01. Engenharia de Software

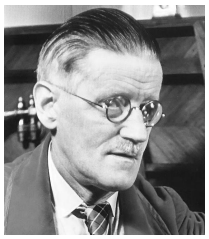
Princípios de Engenharia de Software (Texto em Elaboração)

Italo S. Vega
italo@pucsp.br

Faculdade de Estudos Interdisciplinares (FACEI)

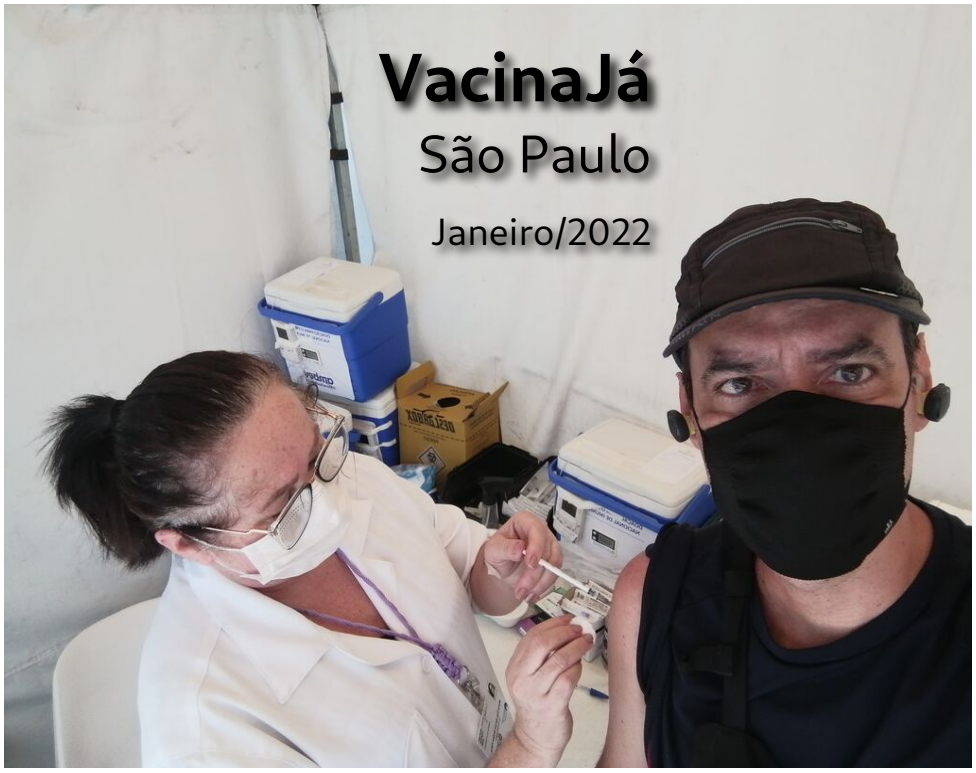
Pontifícia Universidade Católica de São Paulo

Sumário

 James Joyce — Mistakes are the portals of discovery.

Apresentação da Disciplina

Professor



italo@pucsp.br

PUCSP-Teams

Plano de Ensino

- O que estudaremos nesta disciplina?
- Como ela será oferecida?

Ementa

- Estudo das Técnicas de Análise e Projeto de sistemas de Software.
- Estudo de Processos de desenvolvimento.
- Estudo dos Aspectos Gerais de Qualidade de Software e Gerência de Projetos.
- Aplicação de Padrões de Projeto.
- Estudo das Arquiteturas de software orientada a dados.
- Avaliação das Tecnologias de implementação de sistemas de software.

Objetivos Gerais

 No contexto da programação artesanal, enfatizar a importância da análise e de o *design* visando melhorar a qualidade do programa.

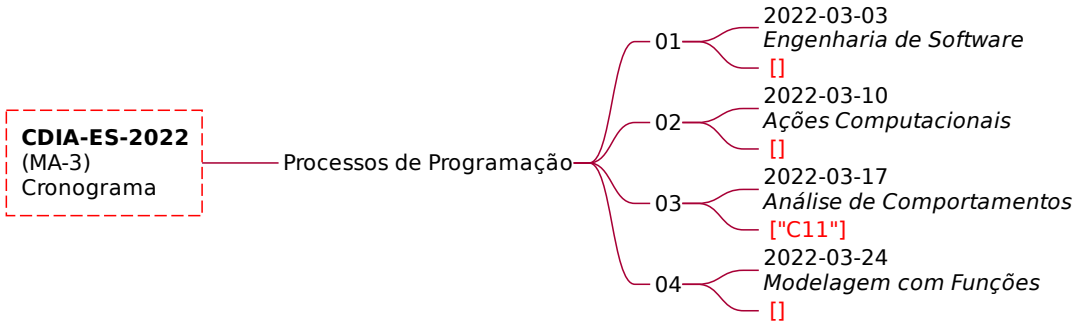
Objetivos Específicos

 Capacitar o aluno para:

