PROJETO FINAL PARA A DISCIPLINA DE PROGRAMAÇÃO AVANÇADA DO PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA DA UTFPR: *MODELO PARA COMPOSIÇÃO DO RELATÓRIO*

Primeiro Autor, Segundo Autor
primeiro.autor@utfpr.edu.br, segundo.autor@utfpr.edu.br

Disciplina: **Programação Avançada** – Prof. Dr. Robson R. Linhares
Departamento Acadêmico de Informática – DAINF - Campus Curitiba
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo – Este documento apresenta um modelo para o texto do relatório de Programação Avançada, além de instruções para o projeto em si. O resumo deve trazer uma visão geral do projeto, contemplando sucintamente a sua motivação e o seu contexto, o seu objeto de estudo (uma aplicação de sistemas de informação), o seu processo de desenvolvimento e os resultados obtidos. Um exemplo de resumo seria: A disciplina de Programação Avançada exige o desenvolvimento de um *software*, no formato de um sistema de informação, para fins de aprendizado de técnicas de programação orientada a objetos em C++, incluindo uma introdução para técnicas de engenharia de software. Para tal, neste trabalho, escolheu-se o jogo Labirinto++, onde o jogador enfrenta inimigos em um dado labirinto. O jogo tem três fases que se diferenciam por dificuldades para o jogador. Para o desenvolvimento do jogo foram levantados textualmente os requisitos e elaborada uma modelagem (análise e projeto) usando Diagrama de Classes em *Unified Modeling Language* (*UML*). Subsequentemente, em linguagem de programação Java, realizou-se o desenvolvimento que contemplou os conceitos usuais de Orientação a Objetos, bem como alguns conceitos avançados como Polimorfismo, Classe Abstrata, Gabaritos, Persistências de Objetos por Arquivos Binários, Sobrecarga de Operadores e *Standard Template Library* (*STL*). Depois da implementação, os testes e uso do jogo demonstraram sua funcionalidade conforme os requisitos e o projeto. Por fim, salienta-se que o projeto permitiu cumprir o objetivo de aprendizado visado.

Palavras-chave ou Expressões-chave (máximo quatro, não excedendo três linhas)**:** Artigo-Relatório Modelo para o Projeto em Programação Avançada, Trabalho Acadêmico Voltado a Implementação em C++, Normas Internas para Elaboração de Projeto, Exemplo de Conteúdos de Projeto de Programação Avançada.

Abstract - This document shows a model for the manuscript to the academic work of *Programação Avançada* as well as it presents general instructions about this academic work. Regarding the contents of the abstract, they should give a general explanation about the work. Precisely, the abstract must shortly present the work motivation and context, its study object (ordinarily an information system), its development process, and the obtained results. An instance of abstract would be:

• • •

Key-words or Key-expressions (maximum four, not exceeding three lines)**:** Paper Template to the Final Project of Programação Avançada, Academic Work Related to C++ Implementation, Internal Rules for Project Development, Examples of Elements for the Work of a Programming Course.

INTRODUÇÃO

Este documento apresenta um modelo para o relatório do projeto de Programação Avançada. Este modelo é baseado no modelo proposto pelo Prof. Jean Marcelo Simão para a disciplina de Fundamentos de Programação 2 [4], o qual por sua vez é baseado em um dado modelo de artigos de Anais do Seminário de Iniciação Científica e Tecnológica da UTFPR. Seu objetivo é mostrar a configuração básica do projeto e do texto, bem como os detalhes sobre a publicação de figuras, tabelas, equações e referências. O idioma oficial é o português, porém serão aceitos também textos em inglês se previamente acordado com o Professor.

Conforme dito em classe e especificado no plano de aulas, parte da nota da disciplina é oriunda de um Projeto Final, conforme a regra para cálculo especificada também no plano de aulas. A parte prática deste projeto consiste primeiramente em levantar requisitos textualmente e modelar (analisar e projetar) o *software* em questão. Este *software* pode ser um sistema de informação como um controle de estoque, um sistema de vendas, ou ainda uma simulação de conceitos da física/química, ou ainda um jogo. A modelagem deve utilizar diagramas em Linguagem Unificada de Modelagem - *Unified Modeling Language* (*UML*), particularmente Diagrama de Classes, porém eventualmente outras formas de representação tais como Diagrama de Atividades e Diagrama de Estados, se os autores acharem conveniente.

