DANIEL MATUKI DA CUNHA
EDUARDO MIYOSHI KASA
PATRÍCIA AKEMI IKEDA

MAC 499
TRABALHO DE FORMATURA SUPERVISIONADO

CORRETOR AUTOMÁTICO DE EXERCÍCIOS PROGRAMA PARA AUXÍLIO NO ENSIO À DISTÂNCIA

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

ORIENTADOR
CARLOS HITOSHI MORIMOTO

Resumo

O presente trabalho se refere a um projeto sugerido pelo professor Carlos Hitoshi, desenvolvido durante o ano de 2006, que compreendeu o desenvolvimento de um sistema para corrigir e auxiliar alunos de cursos introdutórios à computação, na realização de Exercícios Programa (EPs).

O volume muito grande de alunos que fazem esses cursos dificulta o ensino através de atividades práticas como os EP, pois eles demoram a ser corrigidos e os alunos ficam sem saber o que acertaram ou o que erraram antes de fazerem uma prova sobre o tema.

Criar esse sistema é muito importante para facilitar o ensino e o aprendizado desses alunos.

Desenvolvemos um sistema (ainda incompleto) que automatiza essa tarefa e ajuda o aluno a ver onde está cometendo erros.

Foram criados quatro módulos: compilador, testador, corretor e a integração com o moodle, que serao discutidas mais para frente.

São Paulo
Dezembro/2006

Sumário

Parte I
O Projeto

1.0 Introdução

Todos os anos, cursos introdutórios à computação são ministrados em diversas unidades da USP, totalizando, em média, mais de 2000 alunos.

Para a maior parte destes alunos, essa é uma matéria completamente diferente do que estavam acostumados a estudar, tanto no colégio quanto no cursinho, tornando seu aprendizado mais difícil. Ficar apenas nas aulas teóricas não é suficiente para capacitar um aluno de escrever suas primeiras linhas de código, sendo fundamental a resolução de EPs. É na prática que as maiores dificuldades são detectadas, sendo necessário o auxílio de monitores e professores.

O grande volume de EPs para serem corrigidos e o número reduzido de monitores inviabiliza um atendimento mais personalizado. Mais difícil ainda é entregar a correção e as notas em tempo hábil. Passado algumas semanas, a maior parte dos alunos não lembra mais os pontos em que tinha dúvida, tampouco entendem a correção feita pelo monitor.

A falta de comunicação entre o aluno e o monitor desmotiva. Quando a correção é liberada, outras atividades já estão em curso (por exemplo, EPs mais elaborados) e, muito provavelmente, os mesmos erros serão cometidos.

Com um corretor automático, os casos mais fáceis são resolvidos rapidamente, evitando que o monitor perca tempo olhando o código daqueles que conseguiram resolver o problema.

2.0 Conceitos e Tecnologias Estudadas

2.1 Moodle

O Moodle é um sistema de gerenciamento de cursos (Course Managemente System - CMS) - um pacote de software livre (GNU-GLP) com princípios pedagógicos, podendo rodar em qualquer sistema que suporte a linguagem PHP.

É um sistema composto por módulos, o que permite ser adaptável de acordo com as necessidades de cada curso, por isso ele tem sido adotado em diversos países com sucesso.

Com o intuito de auxiliar no aprendizado, o sistema Moodle foi implantado na USP, facilitando a comunicação entre alunos e professores, além de disponibilizar material referente às aulas.

2.2 Exercício Programa

A teoria passada em sala de aula nem sempre é absorvida pelos alunos, pelo fato de serem conceitos muito abstratos. A melhor alternativa são os exercícios práticos passados pelo professor, ajudando não apenas a absorver melhor os conceitos vistos na aula, como a aprender coisas novas, como por exemplo, erros de lógica que são percebidos apenas ao executar um programa.

