupo em questão. Ou seja, o estudante, membro de um grupo, terá a oportunidade de esclarecer cada um dos tópicos da disciplina, no contexto do trabalho que o grupo escolheu, ou seja, de forma "prática", "concreta" e alinhada com o trabalho em questão. Naturalmente, a cada encontro com o docente haverá, necessariamente, sem exceção, orientação do docente sobre o que se espera do encontro, o que já era para ter sido feito e o que é esperado para o futuro próximo em termos de atividade do grupo.

Visão Geral – Apresentação da disciplina e discussão de conceitos associados à construção de software, tais como: minimização complexidade, antecipação de mudança, verificação/validação, padrões), projeto de software (software design), qualidade de processo/produto. Apresentação das aplicações (trabalho prático de construção).
Planejamento:linguagens de programação e processos de construção.
Gerência de construção: controle de versão, inspeção e revisão de código.
Fundamentos de codificação: estratégias recomendadas para criar código, variáveis, classes, interfaces, polimorfismo, rotinas, recursão, condições, laços, tratamento de exceção, reflexão, programação defensiva, padrão de codificação (leiaute e estilo), documentação, ferramentas de programação.
Projeto (design) detalhado e codificação: noções de projeto detalhado, especificação de projeto, análise sintática (parsing), expressões regulares, parametrização (generics), closure, logging, configuração de software em tempo de execução. Internacionalização. Técnicas de construção baseadas em estado e tabelas.
Refatoração
Testes de unidade
Detecção e remoção de defeitos (debugging).
Integração: integração contínua.

Procedimentos Didáticos

Legenda	Descrição	Objetivo
AEX	Aula teórica	Transmitir conhecimento utilizando quadro ou slides.
AP	Aula prática	Proporcionar ao aluno a aplicação prática do conteúdo ministrado em aula teórica.
ED	Estudo dirigido	Desenvolver a capacidade analítica, capacidade de síntese, de avaliação crítica e de análise.
OTR	Outros	Transmitir conhecimento utilizando quadro ou slides.
RE	Aula teórica com resolução de exercícios	Desenvolver o raciocínio lógico, criatividade e capacidade de abstração e a capacidade de identificar, analisar e projetar soluções de problemas.
SE	Seminários	Desenvolver o raciocínio lógico, criatividade, capacidade de abstração, capacidade para identificar, analisar, projetar soluções de problemas, a capacidade de comunicação oral e a capacidade de trabalhar em grupo.
TG	Trabalho em grupo	Desenvolver a capacidade de comunicação oral e escrita. Capacidade de trabalhar em grupo.

Conteúdo Programático / Cronograma

Inicio	Proc. Didático	Tópico	# Aul.
07/03/25	AEX, OTR	Visão Geral (4h) – Apresentação da disciplina e discussão de conceitos associados à construção de software, tais como: minimização complexidade, antecipação de mudança, verificação/validação, padrões), projeto de software (software design), qualidade de processo/produto. ATIVIDADE SUPERVISIONADA (baseia-se na identificação de questões relevantes do contexto de construção de sofware para serem esclarecidas com o docente a cada encontro).	4
12/03/25	AEX, RE, OTR	Apresentação dos trabalhos práticos (aplicações). SPRINT 0 (identificação do trabalho a ser realizado pelo grupo). Atividade Supervisionada.	8
19/03/25	AP	SPRINT 1 (esclarecimento de dúvidas sobre a aplicação escolhida pelo grupo). Atividade supervisionada.	4
21/03/25	AEX, AP, TG, OTR	SPRINT 2 (proposta de plano de construção). Ver modelo sugerido pelo docente. Atividade Supervisionada.	4
26/03/25	AEX, AP, OTR	Abordagem de vários tópicos pertinentes à construção e experimentação com tecnologias alinhadas com as necessidades da aplicação do trabalho em grupo.	12