Em seguida à modelagem do projeto, este deve ser desenvolvido/implementado este *software* em linguagem C++, respeitando os princípios da orientação a objetos (explicados em classe), salientando aqui coesão e desacoplamento.

Este trabalho é proposto visando ampliar a aplicação dos conceitos aprendidos em classe ou mesmo a aplicação de novos conceitos aprendidos extraclasse. Assim sendo, o *software*/jogo escolhido para ser implementado deve ter complexidade tal que permita utilizar diversos recursos da técnica de orientação a objetos, sobretudo os ensinados em classe. Portanto, deve-se encaminhar inicialmente ao professor da disciplina uma proposta de projeto, que permitirá que este avalie se o sistema escolhido se faz apropriado.

Uma vez escolhido e implementado o *software*, este será expresso em um documento escrito. O documento será entregue no dia da apresentação do projeto, conforme combinado em classe e especificado no documento de planejamento da disciplina. Tanto a apresentação do desenvolvimento (levantamento de requisitos, modelagem e implementação) do *software*, quanto o documento escrito serão avaliados, permitindo compor uma nota para o projeto. O desenvolvimento será avaliado inclusive por meio do seu acompanhamento, que se dá pelas interações para com o professor que devem ser solicitadas pelos discentes (i.e. alunos).

Quanto ao relatório escrito, este deve conter um conjunto de elementos segundo um modelo dado, o qual é detalhado nas seções subsequente deste presente documento. Justamente esse modelo tem por finalidade padronizar o trabalho escrito a ser apresentado na disciplina de Programação Avançada do PPGCA. Os projetos apresentados que não sigam o padrão aqui apresentado poderão, a critério do professor, ser penalizados e (no limite) até rejeitados. Idem para projetos não escritos corretamente.

Os documentos **não** poderão ser entregues ao professor de maneira impressa (nem mesmo se usar frente e verso). Assim sendo, apenas a versão digital será aceita. Na verdade, é necessário enviar o documento escrito em formato digital *.doc/.docx* e *.pdf* via link no Moodle, a ser disponibilizado pelo professor na época apropriada. Também é necessário enviar as implementações respectivas, diagramas de projeto e demais materiais de suporte.

Quanto à introdução em si do relatório, mais precisamente, ela deve apresentar quatro parágrafos contendo:

(1) em que contexto (*i.e.* disciplina de Programação Avançada) este projeto se dá e qual é o objetivo de tal realização;

(2) qual é o objeto de estudo e da implementação do projeto (*i.e.* jogo ou outro sistema escolhido e acordado previamente com o professor);

(3) o método utilizado que em suma é o ciclo clássico de Engenharia de *Software* de forma simplificada (i.e. definição dos requisitos, modelagem-projeto via diagramas em *UML*, implementação em C++ orientado a objetos e testes pelo uso do *software*); e

(4) introdução às seções subseqüentes.

Uma vez explicado o necessário à introdução em si, este presente documento-modelo de artigo-relatório apresenta demais seções necessárias (mas não limitantes) que o projeto deve conter, bem como seus conteúdos. Estas indicações de conteúdos mostram o que se faz necessário contemplar em cada seção, além de explicar alguns itens de formatação de elementos contemplados.

EXPLICAÇÃO DO *SOFTWARE*

Nesta seção se deve discorrer a explicação do *software* do ponto de vista das funcionalidades e características percebidas pelos seus usuários. Portanto, salienta-se que esta seção **não** propõe (em absoluto) explicar o projeto ou a implementação do *software*.