Um Exercício Programa é uma tarefa prática que focaliza alguns tópicos da matéria que está sendo ensinada pelo professor. Geralmente são passados entre 3 e 4 Eps por semestre, e os alunos tem em média, 4 semanas para concluir cada um. Tendo em vista um aprendizado mais completo, alguns professores permitem que duplas sejam formadas na resolução do EP, contribuindo para a discussão do problema e o desenvolvimento da melhor solução.

2.3 Java

Para um corretor de Exercícios Programa em C, seria mais lógico que fizéssemos o projeto em C ou C++, mas infelizmente não tínhamos conhecimento suficiente para implementar nessa linguagem. Devido à maior familiaridade com Java, uma linguagem orientada a objetos criada pela Sun Microsystems, a escolhemos para a realização deste projeto.

A grande vantagem do Java é que, por ser orientado a objetos e com um bom desenho de projeto, é possível criar um sistema altamente flexível, extensível, de fácil manutenção e entendimento. A grande variedade de APIs (*application programming interface*) para Java é que a torna mais interessante para ser utilizada, além é claro, da portabilidade entre diferentes arquiteturas de CPU.

2.4 JavaCC

Java Compiler Compiler[5] (JavaCC) é um programa que gera analisadores. Um gerador de analisadores é uma ferramenta que lê uma gramática e converte em um programa Java que pode reconhecer essa gramática.

Um analisador divide uma entrada de texto em pequenas partes para que seja possível analisá-la. Em nosso contexto, o JavaCC foi utilizado para gerar um programa que lê arquivos em C para que possamos extrair as funções e poder testá-las, além de ser possível verificar possíveis erros no código.

2.5 XML

XML (eXtensible Markup Language) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet, mas podendo ser utilizado para outros fins, como o nosso.

2.6 FindBugs

FindBugs[4] é uma ferramente escrita em Java que busca por padrões de bugs em códigos-fonte Java. É uma ferramente muito importante pois muitos bugs passam desapercebidos pelos programadores, por melhores que eles sejam.

Um exemplo de padrão de bug em C seria:

```
for (;;;) {} // Claramente aqui ocorre loop infinito
```

3.0 Atividades Realizadas

Para facilitar a criação, manutenção e extensibilidade, o projeto foi divido em quatro módulos independentes: o **Compilador**, o **Testador**, o **Corretor** e o módulo de **Integração com o moodle**, sendo que os três primeiros foram escritos em Java 5 e o último em PHP.

Os módulos seguem o fluxo usual de correção de um EP, isto é, o programa é compilado, depois testado, tem seu código verificado e por último é colocado via web para o aluno ter acesso aos erros cometidos.

3.1 Compilador

O objetivo do compilador é verificar se é possível compilar o EP e, se for possível, detectar possíveis problemas em tempo de compilação.

Neste primeiro módulo, o GCC foi utilizado na compilação dos programas em C, gerando mensagens de *errors* e *warnings*, que serão utilizadas pelo testador.

Mensagens de *errors* (ou erros) são causadas por problemas graves e que impossibilitam a execução do programa, enquanto que *warnings* (ou avisos) são causados por problemas que aparentemente não interferem na execução, mas podem causar efeitos colaterais.

As mensagens podem ser traduzidas em uma linguagem mais amigável para facilitar a interpretação do aluno iniciante. Tendo em vista de que esta é uma ferramenta para auxiliar na introdução à programação, os respectivos *erros* e *avisos* gerados pelo compilador também são mostrados na tela (além daqueles que não possuem uma tradução), contribuindo para que os alunos se familiarizem com o ambiente do GCC no processo de correção de erros do programa.

O módulo foi escrito de forma a permitir a troca de um compilador por outro, com a modificação de apenas uma linha do sistema (por exemplo, caso o ambiente mude do Linux para o Windows).

3.1.1 GCC.XML

Para permitir a adição e modificação das mensagens de maneira prática, estas são colocadas em um arquivo XML (no caso, gcc.xml, pois o compilador utilizado foi o GCC do Linux), formando um banco de dados que pode ser estendido de acordo com a necessidade de cada matéria e/ou professor.

