MC426 – Engenharia de Software
1S 2023
Julho/2023 versão 1.0
Turma MC426B – Sala CC52

Plano de Desenvolvimento da Disciplina

Turma MC426B – Sala CC52
Horário: terças 21:00 – 23:00
quintas 19:00 -- 21:00
Professora: Cecília Mary Fischer Rubira
cmrubira@ic.unicamp.br
IC-Unicamp

PED
Sara das Mercês Silva s219590@dac.unicamp.br

PAD

Pré-Req.: MC302
Ementa: Paradigmas da Engenharia de Software. Processos de Software. Modelos de Processo de Software. Extração e Especificação de Requisitos. Análise e Projeto de Sistemas de Software. Padrões de Arquitetura e Padrões de Projeto.

Programa:

1. Paradigmas da Engenharia de Software
1.1 Visão geral da área de Engenharia de Software;
1.2 Conceitos de Produto (sistemas de software) e de Processo de Desenvolvimento;
1.3 Modelos de Processo de Software.
1.4 Reutilização de Software e Engenharia de Software Baseada em Componentes;
1.5 Métodos Ágeis de Desenvolvimento de Software.

2. Extração e Especificação de Requisitos
2.1 Requisitos de Software: Requisitos Funcionais, Não-Funcionais, de Usuário e de Sistema;
2.2 Técnicas para extração de requisitos;
2.3 Especificação de requisitos;
2.4 Modelos de casos de uso

3. Análise de Sistemas de Software
3.1 Análise Orientada a Objetos: modelagem estática e dinâmica;

3.2 Modelos de Sistema: modelo de classes, modelo de estados, modelo comportamentais (modelo de sequência e de comunicação), modelo de atividades, modelo de componentes em UML;
3.3 Padrões de Análise.

4. Projeto de Sistemas de Software
4.1 Conceitos Básicos: Abstração, Refinamento, Encapsulamento, Módulo, Hierarquia, Componentização.
4.2 Projeto Arquitetural e Projeto Detalhado;
4.3 Arquitetura de Software, Visões Arquiteturais e Atributos de Qualidade;
4.4 Projeto Orientado a Objetos e Processos de Desenvolvimento Centrados na Arquitetura;
4.5 Padrões e Estilos Arquiteturais;
4.6 Projeto de Componentes e de suas Interfaces;
4.7 Padrões de Projeto e Implementação de Modelos Arquiteturais.

Programa Detalhado:
- Conceitos da análise OO e projeto OO com foco na engenharia de modelagem de sistemas de software.
- Modelagem estática: construção de modelos de classes.
- Modelagem dinâmica: construção de modelos dinâmicos: modelos de sequência, de colaboração e de atividades.
- Conceitos Básicos: identificar objetos e classificá-los em classes, especificar atributos e operações das interfaces públicas (TAD e encapsulamento), herança e polimorfismo, e tratar relacionamentos de generalização/especialização, agregação e associação entre as classes.
- Conceitos Avançados: herança múltipla, interfaces, pacotes, metaclasses, delegação.
- Padrões de projeto: model-view-controller (MVC), composite, observer, singleton, abstract factory, etc
- Conceitos básicos de arquitetura de software, definição de componentes, conectores e configurações arquiteturais.
- Projeto arquitetural e padrões arquiteturais (layer, centrado em dados, máquina virtual, etc)
- Visão geral de Atributos de qualidade (security, availability, safety, reliability,etc) associados ao modelo arquitetural.
- Métodos de desenvolvimento de software centrados na arquitetura.

Objetivo da disciplina:
Ao final da disciplina o aluno deve ser capaz de criar modelos usando os conceitos do modelo de objetos e deve ser capaz de, partindo da especificação de um problema, criar modelos (estáticos e dinâmicos) que representem uma solução para o problema. Também o aluno deve ser capaz de definir um modelo inicial da arquitetura de software do sistema alvo, de acordo com os requisitos de qualidade priorizados durante o projeto arquitetural.