Isto dito, esta seção assim como as demais, devem seguir as regras de formatação dadas. Justamente, quanto à formatação, o texto do trabalho deve seguir as seguintes regras (as quais foram em geral seguidas para compor este presente modelo):

- O trabalho deve ser totalmente digitado em fonte Times New Roman. Esta diretriz inclui, portanto, o título do trabalho, autores, filiação e endereços, títulos de seções e legendas de figuras e tabelas, além do texto normal do trabalho. O texto deve ser digitado com alinhamento 'justificado' ou ajustado.
- O trabalho completo, incluindo figuras e tabelas, deve ser limitado a doze (12) páginas (no máximo) em papel de tamanho padrão A4 (21 cm x 29,7 cm). Não reduzir figuras e tabelas a tamanhos que sacrifiquem o entendimento dos símbolos e legendas nelas contidos.
- Cada página, no tamanho A4, deve ser formatada de modo a apresentar 2,5 cm de margem em todos os lados do documento. Dentro desta área o texto deve ser formatado em uma única coluna, sem incluir moldura no texto**.**
- O título deve ser digitado em negrito, em letras maiúsculas, centralizado e em tamanho 14 pt, não excedendo três linhas, seguido de uma linha em branco (12 pt) e pelas linhas que conterão o(s) nome(s) do(s) autor(es), em tamanho 12 pt. Em seguida, deverá vir a filiação e o(s) endereço(s) para correspondência do(s) autor(es) (tamanho 10 pt) separada por uma linha em branco. Deve-se deixar 3 linhas de espaço antes do resumo, e uma linha entre os itens subsequentes (palavras-chave, *abstract* e *Key-words*).
- Digitar o título **Resumo** em negrito, alinhado à esquerda, tamanho 10 pt, seguido de um traço. Sem trocar de linha, digitar o resumo, em tamanho 10 pt com alinhamento justificado. Pular uma linha e digite o título **Palavras-chave:** em negrito, alinhado à esquerda, tamanho 10 pt. Digitar então no máximo quatro (4) palavras-chave, separadas por vírgulas, com somente a primeira letra de cada palavra chave em maiúscula. Na sequência devem vir o ***Abstract*** e ***Keywords*****:** em inglês, seguindo o mesmo padrão de formatação do resumo e das palavras-chave. Tanto o resumo quanto o *abstract* devem conter no máximo 200 palavras.
- A seguir, separado por 2 linhas (12 pt), o texto deve ser iniciado pela Introdução. Os títulos das seções (Introdução, etc.) devem ser escritos em negrito, sem numeração, em maiúsculo e alinhados à esquerda, sendo que o conteúdo, propriamente dito, deve ser iniciado após espaçamento de uma linha e tabulação (1 cm).
- Ao final de cada seção deve-se deixar uma linha em branco. Todo o texto deverá ser escrito em espaço simples. Para as subseções, somente a primeira letra do subtítulo deve ser maiúscula, sendo todas em negrito, sem numeração, com o título alinhado à esquerda. Uma vez escrito o subtítulo, pular uma linha. Após esta linha (em branco), iniciar o texto da subseção.

- As ilustrações e gráficos podem ser em preto-e-branco ou em escala cinza ou mesmo coloridos, mas sempre centralizados. As notas de rodapé[1] devem ser colocadas na parte inferior da página correspondente separadas por um traço conforme modelo. Usar o tamanho de 8 pt.
- As referências bibliográficas devem ser listadas no fim do artigo, na ordem de citação, conforme formato da Associação Brasileira de Normas Técnicas (ABNT). No texto, as citações devem ser referenciadas por seu número colocado entre colchetes, por exemplo, [1] e [2].

Para apoiar a explicação do *software*, aconselha-se utilizar de recursos como gráficos, telas e figuras do próprio *software*. As figuras, tabelas, etc., devidamente referenciadas no texto, podem ser colocadas da maneira mais conveniente para o autor em uma ou duas colunas, desde que o texto permaneça em apenas uma coluna. Antes e após os elementos não textuais e suas respectivas legendas, deve-se deixar uma linha de espaçamento.

Os autores não devem se esquecer da colocação de legendas nas figuras, tabelas e outros elementos gráficos. As figuras devem ser numeradas seqüencialmente com algarismos arábicos conforme o exemplo da figura 1.

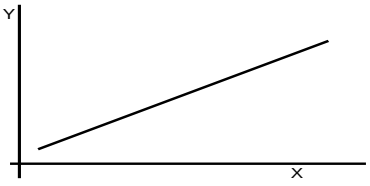

Figura 1. Centralizada na coluna e com legenda abaixo da figura.