Nesta figura, temos um exemplo simples do arquivo XML, com uma mensagem de *warning* gerado pelo GCC:

```
<warning>
  <mensagem>might be used uninitialized</mensagem>
  <descricao>
O compilador está avisando que talvez uma variável está sendo
usada sem ter sido inicializada.
Pode ser que a variável esteja sendo inicializada dentro de um
while ou if e o compilador não consegue detectar isso.
É bom dar uma olhada em seu programa para ter certeza de que não
há erro de lógica.
  </descricao>
</warning>
```

3.2 Testador

Este módulo verifica se, dado uma entrada, a saída está de acordo com o esperado, especificado em um arquivo XML (por exemplo, gabarito.xml), feito pelo monitor/professor da disciplina.

Preferencialmente, no arquivo XML são colocados entradas/saídas que, dependendo do Exercício Programa, são mais críticas e dão maior certeza de que os testes estejam corretos. Por exemplo, em um programa de cálculo de fatorial, seriam testados os seguintes valores:

Entrada	Saída
-1	Não é possível calcular fatorial de número negativo
0	1
1	1
2	2
3	6
8	40320

O arquivo XML para o dois fatorial ficaria assim:

```
<teste>
  <nome>Teste de Fatorial 2</nome>
  <descricao>Testa o 2!</descricao>
  <nota>25</nota>
  <entrada>2</entrada>
  <saida>2</saida>
</teste>
```

E se repetiria para os demais valores de entrada.

3.2.1 Segurança

Segurança é um assunto muito importante para uma ferramenta automática, pois como não há a análise por parte de uma pessoa, códigos maliciosos passam despercidos. Por enquanto, a única forma de segurança encontrada, sem afetar performance, foi apenas a busca pela palavra-chave **System**, que permite executar, dentro de um programa em C, funções existentes no sistema, como por exemplo o **rm**.

Idealmente seria necessário criar um shell seguro que isole a execução dos teste em um ambiente onde o que se encontra é apenas o mínimo necessário para a execução de programas em C, mas infelizmente esse processo ainda é lento.

3.3 Corretor

O Corretor, através do resultado do módulo de testes e por análise do código fonte, buscando por padrões de bugs, gera um relatório de problemas encontrados no EP.

A análise do código ainda está incompleta, porém ainda é necessário uma maturidade maior do analisador criado pelo JavaCC.

3.4 Integração com o Moodle

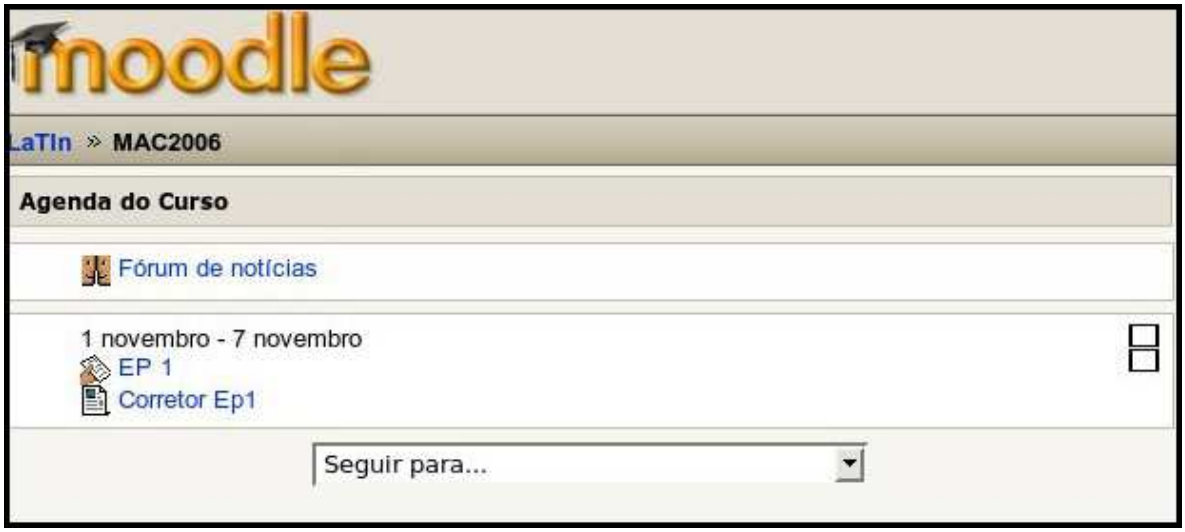
Para a integração com o Moodle, desenvolvemos um módulo de atividades (activities module) seguindo o padrão estabelecido pelo o sistema. Este módulo foi desenvolvido utilizando a linguagem PHP, norma estabelecida pelo sistema, e um professor pode adicioná-lo a uma matéria da mesma forma que é feito com o módulo de entrega de trabalhos, mantendo o padrão do sistema Moodle.

