Mestrado em Informática
UCE30-EL, Engenharia de Linguagens

Projecto Integrado

Ano Lectivo de 2010/11

1 Objectivos

Com este projecto integrado pretende-se sedimentar os conhecimentos introduzidos nas aulas teóricas dos 4 módulos desta UCE: Engenharia Gramatical, Processamento Estruturado de Documentos, Análise e Transformação de Software e Scripting no Processamento de Linguagem Natural.

2 Resultados da aprendizagem

No final do projecto, os alunos devem ter adquirido um conjunto de competências tecnológicas, específicas e genéricas detalhadas na rubrica "Resultados de Aprendizagem"da página oficial da disciplina, em
`http://wiki.di.uminho.pt/twiki/bin/view/Mestrado/EL`.
Nomeadamente,

- Capacidade para desenvolver especificações da sintaxe/semântica de linguagens com gramáticas de atributos.
- Capacidade para gerar programas usando ferramentas automáticas baseadas em gramáticas de atributos.
- Capacidade para gerar ou utilizar ambientes de desenvolvimento estruturais e orientados à semântica.
- Capacidade para representar, armazenar e manipular Conhecimento com eficiência.
- Capacidade de escrever scripts para automatização de uma variedade de tarefas e transformações.
- Capacidade de resolver problemas usando transformações via Expressões Regulares.
- Capacidade de compreender as vantagens e o funcionamento de sistemas guiados por regras de produção (condição-reacção).
- Capacidade de construir Linguagens de Domínio Específico (DSLs) concretas, bem como produzir com eficiência eficientes processadores.
- Capacidade de construir e usar corpora.
- Capacidade de extrair informação diversa a partir de corpora.
- Capacidade de construir dicionários electrónicos.
- Capacidade de construir pequenos protótipos para modelar linguagem natural.
- Capacidade de construir front-ends poderosos para a análise de linguagens de programação.
- Capacidade de desenvolver software como uma tarefa de transformar, de forma sistemática eficiente, programas.

- Capacidade de utilizar métricas e técnicas de transformação de programas para optimizar programas (e.g. cálculo parcial, detecção de código morto), efectuar debugging de programas (e.g. slicing), melhorar a estrutura dos programas (e.g. refactoring).
- Capacidade de definir testes para software e testar automaticamente programas em diferentes linguagens de programação.
- Capacidade para criar representações visuais adequadas à compreensão clara do conhecimento complexo detido.
- Capacidade de compreender o ciclo de vida dos documentos estruturados, sabendo identificar as várias fases e as tecnologias a utilizar em cada uma.
- Capacidade de especificar uma linguagem de anotação para um conjunto de requisitos.
- Capacidade de implementar transformações de documentos para diversos fins: extracção de conhecimento, publicação na Web, intercâmbio de informação, ...
- Capacidade de utilizar soluções de armazenamento para documentos anotados.
- Capacidade de definir as camadas necessárias para integrar e realizar o intercâmbio de informação entre sistemas de informação distintos.
- Capacidade de implementar um projecto de publicação electrónica recorrendo a normas internacionais abertas: XML, XSLFO e XSL.
- Capacidade de programar a geração automática de sítios Web a partir de um repositório de documentos XML.
- Capacidade de utilizar linguagens de anotação e respectivas ferramentas desenvolvidas por outrém.

3 Organização e Funcionamento

O projecto será desenvolvido em grupos de 2, fora das aulas, em ambiente Linux (este ano será o único ambiente de trabalho oficial).
A única aula presencial de 1h/semana será usada para:

- Apresentação do projecto (no todo e em cada uma das suas fases), discussão do enunciado e identificação dos requisitos, alinhamento de estratégias e planeamento da resolução.
- Formação teórica básica em temas tecnológicos transversais aos 4 módulos e essenciais ao desenvolvimento do projecto, com proposta de temas a investigar.
- Gestão dos projectos; Apresentações Intercalares (avaliação); e Seminários.

No fim de cada semestre e no fim do ano lectivo, cada grupo apresentará à equipe docente o trabalho realizado e os resultados obtidos, devendo entregar um relatório técnico de desenvolvimento devidamente estruturado e fundamentado, escrito em LaTeX.

3.1 Calendarização

O projecto deve ser executado ao longo de todo o ano estando a **entrega final** agendada para a `27/06/2011`.
Para controlo da situação e avaliação intermédia, haverá 3 apresentações intercalares do projecto, que coincidem com o meio do 1º semestre, o início do 2º semestre e o meio do 2º semestre. Mais concretamente nos dias: `13/12/2010`, `07/03/2011` e `02/05/2011`.

4 Enunciado

O que se pretende neste projecto é desenvolver um Software para Análise e Avaliação de Programas, que doravante designaremos por SAAP.
O objectivo principal é criar um ambiente de trabalho, com um interface Web, que permita a docentes/alunos avaliar/submeter programas automaticamente. Este ambiente Web terá um funcionamento parecido com o bem conhecido programa `Moshak`.
O programa SAAP deverá permitir a docentes e alunos registarem-se na sua base de dados (através de um login/password) com diferentes tipos de permissões.

- Os docentes terão possibilidade de criar um exercício, submetendo para tal um enunciado e um conjunto de dados de entrada/saída que serão usados na avaliação do software submetido pelos alunos (isto é uma *bateria de testes*).
- Os alunos podem associar-se a um exercício já submetido por um docente, e submeterem soluções (isto é, programas em código fonte) possíveis para esse exercício.

Neste momento, o software SAAP deverá analisar o programa submetido pelo aluno e fazer várias tarefas, nomeadamente:

- Verificar se o código fonte proposto compila (ou não). No caso do programa não compilar deve indicar qual/quais os erros encontrados. Esta mensagem de erro deve ser
- No caso do programa compilar, o SAAP deve executá-lo e verificar se para os dados de entrada submetidos pelo docente, o programa produz o resultado esperado (isto é, os respectivos dados de saída).
- O [SAAP deve ainda dar indicação ao aluno e ao docente da qualidade do software. Assim, deverá não só usar os resultados obtidos no item anterior, mas também calcular métricas de software sobre o program fonte submetido pelo aluno. Várias métricas que dão uma boa noção da qualidade do código fonte serão estudadas.
- O SAAP deve ainda guardar todo historial de cada aluno de modo ao docente poder analisar como ele evoluiu no decorrer do desenvolvimento do projecto.

4.1 Tarefas a Desenvolver

Para concretizar o trabalho é necessário realizar as seguintes tarefas, algumas das quais podem prosseguir em paralelo:

1. Analisar o problema e conceber o sistema desenhando a sua arquitectura.
2. Conceber e Implementar a Interface do sistema com os utilizadores, usando como base um sistema Web.
3. Modelar os dados do problema.
4. Implementar a Base de Dados.
5. Estudar a metodologia escolhida para analisar a qualidade do código fonte dos programas.
6. etc....