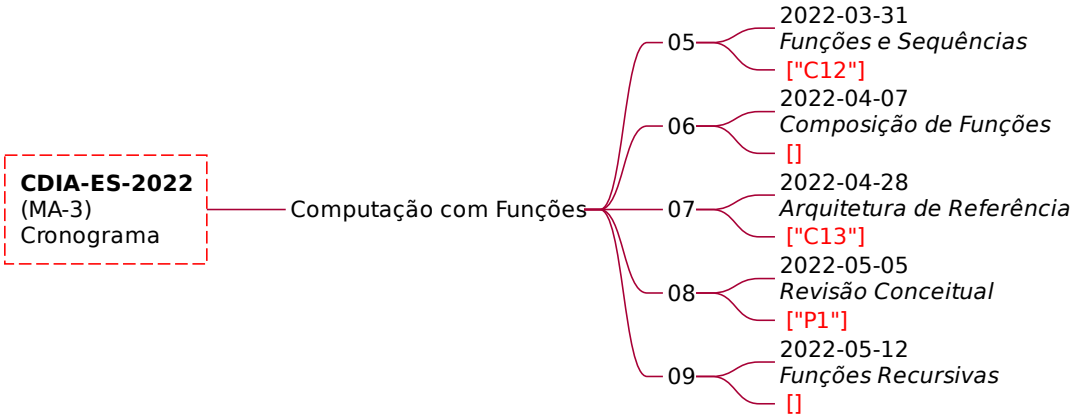
- representar de estruturas de dados
- formalizar modelos em uma notação matemática
- formalizar modelos na linguagem Python
- comparar modelos formalizados como em uma notação matemática e em Python
- elaborar modelos contínuos e discretos e relacioná-los com programas

Temas de Estudo

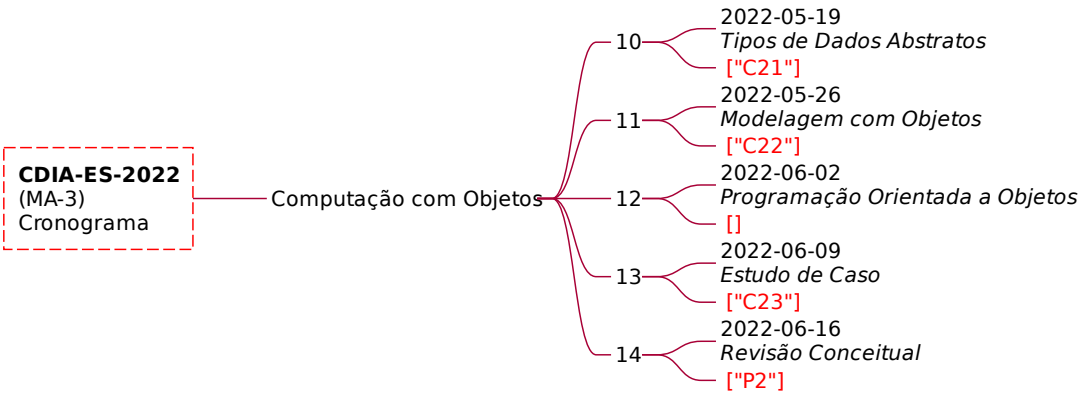
Cronograma 1



Cronograma 2



Cronograma 3



Bibliografia

Bibliografia Básica

https://www.pucsp.br/biblioteca

- PRESSMAN, R. S. Engenharia de Software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.
- SCHACH, S. R. Engenharia de software: os paradigmas clássico e orientado a objetos. Porto Alegre: Bookman, 2011.
- SOMMERVILLE, I. Engenharia de software. 8. ed. Boston: Addison Wesley, 2007.