3.4.1 Interfaces

As interfaces web foram desenvolvidas procurando sempre manter a simplicidade de utilização. Seguindo alguns exemplos como a interface do módulo de entrega de tarefas e também a interface de saída de erros do programa Eclipse, conseguimos fazer com que o módulo corretor seja fácil de utilizar e entender.

A interface é dividia em três partes, envio da tarefa, envio do gabarito e resultados dos testes. Abaixo temos alguns exemplos de utilização da interface descrita.

Exemplo 1: Exibição do módulo



Neste exemplo podemos verificar como o módulo fica disponível para o aluno após ser adicionado pelo professor. Veja que o módulo corretor neste caso ("Corretor Ep1") é relativo ao primeiro Exercício Programa, sendo que podemos ter várias instâncias do módulo corretor, cada uma correspondente a um exercício programa diferente.

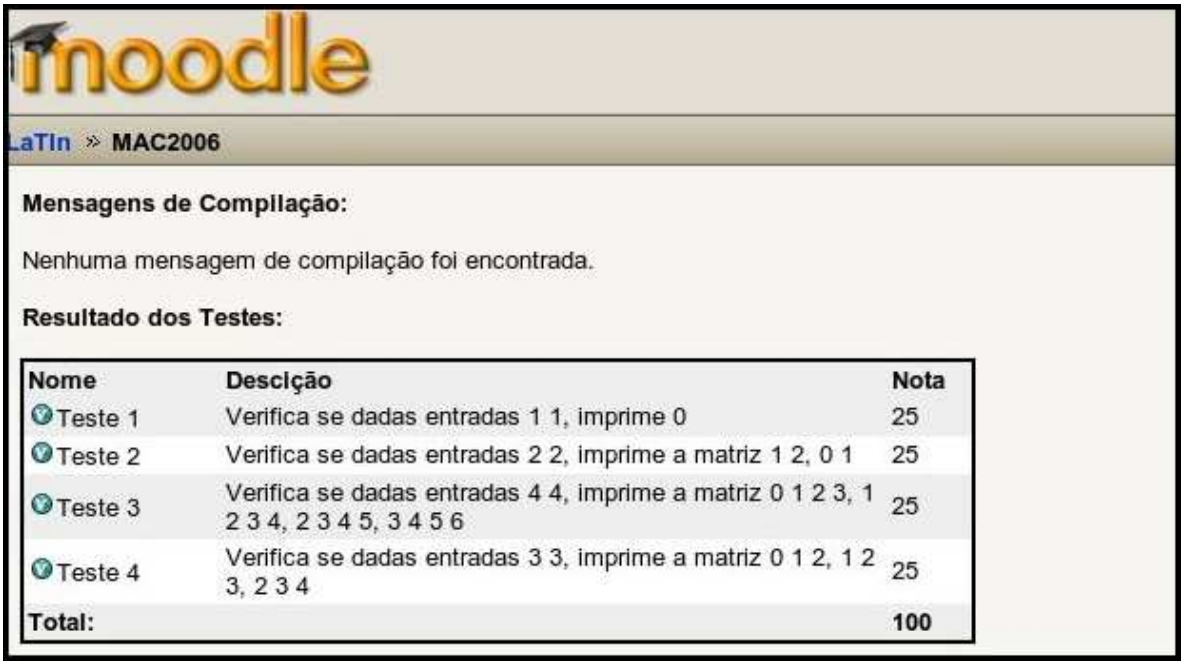
Exemplo 2: Envio da Tarefa