Critério de Avaliação:
Tarefas com entregas ao longo do curso: Projeto em grupo e Atividades em duplas.

A nota final da disciplina será a média das notas dos projetos e das atividades em duplas:

MediaFinal = (NotasAtividades + NotasProjeto) / 2

Se MediaFinal >= 5.0 o estudante está aprovado;
caso contrário, ele estará de exame.

MediaExame = (MediaFinal + NotaExame) / 2, se MediaExame > = 5.0 o estudante está aprovado; caso contrário, reprovado.
Obs. A matéria do exame

Tema e Cronograma do Projeto: a serem definidos.
Grupos de trabalho para o Projeto: a serem definidos.
Horário de Atendimento: a ser definido.

Cronograma Semanal de Atividades: Diário de Bordo no Moodle.

Referências:

Sommerville 9E
https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/

Sommerville 8E
https://ifs.host.cs.st-andrews.ac.uk/Books/SE8/

Bibliografia:
I. Sommerville. Software Engineering, Addison-Wesley, 10th edition, 2015.
G. Booch, J. Rumbaugh and I. Jacobson. The Unified Modeling Language User Guide, second edition, Addison-Wesley, 2005.
R. S. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, 8th Edition, McGrow Hill, 2014.
L. Bass, P. Clements & R. Kazman, Software Architecture in Practice, Second Edition, Addison-Wesley, 2003, SEI Series in Software Engineering.
M. Shaw & D. Garlan. Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, 1996.
F. Buschmann et al., A System of Patterns: Pattern-Oriented Software Architecture, Wiley, 1996.
E. Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

Bibliografia Expandida

Engenharia de Software
I. Sommerville, Software Engineering, Addison-Wesley, 10th edition, 2015.

I. Sommerville, Software Engineering, Addison-Wesley, 9th edition, 2011.
I. Sommerville, Software Engineering, Addison Wesley, 8th edition, 2007.
R. S. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, 8th Edition, McGrow Hill, 2014.
Roger S. Pressman, Software Engineering A Practitioner's Approach 7th Edition, Roger S Pressman, R. S. Pressman & Associates, Inc., 2010.
Pankaj Jalote. A Concise Introduction to Software Engineering. London: Springer-Verlag London, 2008.
Ghezzi,C., Jazayeri,M. & Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, 1991.
E.J. Braude, Software Engineering: An OO Perspective, John Wiley & Sons, INC., 2001.
W.P. Paula Filho, Engenharia de Software: Fundamentos, Métodos e Padrões, Segunda Edição, Editora LTC, 2001.

Modelagem e Projeto Orientado a Objetos

Rumbaugh, J. et al., Object-Oriented Modeling and Design, Prentice Hall, 1991.
Booch, G., Object-Oriented Design with Applications, Benjamin-Cummings, 1991.
G. Booch, J. Rumbaugh & I. Jacobson, The Unified Modeling Language User Guide, Second Edition, Addison Wesley, 2005.
H. Eriksson & M. Penker, UML Toolkit, Wiley, 1998.
P. Muller, Instant UML, Wrox, 1997.
C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, Second Edition, Prentice-Hall, 2002.
P. Harmon & M. Watson, Understanding UML, Morgan Kaufmann, 1998.
C. Richter, Designing Flexible OO Systems with UML, MTP, 2001.
M. Page-Jones, Fundamentals of OO Design in UML, Addison-Wesley, 2000.
P. Stevens, Using UML: software engineering with objects and components, Addison-Wesley, 1999.
Y. Lau, The Art of Objects: OO design and architecture, Addison-Wesley, 2001.
A. Carvalho & T. Chiossi, Introdução `a Engenharia de Software, Editora da UNICAMP, 2001.
C.M.F. Rubira, Apostila Introdução à Análise Orientada a Objetos e ao Projeto Arquitetural, IC-UNICAMP, 2022

Metodologias OO

I. Jacobson, G. Booch & J. Rumbaugh, The Unified Software Development Process, Addison Wesley,1999.
Krutchen, P., The Rational Unified Process: An Introduction, Second Edition, Addison-Wesley, 2000.
J. Cheesman & J. Daniels, UML Components: A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2001.
D. D'Souza & A. Wills, Objects, Components and frameworks with UML: The Catalysis Approach, Addison-Wesley, 1999.
S.R. Palmer & J.M. Felsing, A Practical Guide to Feature-driven development, The Coad Series.

