UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA
INF01120 - TÉCNICAS DE CONSTRUÇÃO DE PROGRAMAS
PROFESSOR: Eduardo Nunes Borges (*enborges@inf.ufrgs.br*)
Semestre 2009/2
Enunciado do Trabalho Prático

Formar grupos de alunos (no máximo 3 membros). Trabalhos individuais podem ser aceitos, mas a sua aceitação deverá ser analisada caso a caso (explique suas razões).

Datas importantes

Entrega da lista de membros do grupo: até 29 de setembro.
Entrega do Trabalho: Documentos e código-fonte: até 19 de novembro.
Demonstração e defesa: nas aulas dos dias 24 e 26 de novembro.

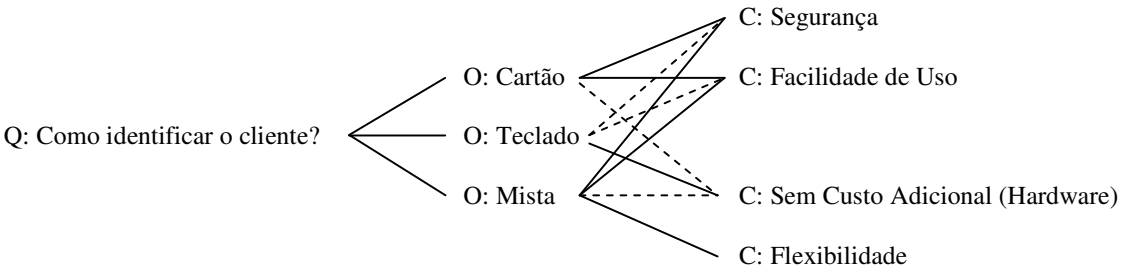
Introdução

O objetivo do trabalho é a definição, implementação, teste e depuração de um editor de modelos QOC (*Questions*, *Options*, *Criteria*), um modelo de *design rationale*, que serve para estruturar e documentar as decisões (e suas justificativas) dos membros da equipe de desenvolvimento de software. QOC é basicamente um modelo hierárquico onde **Questões** referem-se às principais decisões a serem tomadas no decorrer do desenvolvimento; **Opções** referem-se a possíveis alternativas a considerar para cada decisão; e **Critérios** referem-se a razões para adoção (critério positivo) ou rejeição (critério negativo) de cada uma das alternativas enumeradas em Opções. A bibliografia básica inclui:

- Moran, T.; Carroll, J. *Design Rationale: Concepts, Techniques and Use*. HillsDale, NJ: Lawrence Erlbaum, 1994.
- MacLean, A; Young, R.; Bellotti, V.; Moran, T. *Questions, Options and Criteria: Elements of Design Space Analysis*. In: Human-Computer Interaction 6(3-4): 201-250, 1991.

A figura abaixo mostra um exemplo de representação gráfica do modelo QOC.

Sistema: Automação Bancária
Subsistema: Software do Terminal do Usuário



Atividades

O trabalho pode ser dividido em 5 etapas:

1ª Etapa

Elaborar uma lista (informal, em português) dos requisitos que o sistema deve possuir no ponto de vista dos autores do projeto. Cada requisito é um item da lista e pode ser do tipo funcional (relativo a uma funcionalidade do sistema) ou não-funcional (relativo a fatores de qualidade como desempenho e usabilidade ou ainda características como cronograma, orçamento, limitações tecnológicas, entre outras). Cada requisito deve ser explicitamente rotulado como FUNCIONAL ou NÃO-FUNCIONAL.

2ª Etapa

PROJETAR a parte funcional do sistema, definindo Classes correspondentes a Tipos Abstratos de Dados que fazem parte do programa. Se for utilizada alguma abordagem não vista em aula, um resumo desta deve ser entregue ao professor junto com a documentação do trabalho, assim como a lista de referências bibliográficas consultadas sobre esta abordagem.

3ª Etapa

PROJETAR a interface do sistema com o usuário. Estabelecer o perfil básico dos usuários do sistema e as tarefas principais que serão suportadas. A documentação deverá incluir uma impressão de cada tela ou unidade de apresentação por página, seguida das explicações e argumentações que justifiquem seu *layout* e seu comportamento.

4ª Etapa

IMPLEMENTAR um protótipo do sistema projetado nas partes anteriores. Não é definida *a priori* a tecnologia a ser usada. O grupo deve escolher uma linguagem orientada a objetos, preferencialmente alguma das vistas em aula, e uma plataforma adequada para o desenvolvimento do software. É extremamente recomendado o desenvolvimento de uma interface gráfica do usuário (GUI) ao invés de uma interface de linha de comando. Existem diversas bibliotecas gráficas (GTK, QT, Swing, etc.) disponíveis para a construção de janelas, ícones, botões, menus, figuras, etc. Reuso de algum código existente é tolerado, mas DEVE SER SINALIZADO na documentação. Implementar o protótipo com o máximo possível de funções que foram projetadas.

5ª Etapa

TESTAR e DEPURAR o que foi implementado antes de demonstrar ao professor. A atividade de teste inclui a especificação de casos de teste funcional de unidade. Esse conjunto de casos de teste deve ser entregue junto da documentação.

6ª Etapa

APRESENTAR as características principais de seu projeto e DEMONSTRAR o sistema para o professor e a turma (20 minutos no MÁXIMO por apresentação). A documentação final do trabalho será entregue ao professor alguns dias antes da demonstração. O professor não apenas fará uso da documentação para utilizar o sistema e avaliá-lo, mas também poderá questionar oralmente os componentes do grupo a respeito de todas as partes do trabalho (5 minutos por apresentação). Deve ficar clara nas respostas a participação e o engajamento de todos os indivíduos no trabalho do grupo.

Observações e Avaliação

O trabalho a ser entregue inclui a documentação associada a TODAS as partes e deve conter também a identificação do grupo (número e nome de todos componentes) e todas as suposições feitas durante a realização do trabalho.

Os critérios de avaliação incluem: especificação de requisitos, decomposição de software OO, projeto das classes, projeto da interface do usuário, implementação, especificação dos casos de teste, uso dos conceitos de modularidade (principalmente no que diz respeito a baixo acoplamento, alta coesão e encapsulamento), uso de convenções de programação e apresentação.

Definições mais precisas que porventura sejam necessárias serão acrescentadas no decorrer do semestre. Em caso de dúvidas, consulte inicialmente a bibliografia básica, depois pesquise em outras bibliografias (use a biblioteca e a Internet) e, em caso de necessidade, consulte o professor via e-mail. A disciplina prevê algumas aulas para acompanhamento dos projetos e esclarecimento de dúvidas. Bom trabalho!

ATENÇÃO!

O plágio é terminantemente proibido e a sua detecção incorrerá na divisão da nota obtida pelo número de alunos envolvidos! Para detectar o plágio, além da análise detalhada do código-fonte de todos os trabalhos, será utilizado o software MOSS (http://theory.stanford.edu/~aiken/moss).