Veja que este exemplo o aluno envia um programa para a correção da mesma forma que é feita no módulo de entrega de tarefas, facilitando a sua utilização pelo aluno, que não precisa aprender a utilizar uma nova interface.

Exemplo 3: Envio de Gabarito

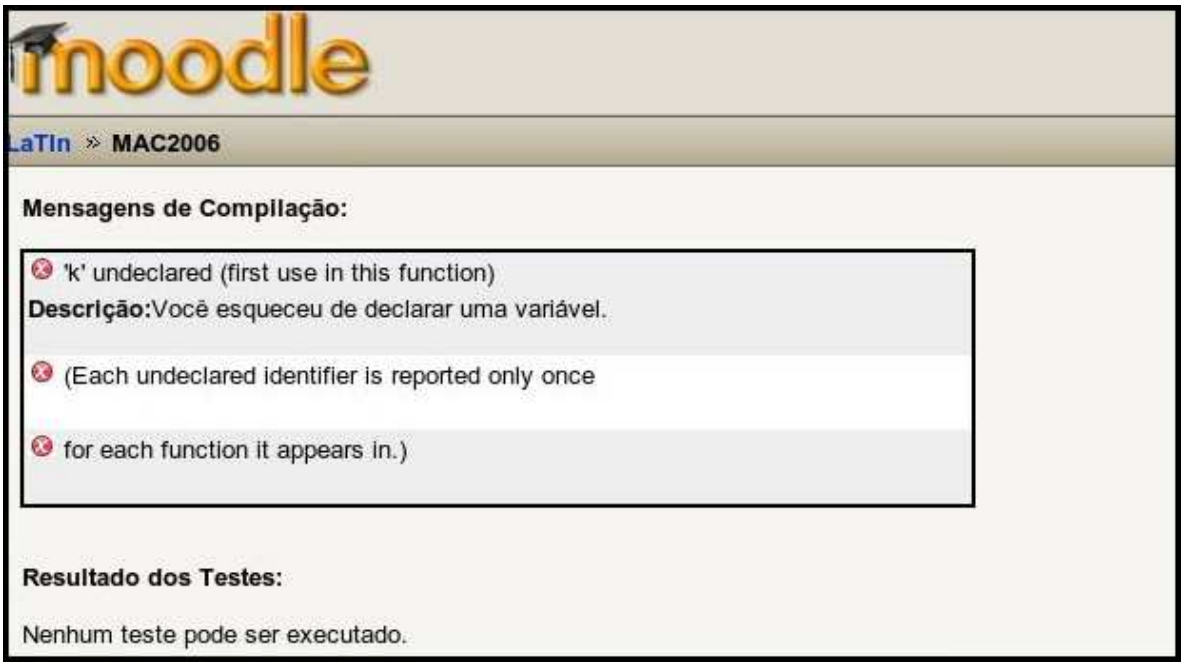
A interface acima é exibida somente quando o usuário tem uma conta especial no Moodle (professor ou monitor), e nela é possível enviar um arquivo que será utilizado como gabarito para as correções á partir do momento de envio do arquivo XML.

Exemplo 4: Saída da correção



A interface acima é exibida imediatamente após o usuário enviar um código fonte para a correção. Nesta interface o usuário pode verificar o resultado dos testes executados na sua tarefa, uma descrição dada pelo professor para cada teste e a nota obtida em cada teste. As informações sobre cada teste devem ser adicionadas pelo professor ou monitor no arquivo de gabarito cadastrado.