Inicio	Proc. Didático	Tópico	# Aul.
04/04/25	AEX, RE, TG, OTR	SPRINT 3 (primeira iteração com código operacional que exercita a arquitetura proposta ou outro componente eleito para tal compatível com o plano). Isso significa envolvimento com fundamentos de codificação: estratégias recomendadas para criar código, variáveis, classes, interfaces, polimorfismo, rotinas, recursão, condições, laços, tratamento de exceção, reflexão, programação defensiva, padrão de codificação (leiaute e estilo), documentação, ferramentas de programação para testes de unidade e integração contínua, dentre outras. Atividade Supervisionada.	12
16/04/25	AEX, AP, OTR	SPRINT 4 (a arquitetura da aplicação deve ser concluída, versão completa, mas não necessariamente definitiva). Atividade supervisionada.	12
25/04/25	AEX, RE, TG, OTR	SPRINT 5 (40% da aplicação implementada). Isso não significa que o código apresentado é definitivo, pois pode exigir refatoração, por exemplo, visando eliminar débito técnico. Neste ponto já devem ter sido abordados de forma prática, o que varia conforme a aplicação, vários temas como projeto detalhado, especificação de projeto, análise sintática (parsing), expressões regulares, parametrização (generics), closure, logging, configuração de software em tempo de execução. Internacionalização. Técnicas de construção baseadas em estado e tabelas. Atividade Supervisionada.	12
07/05/25	AEX, RE, TG, OTR	SPRINT 6 (60% da aplicação implementada). Refatoração, testes de unidade e débito técnico são atividades esperadas. Atividade Supervisionada.	12
16/05/25	AEX, RE, TG, OTR	Abordagem dos tópicos da disciplina e relação com o trabalho prático. Oportunidade para esclarecer dúvidas sobre o trabalho, eventuais dúvidas que persistam sobre as avaliações (critérios e outros).	4
21/05/25	AEX, RE, TG, OTR	SPRINT 7 (80% da aplicação implementada). Atividade Supervisionada.	12
30/05/25	AEX, RE, TG, OTR	SPRINT 8 (90% da implementação realizada). Integração contínua implementada e com resultados esperados da análise estática de código conforme critérios anteriormente estabelecidos. Atividade Supervisionada.	12
11/06/25	AEX, AP, OTR	Revisão de tópicos e dúvidas gerais sobre o conteúdo da disciplina (inclusive avaliações, critérios e outros).	4
13/06/25	AEX, AP, OTR	SPRING 9 (100% da aplicação concluída). Refatoração e outros ajustes na documentação são esperados. Atividade supervisionada.	12
25/06/25	OTR	Conclusão da disciplina. Prazo final para últimos ajustes na aplicação. Nota final disponibilizada.	4
		Total	128

Critério de Avaliação

Composição da Carga Horária da Disciplina.
Teórica 40hs
Prática 88 Hs.
As atividades da disciplina serão realizadas, todas elas, em laboratório. Em particular, as práticas serão realizadas por meio do desenvolvimento de uma aplicação (software). As aplicações a serem desenvolvidas serão apresentadas no primeiro encontro. Cada grupo fará uma das aplicações e será composta por pelo menos 3 membros e no máximo 5. Todos os membros deverão estar aptos a responder questões sobre qualquer artefato do software construído. As atividades supervisionadas também serão realizadas em grupo e no escopo da aplicação a ser desenvolvida pelo grupo. A construção será realizada em Sprints com duração de 2 semanas cada uma. As avaliações serão realizadas sobre os entregáveis produzidos pelas Sprints.

A Nota Final será a média aritmética simples das avaliações das sprints.

Observações:
1. Estará aprovado o aluno que atingir média (NF) igual ou superior a 6.0 e frequência igual ou superior a 75% da carga horária da disciplina. A frequência será aferida durante as aulas presenciais.
2. Todas as atividades deverão ser registradas no repositório do github criado especificamente para esta finalidade e do qual todos os membros do grupo em questão devem fazer parte e contribuir.