Linguagens de Programação OO

K. Arnold & J. Gosling, The Java Programming Language, second edition, Addison-Wesley, 1997.

Flanagan, D. Java in a Nutshell, O'Reilly & Associates, 1996.
K. Beck, Extreme Programming: embrace change, Addison-Wesley, 2000.

Casos de Uso

G. Schneider & J.P. Winters Applying Use Cases: A Practical Guide, Second Edition, Addison-Wesley, 2001.
A. Cockburn Writing Effective Use Cases, Addison-Wesley, 2001.
D. Kulak & E. Guiney Use Cases: Requirements in Context, Addison-Wesley, 2000.
I. Jacobson, OO Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

Design Patterns

M. Fowler, Analysis Patterns: Reusable Object Models, Addison-Wesley, 1997.
M. Grand, Patterns in Java, Wiley, 1998.
E. Gamma et al., Design Patterns: Elements of reusable OO Software, Addison-Wesley, 1995.
Pattern Languages of Program Design, 1, 2 , 3 e 4, Software Patterns Series.

Frameworks Orientado a Objetos

W. Pree, Design Patterns for OO Software Development, Addison-Wesly, 1995.
M. Fayad & R. E. Johnson, Domain-Specific Application Framewoks, Wiley, 2000.
M. Fayad, D. C. Schmidt & R.E. Johnson, Implementing Applcation Frameworks, Wiley, 1999.
M. Fayad, D. C. Schmidt & R.E. Johnson, Building Applcation Frameworks, Wiley, 1999.

Arquitetura de Software

F. Buschmann et al., A System of Patterns: Pattern-Oriented Software Architecture, Wiley, 1996.
L. Bass, P. Clements & R. Kazman, Software Architecture in Practice, Second Edition, Addison-Wesley, 2003, SEI Series in Software Engineering.
C. Hofmeister et al., Applied Software Architecture, Addison-Wesley, 2000.
M. Shaw & D. Garlan. Software Architecture: Perspectives on an emerging discipline, Prentice Hall, 1996.
J. Bosch, Design and use of software architecture, Addison-Wesley, 2000.
L. Barroca et al. Software Architectures: advances and applications, Springer, 2000.
L. Hohmann, Beyond Software Architecture: creating and sustaining winning solutions, Addison-Wesley, 2003.
P.C.Clements & L.Northrop. Software Architecture: an executive overview. technical report CMU/SEI-96-TR-003, Software Engineering Institute, Carnigie-Mellon University, February 1996.
F. Bachman et al., The architecture-based design method, technical report CMU/SEI-2000-TR-001, Software Engineering Institute, Carnegie-Mellon University, January 2000.
P. Krutchen. The 4+1 view model of software architecture. IEEE Software, pages 42–50, November 1995.
M. Barbacci et al., Steps in architecture tradeoff analysis method: quality attribute models and analysis. Technical report CMU/SEI-97-TR-029, Software Engineering Institute, Carnegie-Mellon University, May 1998.

Desenvolvimento Baseado em Componentes

C. Szyperski, Component software: beyond OO programming, Addison-Wesly, 1998.
G.T.Leavens & M.Sitaraman, Foundations of Component-based Systems, Cambridge University press, Cambridge, UK, 2000.
F. Bachman et al., Volume II: Technical concepts of component-based software engineering, technical report CMU/SEI-2000-TR-008, Software Engineering Institute, Carnegie-Mellon University, April 2000.