Caso os autores façam uso de equações em alguma parte do texto, todas estas equações deverão ser tabuladas a 1 cm da margem esquerda e numeradas seqüencialmente, com os números entre parênteses, conforme o exemplo abaixo:

$$e(t) = \sum_{n=1}^{5} \frac{1}{2+n} \cos\left(2\pi m t\right) \quad (1)$$

As equações devem ser referenciadas no texto da seguinte forma: "Substituindo a equação (1) na equação (3), obtém-se ..."

DESENVOLVIMENTO DO *SOFTWARE* NA VERSÃO ORIENTADA A OBJETOS

Nesta seção se deve discorrer a explicação do desenvolvimento do jogo/*software* utilizando orientação a objeto, culminando na programação em C++. A explicação deve ser feita de maneira tal a **não** ser um relatório técnico repleto de detalhes, mas que seja capaz de sintetizar e valorizar os recursos técnicos utilizados (i.e. sucinto e suficiente).

Nesta explicação, deve-se primeiramente listar os **requisitos funcionais** levantados para o *software* em questão. Os requisitos devem estar enquadrados em uma tabela de três colunas, onde a primeira coluna traz os requisitos, a segunda coluna traz os pesos atribuídos pelo professor a cada requisito e a terceira a situação (*status*) do requisito, que pode ser 'realizado',

[1] Exemplo de nota de rodapé – Apenas os diagramas em UML que devem ser impressos, preferencialmente em papel reciclado.

‘semi realizado’, ‘abandonado’ etc. **A Tabela 1 exemplifica o exposto, no contexto de um jogo**.

Tabela 1. Lista de Requisitos de um Jogo e suas Situações.

N.	Requisitos Funcionais	Peso atribuído	Situação
1	Apresentar menu de opções aos usuários do Jogo	10	Requisito previsto inicialmente e realizado
2	Permitir um ou dois jogadores aos usuários do Jogo	5	Requisito previsto inicialmente e realizado
3	Disponibilizar ao menos três fases que podem ser jogadas sequencialmente ou selecionadas.	10	Requisito previsto inicialmente e realizado parcialmente – faltou ainda a última fase.
4	Ter seis tipos distintos de inimigos	10	Requisito previsto inicialmente e realizado
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias.	5	Requisito previsto inicialmente e realizado
6	Ter inimigo “Chefão” na última fase	5	Requisito previsto e não realizado
7	Ter quatro tipos de obstáculos.	5	Requisito previsto inicialmente e realizado
8	Ter em cada fase entre um e quatro tipos de obstáculos com número aleatório de obstáculos.	10	Requisito previsto inicialmente e realizado
9	Ter representação gráfica de instâncias.	5	Requisito previsto inicialmente e realizado
10	Ter em cada fase um cenário de jogo com obstáculos.	5	Requisito previsto inicialmente e realizado
11	Gerenciar colisões entre jogador e obstáculos.	5	Requisito previsto inicialmente e realizado
12	Gerenciar colisões entre jogador e inimigos.	10	Requisito previsto inicialmente e realizado
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (*ranking*).	5	Requisito previsto inicialmente e realizado
14	Permitir Pausar o Jogo	5	Requisito previsto inicialmente e realizado
15	Permitir Salvar Jogada.	5	Requisito previsto e não realizado

A explicação do desenvolvimento segue devendo-se:

- Utilizar um Diagrama de Classes em *UML* para explicar as classes e suas relações (que normalmente atendem aos requisitos).
- Utilizar demais diagramas da *UML* **caso** o(s) aluno(s) os saiba(m) ou esteja(m) fazendo esforços para apreendê-los.
- À luz dos diagramas, explicar o desenvolvimento de maneira sucinta e suficiente.
- Valorizar as ‘sofisticações’ que tenham sido realizadas, como uma eventual função que permita ao jogador humano enfrentar um dado ‘jogador artificial’.

– Valorizar a interdisciplinaridade como a aplicação de conceitos de física e matemática aprendidos em disciplinas do ensino médio e (preferencialmente) em disciplinas da graduação.