Exemplo 5: Mensagens de compilação



Na interface acima pode ser observada a saída exibida quando o programa contém erros de compilação. A tabela "Mensagens de Compilação" exibe as mensagens originais do compilador e em seguida uma descrição dada pelo professor para cada mensagem de compilação.

3.4.2 Estrutura do Módulo

O módulo corretor é composto por duas partes, a parte PHP que implementa as funções do módulo e a parte SQL que executa a criação de uma tabela no banco de dados MySQL do sistema Moodle.

Na parte SQL foi criada uma tabela para guardar as informações de cada instância do módulo, lembrando que cada vez que um professor adiciona um módulo corretor na sua matéria é criada uma nova instância deste módulo, guardando os nomes, descrições, identificadores e outras informações.

Na parte PHP foram criadas classes para tratar as quatro principais funções do módulo:

3.4.2.1 Envio de arquivos

A funcionalidade de envio de arquivos, utilizada quando é enviada uma tarefa ou um gabarito, foi criada utilizando as funções já implementadas pelo sistema Moodle. Essas funções fazem uma série de verificações no arquivo que está sendo enviado como por exemplo a verificação de arquivos nocivos, e guardam os arquivos em uma área em disco reservada para o moodle (Moodle file dir).

3.4.2.2 Armazenamento do gabarito

Para cada vez que um professor adicionar o módulo corretor em sua matéria é necessário armazenar um gabarito diferente para efetuar a correção. Para implementar essa funcionalidade utilizamos o identificador de cada instância do módulo para criar uma área em disco para guardar arquivos específicos como o gabarito.

3.4.2.3 Execução da correção

Para implementar a execução do sistema corretor Java a partir do módulo PHP foram estudadas algumas possibilidades (descritas no tópico 1.4.3) e foi decidida a utilização da função *shell_exec* do PHP que executa um programa na linha de comando utilizando o usuário www-data.

3.4.2.4 Criação das Interfaces dinâmicas

As interfaces dinâmicas do módulo são aquelas que são criadas a cada execução da correção (descritas nos exemplos 4 e 5 do tópico 1.4.1). A criação dessas interfaces foi implementada executando uma interpretação da saída da execução do sistema corretor Java, e assim gerando uma página HTML formatada para facilitar a visualização das informações.

3.4.3 Integração PHP X Java

A questão da integração PHP x Java foi levantada quando verificamos que somente módulos escritos na linguagem PHP podem ser adicionados no sistema Moodle. Procuramos então executar uma pesquisa para analisar quais as formas que essa integração poderia ser implementada e descobrimos as seguintes:

3.4.3.1 Applet Java

Esta opção permite que um programa Java seja criado segundo certos padrões, podendo se adicionado em uma página HTML utilizando tags especiais.

Descartamos esta opção pois verificamos que o aluno que fosse utilizar o corretor automático obrigatóriamente teria que instalar um plugin para executar uma Applet Java. Também questionamos a opção pois uma programa Java rodando como Applet possui uma série de restrições de segurança que dificultam o seu desenvolvimento, como por exemplo a leitura de arquivos.

3.4.3.2 Extensões Java do PHP

O sistema PHP implementa a extensão Java (java extension) permitindo que Objetos Java sejam traduzidos e utilizados a partir de um código PHP. Descartamos esta opção por verificar que sua implementação esta descrita pelo fornecedor (http://www.php.net/manual/en/ref.java.php) como experimental e sujeita a mudanças sem prévia notificação. Também verificamos que muitos usuários que optaram por essa solução tiveram problemas em funções básicas do Java, classificando esta opção como instável.