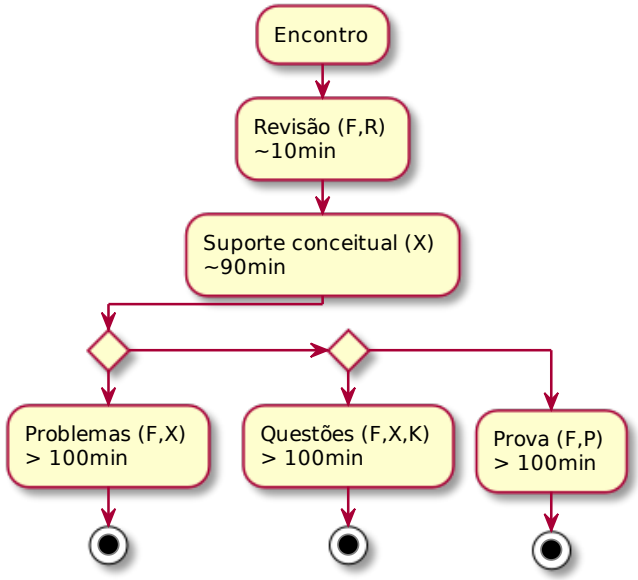
Bibliografia Complementar

- ROSSUM, G. V. The Python Language Reference Manual. (F. L. Jr. Drake, Org.). Network Theory, 2003.
- BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. The unified modeling language user guide. São Paulo: Pearson Education, 2005.

- BRAUDE, E. Projeto de software. Porto Alegre: Bookman, 2005.
- LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007.
- MARTIN, R. C. UML for Java programmers. New Jersey: Prentice Hall, 2002.
- VLISSIDES, J. Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000

Encontros e Avaliação

Dinâmica dos Encontros



 "**Aprender** fazendo projetos de **pequeno** porte!"

Critérios de Avaliação

$MF = \frac{N_1+N_2}{2}$

$N_i = 60\%C_i + 33\%P_i + 7\%F_i$

- P: Provas
- C: Atividades
- F: Frequência nos encontros

F será contabilizada presencialmente.

Plágio

> [...] acontece quando alguém **reescreve** frases, citações ou conceitos de outros autores, seja de livros ou da internet, sem dar os devidos créditos.

Plágio conceitual Caso você mencione uma conceituação **criada por outra pessoa** sem mencionar a fonte, será considerado um "plágio conceitual".

- https://guiadoestudante.abril.com.br/estudo/o-que-plagio-academico-como-evitar/

 **Situações de plágio serão punidas com nota ZERO**

> "O plágio é considerado **crime**, já que é uma violação de direitos autorais."

Regras Sanitárias

Obedecendo ordens da **Direção**:

> Existir algum estudante **sem uso** de máscara protetora autorizada pela ANVISA durante o período do encontro presencial, a aula será **interrompida** e o caso **relatado** à Coordenação de Curso.

Introdução

> There has already been a debate on the **increasing information overload** that has begun in the so-called knowledge society (Marotzki & Jörissen, 2010). There is a distinct difference between **knowledge of how things can be better produced**, more efficiently (Verfügungswissen) and **knowledge of why or for what reason things are done or produced** (Orientierungswissen). Both forms are inversely proportional: Whereas Verfügungswissen is easily accessible due to open formats and open archiving, it does not, however, contribute to an increase of Orientierungswissen.
> — M. Deimann e R. Farrow, http://www.irrodl.org/index.php/irrodl/article/download/1370/2542 (grifo pessoal)

Aprendizagem Baseada em Investigação

O material de estudo foi concebido nos moldes das metodologias ativas de aprendizagem, com ênfase em uma estratégia investigativa (*inquiry-based learning*, Stefanidou, Stavrou, Kyriakou, & Skordoulis (2020), Wright (2013), Siemens Stiftung ([s.d.])) de programação de computadores, com influência de Krathwohl (2002). As seguintes linhas se destacam na elaboração do material.

Lógica e Programação

O suporte de formalização lógica contou com os trabalhos de Lamport (2002), Wordsworth (1992), Barker-Plummer, JonBarwise, & Etchemendy (2011), Wadler (2015) e Ramos, Neto, & Vega (2009).

Python Interpretado

A plataforma Java que serviu de base para a realização dos experimentos teve a versão:

```
python --version
Python 3.8.10
```

 O processo típico de programação usando a tecnologia Python inclui duas etapas fundamentais: (i) edição e (ii) execução. Um serviço *online* que suporta tal estilo de programação é oferecido no seguinte endereço:

https://replit.com/languages/python3

Dessa maneira, o processo de programação segue um estilo interativo e incremental. Elementos do programa são desenvolvidos e executados um de cada vez.