Para a implementação em C++ pode-se, opcionalmente, utilizar uma biblioteca gráfica (*e.g. Allegro*, *TCL*/*TK*, *OpenGL* ou outra), pois isto valoriza o trabalho esteticamente além de demonstrar a capacidade de 'pesquisa' e aprendizado. No entanto, **os projeto que utilizarem biblioteca gráfica não serão necessariamente melhor avaliados do que os trabalhos que não utilizarem**, pois o principal objetivo da avaliação é verificar a correta aplicação dos conceitos de Orientação a Objetos.

Esta seção em questão é muito importante no projeto e será corrigida com muita atenção pelo professor. Pede-se, por fim, que todos os autores revisem cuidadosamente a versão final do trabalho para evitar erros de português, digitação e/ou formatação.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Nesta seção, em relação aos conceitos aprendidos, deve-se apresentar uma tabela de conceitos utilizados e não utilizados tal qual a Tabela 2. Deve-se também apresentar outra tabela justificando o uso ou não uso, tal qual a Tabela 3.

Oportunamente, todas as tabelas que venham a ser utilizadas deverão ser numeradas seqüencialmente com algarismos arábicos, conforme o exemplo abaixo:

Tabela 2. Lista de Conceitos Utilizados e Não Utilizados no Projeto.

N.	Conceitos	Uso	Onde
1	**Elementares:**		
	- Classes, objetos,	Sim	Todos .h e .cpp
	- Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp
	- Métodos (com e sem retorno).	Sim	Todos .h e .cpp
	- Métodos (com retorno *const* e parâmetro *const*).	Sim	Todos .h e .cpp
	- Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp
	- Classe Principal.	Sim	Main.cpp & Principal.h/.cpp
	- Divisão em .h e .cpp.	Sim	No projeto.
2	**Relações de:**		
	- Associação		
	- Agregação via associação		
	- Agregação propriamente dita.		
	- Herança elementar.		
	- Herança em diversos níveis.		
	- Herança múltipla.		
3	**Ponteiros, generalizações e exceções**		
	- Operador *this*		
	- Alocação de memória (*new* & *delete*)		
	- Gabaritos/*Templates* criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates)		
	- Uso de Tratamento de Exceções		
4	**Sobrecarga de:**		
	- Construtoras e Métodos.		
	- Operadores		
	Persistência de Objetos		
	- Texto via Arquivos de Fluxo		
	- Binário		

5	**Virtualidade:**		
	- Métodos Virtuais.		
	- Polimorfismo		
	- Métodos Virtuais Puros / Classes Abstratas		
	- Coesão e Desacoplamento		
6	**Engenharia de Software**		
	- Levantamento de Requisitos Textualmente e Tabelado		
	- Diagrama de Classes em *UML*		
	- Outros diagramas em *UML*: Diag. de Casos de Uso, Diag. de Atividades, Diag. de Estados, Diag. de Seqüência, Diag. de Pacotes etc.		
7	**Biblioteca Gráfica**		
	- Funcionalidades Elementares.		
	- Funcionalidades Avançadas como: • tratamento de colisões • duplo *buffer* • *especificar aqui outras*		
	Obs.: especificar quais funcionalidades.		
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc		
	- Ensino Médio (especificar quais Conceitos aqui)		
	- Ensino Superior (especificar quais Conceitos aqui)		
8	**Organizadores:**		
	Espaço de Nomes (*Namespace*) ou pacotes criados pelos autores.		
	Classes aninhadas.		
	Estáticos e String:		
	Atributos estáticos e chamadas estáticas de métodos.		
	A classe Pré-definida *String* ou equivalente.		
9	**Standard Template Library (STL)**		
	Vector da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).		
	List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).		
	Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas ***ou*** *Multi-Mapas*.*		
	*Obs.: Listar apenas os utilizados		
10	**Uso de Conceito Avançado no tocante a Orientação a Objetos.**		
	Ou Padrões de Projeto: GOF **Ou Programação orientada a eventos e visual:** *Objetos gráficos como formulários, botões etc (*Listar apenas os utilizados) **Ou Programação concorrente:** *Threads (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando* Posix, C-Run-Time, Win32API ou afins (com ou sem uso de *Mutex, Semáforos,* ***ou*** *Troca de mensagens*). ***Ou API de Comunicação em Rede:*** *Cliente Servidor.*		

Ressalta-se que as legendas das figuras devem ser colocadas abaixo, enquanto que as legendas das tabelas devem ser colocadas acima das mesmas.