Convém ressaltar que a avaliação será realizada sobre os artefatos presentes no repositório, que também registrará as ações individuais de cada membro.
3. Período para solicitar segunda chamada, até 7 dias após a data da aplicação da atividade avaliativa.
4. Período para solicitar revisão de nota, até 7 dias após a data da entrega da nota.
5. Será atribuída a nota 0,0 (zero) a qualquer atividade ou trabalho não realizado ou não registrada no repositório em questão até a data estipulada. Casos excepcionais serão tratados em comum acordo com a docente.
6. Todas as atividades são supervisionadas. As atividades supervisionadas referem-se às atividades práticas e devem ser desenvolvidas segundo Resolução CNE/CES 03/2007 de 2 de julho de 2007, a qual considera que os Bacharelados do período noturno dividem cada hora de atividade acadêmica em 45 minutos de preleções e aulas expositivas e 15 minutos de atividades práticas supervisionadas que podem ser realizadas a distância ou não, mas com supervisão do professor.
7. Os alunos que se envolverem em plágio (desvios de conduta, seja como facilitador ou como beneficiário) receberão nota 0 (zero) para a atividade correspondente. O caso poderá ser levado ao conhecimento da Coordenação do Curso, do Núcleo Docente Estruturante e do Conselho Diretor do Instituto de Informática para as providências cabíveis e legais. O pedido de segunda chamada deverá ser protocolado conforme condições estipuladas na Resolução CONSUNI específica (RGCG) em vigor.
8. Este Plano está amparado pelas normativas e portarias emanadas dos órgãos governantes superiores, pelas resoluções, instruções normativas e diretrizes didático-pedagógicas da UFG e do INF, em vigor, que definem e regulam o funcionamento do ensino remoto excepcional.
9. Os critérios específicos a serem observados nos objetos de avaliação são cobertos no conteúdo da disciplina e serão apresentados ao longo do curso. Cabe ao estudante esclarecer todo e qualquer elemento que faça parte da avaliação que não seja da sua compreensão.
10. No limite previsto pela Universidade, parte das aulas poderá ser realizada na modalidade não presencial (EAD). Neste caso a frequencia será aferida através da entrega de atividade determinada para a aula remota;
11. É obrigatório o uso de e-mail institucional em comunicações relacionadas à disciplina, conforme a política de comunicação da UFG (Resolução CONSUNI 10/2019).
12. O atendimento extraclasse aos alunos será disponibilizado conforme o quadro de horários afixado na porta da sala do professor.

Data da Realização das Provas

Não haverá provas. A avaliação será aferida, conforme especificado na Seção "Critério de Avaliação".

Local de Divulgação dos Resultados das Avaliações

As avaliações serão constantemente disponibilizadas no SIGAA.

Bibliografia Básica

- MCCONNELL, S. Code Complete: um guia prático para a construção de software 2. a edição. Porto Alegre, RS: Bookman, 2005. ISBN 8536305045. - GOODLIFFE, P. Como ser um programador melhor. Novatec, 2015. ISBN 978-85-7522-415-1. - ORAM, A.; WILSON, G. Beautiful code. O'Reilly, 2007. ISBN 9780596510046.

Bibliografia Complementar

- FOWLER, M. Refatoração: aperfeiçoando o projeto de código existente Porto Alegre: Bookman, 2004, ISBN 8536303956. - IRVINE, K. R. C++ and object-oriented programming Upper Saddle River: Prentice-Hall, 1997. 526 p. Bibliografia e indice ISBN 0023598522 (broch.) - BECK, K. Implementation patterns Upper Saddle River, NJ: Addison-Wesley, 2008. ISBN 0321413091. - MARTIN, R. C. Clean code: a handbook of agile software craftsmanship. Prentice Hall, 2009. ISBN 0132350882. - AGANS, D. J. Debugging the nine indispensable rules for finding even the most elusive software and hardware problems. AMACOM, 2002. ISBN 0814471684.

Bibliografia Sugerida

[1] FEATHERS, Michel, Working Effectively with Legacy Code, 1st. edition, Prentice Hall, 2004; [2] GOODLIFE, Pete, Code Craft: The Pratice of Writing Excellent Code, 1st. Edition, No Starch Press, 2006; [3] PILONE, Dan and MILES, Russ, Head First Software Development, 1st. Edition, O'Reilly, 2008; [4] FREEMAN, Elisabeth et al, Head First Design Patterns, 1st. Editon, O'Reilly Media, 2004;

Termo de Entrega	Termo de Aprovação
Apresentado à Coordenação no dia	Aprovado em Reunião de CD no dia
Prof(a) Fabio Nogueira de Lucena *Professor*	*Prof. Dr. Eliomar Araújo de Lima* *Diretor do Instituto de Informática*

Termo de Homologação
Data de Expedição: Goiânia, ____ de _____ de _____.