 Achei um tutorial a respeito da linguagem Python:

https://www.tutorialspoint.com/python/index.htm

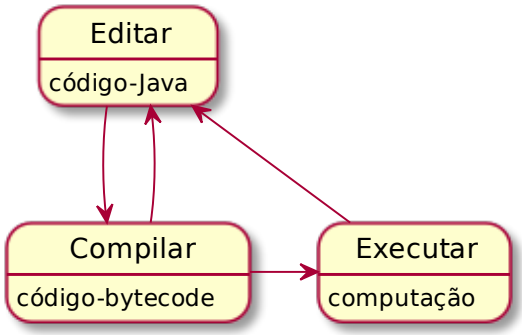
Cadernos Interativos JupyterLab

A tecnologia dos cadernos Jupyter (próxima seção) oferecem um espaço interativo para a exploração de códigos em Python. O conteúdo dos cadernos combinam seções de texto e de código. Cria-se a possibilidade de documentar uma parte da especificação de um comportamento com o respectivo código executável. Tal combinação favorece o desenvolvimento de uma lógica simbólica no contexto de um determinado problema com a exploração dinâmica do seu uso. Trata-se de uma técnica efetiva no desenvolvimento de software[1]. Utilizou-se a seguinte versão da tecnologia Jupyter como ambiente de estudo:

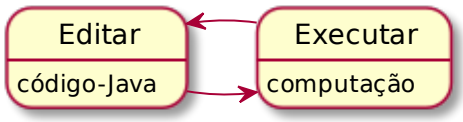
```
jupyter-lab --version
3.0.16
```

[1] "Programmers wanted to know where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails (http://dannorth.net/introducing-bdd/).

 — Aprendi que um processo de desenvolvimento de código Java inclui três passos básicos. Trata-se de um processo criativo que envolve ciclos de **refinamento**. No caso de linguagens compiladas:



No caso de linguagens interpretadas:



 — A plataforma **JupyterLab** oferece recursos para a elaboração de documentos contendo fragmentos executáveis de código, semelhante ao caso das linguagens interpretadas:

> https://jupyterlab.readthedocs.io/en/stable/index.html

Existe um serviço *online* do ambiente oferecido pela equipe do projeto (Clique em **JupyterLab**):

> https://jupyter.org/try

Instalação

 — Recomendo uma instalação local do ambiente para facilitar o estudo. Um procedimento de instalação da plataforma encontra-se em:

> https://www.tutorialspoint.com/jupyter/jupyterlab_installation_and_getting_started.htm

Personagens e Histórias OC2-RD2

Histórias OC2-RD2 A técnica OC2-RD2 (Vega, 2018) ajuda na produção de histórias para ambientes de aprendizagem híbridos e interativos, com um ritmo caracterizado pela alternância entre observações objetivas (essencialmente, execução de código) com subjetivas (análise da execução baseada em lógica).

Várias histórias **interativas** envolvendo três **personagens** centrais serão narradas ao longo dos encontros:

 — Meu nome é Fubã e represento a faceta de **curiosidade** das pessoas e possuo um perfil **investigativo**.

 — Sou Espec, um colega do Fubã e colaboro com uma faceta equilibrada de **fundamentos teóricos** e **capacidade de realização**, principalmente na área de desenvolvimento de software.

 — Como "profe" do Fubã e Espec, procuro ajudá-los na sua **formação**, cuidando para que os conceitos apoiem-se em uma **bibliografia acadêmica** e sejam aplicados com um certo grau de **rigor na solução** de problemas computacionais.

Entrarão em cena muitos convidados, contribuindo para o conteúdo das histórias de aprendizagem. Vamos a nossa primeira história!

Exercícios

Os exercícios são classificados com asteriscos. Aqueles sem asteriscos destacam pontos de fixação de conceitos e definições a serem lembrados. Exercícios com um asterisco envolvem contextos voltados para acionar o nível de entendimento do aprendiz. Finalmente, exercícios marcados com dois asteriscos oferecem desafios de aplicação e, por vezes, exigem que o aprendiz recorra a níveis cognitivos mais elevados.

1 Algoritmos e Inteligência

1.1 Quebra-cabeça Batalha Naval

Contexto Conversando com um colega, Fubã ficou empolgado com a história dos quebra-cabeça Batalha Naval[1].