Tabela 3. Lista de Justificativas para Conceitos Utilizados e Não Utilizados no Trabalho.

No.	Conceitos	Situação
1	**Elementares**	Classe e Objetos foram utilizados porque ... Atributos, Variáveis e Constantes foram utilizados porque ... . . .
2	**Relações**	Associação foi utilizada porque....
...	. . .	...
10	**Uso de conceitos avançados.**	Padrões de Projeto não foram utilizados porque....

DISCUSSÃO E CONCLUSÕES

Esta seção deverá apresentar uma reflexão sobre o desenvolvimento e os resultados obtidos. Certamente uma conclusão bem elaborada ajuda na avaliação do professor. Por sua vez, a avaliação da qualidade do projeto como um todo (pelo professor) será baseada em:

- Quantidade e qualidade de conceitos utilizados na elaboração do *software*, envolvendo particularmente os bons princípios de Orientação a Objetos, como organização, encapsulamento e reutilização, todos baseados no princípio de coesão e desacoplamento doutrinados em classe.
- Complexidade do problema e respectiva quantidade e qualidade de projeto e código o que, novamente, envolve a correção na aplicação dos princípios da Orientação a Objetos (i.e. coesão, desacoplamento, encapsulamento, organização e reutilização), o número de classes, o número e a forma de relacionamento, a complexidade algorítmica etc.
- Qualidade do relatório escrito e da apresentação.

CONSIDERAÇÕES PESSOAIS

Caso o(s) estudante(s) deseje(m) expressar algum sentimento relativo, por exemplo, aos aprendizados e dificuldades encontrados, isto deve ser feita nesta seção opcional. Nesta seção pode-se até utilizar primeira pessoa, entretanto seria melhor a forma impessoal.

Neste sentido, **todas as demais seções devem ser escritas de forma impessoal** (o que significa não usar primeira e segunda pessoa singular ou plural – em suma não usar “eu” ou “nós” no texto). Ainda, salienta-se que o trabalho deve ser redigido linguagem correta, na forma culta e sem exageros poéticos, com textos não prolixos e bem encadeados.

DIVISÃO DO TRABALHO

Esta seção deverá ter uma tabela salientando quem desenvolveu cada classe/módulo do *software* e realizou as demais atividades como as de ‘engenharia de *software’*, a redação do trabalho escrito, a revisão da redação do trabalho e a preparação da apresentação do projeto.

Tabela 4. Lista de Atividades e Responsáveis.

Atividades.	Responsáveis
Levantamento de Requisitos	Fulano e Ciclano
Diagramas de Classes	Fulano e Ciclano
Programação em C++	Fulano e Ciclano em geral
Implementação de *Template*	Fulano

Implementação da Persistência dos Objetos...	Ciclano
...	
Escrita do Trabalho	Mais Fulano que Ciclano
Revisão do Trabalho	Mais Ciclano que Fulano

AGRADECIMENTOS

Havendo agradecimentos de ordem profissional, estes deverão vir antes das referências. Aqui se pode salientar e agradecer caso outra equipe tenha revisado o trabalho.

REFERÊNCIAS

Nesta seção devem ser listadas as referências citadas ao longo do texto. Seguem exemplos:

[1] DEITEL, H. M.; DEITEL, P. J. C++ Como Programar. 5ª Edição. Bookman. 2006.

[2] STADZISZ, P. C. Projeto de Software usando UML. Apostila CEFET-PR 2002. http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/EngSoftware/Apostila%20UML%20-%20Stadzisz%202002.pdf

[3] SIMÃO, J. M. Site da Disciplina de Fundamentos de Programação 2, Curitiba – PR, Brasil, Acessado em 09/09/2015, às 15:15: http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/Fundamentos2.htm.

[4] SIMÃO, J. M. Trabalho para a Disciplina de Fundamentos de Programação 2 do Curso de Engenharia Eletrônica da UTFPR: Modelo para Composição do Trabalho. Acessado em 29/02/2016: http://ppgca.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos2/TopicosTrab/Modelo_para_Trabalho_FundamentosProgramacao2_2015_09_10.pdf.