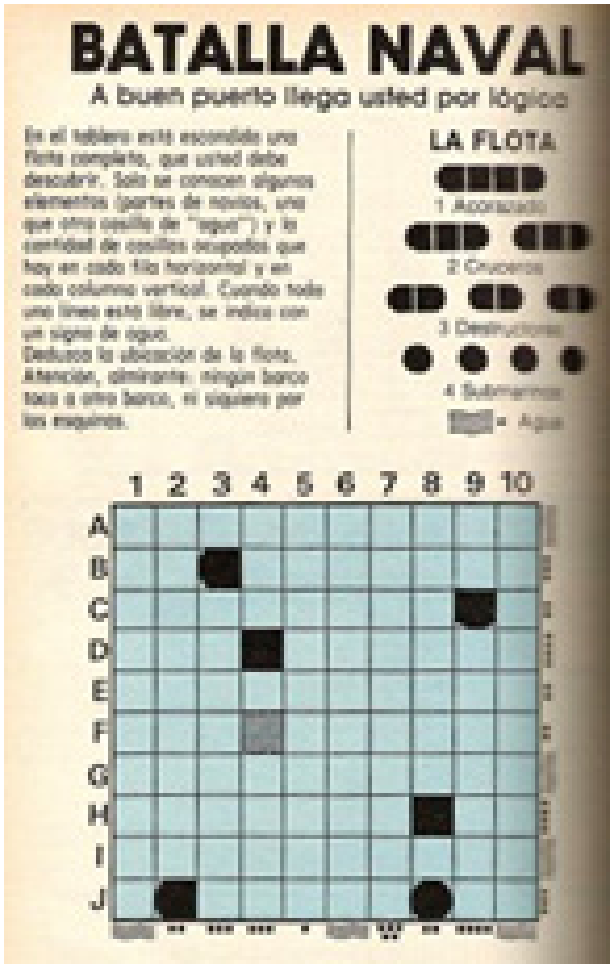


Figura 1.1: https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

Em um campo de batalha com dimensão 5×5 encontram-se cinco navios escondidos a serem destruídos por disparos de torpedos. Cada navio ocupa uma única posição do campo de batalha. O jogador recebe uma pista a respeito de quantos navios encontram-se em cada linha e coluna do campo.

[1]https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

 — Vou criar uma versão simplificada para aprimorar o meu conhecimento a respeito de programação orientada a objetos:

`fonte/bn/monolitico/dist/BN_Monolitico`

Enunciado Desenvolver uma versão executável do quebra-cabeça Batalha Naval.

Resumo

1. Desenvolvimento de código por compilação: edita-compila-executa.
2. Desenvolvimento de código por interpretação: edita-executa.

Referências

Barker-Plummer, D., JonBarwise, & Etchemendy, J. (2011). *Language, proof, and logic* (2º ed, p. 620). CSLI Publications.

Krathwohl, D. R. (2002). A revision of Bloom's Taxonomy: an overview. *Theory in practice*, *41*(4), 212–218.

Lamport, L. (2002). *Specifying Systems: The TLA+ language and tools for hardware and software engineers*. Addison-Wesley. Recuperado de https://lamport.azurewebsites.net/tla/book-21-07-04.pdf

Ramos, M. V. M., Neto, J. J., & Vega, I. S. (2009). *Linguagens Formais: Teoria, Modelagem e Implementação*. Bookman.

Siemens Stiftung. ([s.d.]). Turning kids into classroom explorers. Recuperado de https://www.siemens-stiftung.org/en/foundation/education/stem-and-inquiry-based-learning/

Stefanidou, C., Stavrou, I., Kyriakou, K., & Skordoulis, C. (2020). Inquiry-Based Teaching and Learning in the Context of Pre-Service Teachers' Science Education. *Universal Journal of Educational Research*. doi:10.13189/UJER.2020.082223

Vega, I. S. (2018). *Elaboração de Histórias OC2-RD2*. PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO—PUC-SP.

Wadler, P. (2015). Propositions as Types. *Communications of the ACM*, *58*(12), p. 75–84. doi:10.1145/2699407

Wordsworth, J. B. (1992). *Software development with Z: a practical approach to formal methods in software engineering* (p. 334). Wokingham, England Reading, Mass.: Addison-Wesley Pub. Co.

Wright, S. (2013). The power of student-driven learning. (TEDx Talks, Org.). Recuperado de https://www.youtube.com/watch?v=3fMC-z7K0r4