3.4.3.3 Função shell_exec

Esta função implementada no PHP permite que programas em linha de comando sejam executados utilizando o usuário *www-data* (exemplo: *shell_exec("java HelloWorld")*). A grande desvantagem de utilizar esta função está na questão da segurança, porém podemos contornar se tomarmos as devidas precauções (veja no tópico 1.4.4). Escolhemos então esta opção pela facilidade de implementação e simplicidade, sendo que o esforço necessário para executar o corretor Java é mínimo.

3.4.4 Segurança do Módulo

A segurança de uma aplicação PHP é um tópico em que foi dedicada muita atenção e cuidado na criação do módulo corretor, pois uma aplicação PHP descuidada pode abrir muitas possibilidades para ataques a um servidor web.

Os pontos mais críticos que tentamos cobrir foram a passagem de parâmetros via URL, utilização da entrada de dados do usuário e a execução da função *shell_exec*.

Para não ser necessário o tratamento dos parâmetros enviados via URL, evitamos completamente a utilização deste método de troca de dados entre páginas, impossibilitando um ataque neste sentido.

No módulo corretor a entrada de dados do usuário é somente feita na entrega da tarefa e do gabarito. Nestes dois casos asseguramos a verificação dos dados utilizando as funções fornecidas pelo sistema Moodle. Estas funções executam diversas verificações tanto no nome dos arquivos quanto no conteúdo (arquivos nocívos), sendo que já foram amplamente testadas no módulo de entrega de tarefas por usuários do mundo todo.

Restou somente a execução da função shell_exec, que é acusada por abrir uma possibilidade para ataques. Esta função é considera insegura pois permite que um usuário execute um programa em linha de comando no servidor web utilizando a conta *www-data*. Porém o que torna ela realmente nociva é que muitos desenvolvedores PHP permitem que o parâmetro da função seja alterado pelo o usuário web veja o exemplo:

Suponha que o desenvolvedor por descuido criou o seguinte código PHP:

shell_exec("java HelloUser " + nome);

Caso o conteúdo da variável "nome" seja, de alguma forma alterável pelo usuário mal intencionado, então este poderia fazer com que "nome" receba o conteúdo " ; exploit ", quebrando a linha de comando e fazendo com que a função shell_exec execute um programa nocivo, atacando o servidor web.

A medida tomada para evitar ataques utilizando a função *shell_exec* foi impossibilitar que o usuário modifique o argumento desta função, ou seja, o argumento é definido pelo módulo corretor independentemente de qualquer ação do usuário.

4.0 Conclusões

O principal objetivo deste projeto é auxiliar os alunos no aprendizado da programação, funcionando como um sistema tutor. Esperamos que com isso, o aluno consiga direcionar melhor os estudos e tenha menos dificuldade em aprender a programar além de procurar ser menos dependente do professor e do monitor.

5.0 Trabalhos Futuros

- Java
  - Modificar o sistema para que corrigisse Exercícios Programa escritos em Java, pois é uma linguagem bastante utilizada pelos alunos de computação.

- Módulo Compilador
  - Testar outros compiladores além do GCC, para ter portabilidade.

- Módulo Testador
  - Testes unitários em cada função do Exercício Programa, contribuindo para uma melhor especificação dos erros cometidos e a nota atribuída ao aluno seria mais precisa.
  - Criar um ambiente seguro e rápido para a execução dos testes.

- Módulo Corretor
  - Procurar utilizar as idéias do FindBugs[4] para tentar encontrar possíveis erros no código e assim, mostrar dicas de correção aos alunos.

- Detecção de Cópia
  - Infelizmente o programa não detecta cópias de exercícios programa, pois é trabalhoso e demorado. É um assunto a ser estudado, pois o objetivo maior é ajudar o aluno a aprender a programar, sendo que o programa é apenas uma ferramenta que o aluno utiliza para verificar onde possui dificuldade.

Parte II
Experiência Pessoal

Desafios e Frustações

O grande problema encontrado foi a demora para a definição do projeto e, com isso, a falta de tempo.

No inicio fazíamos reuniões com o Hitoshi, mas infelizmente (ou felizmente para ele) ele foi viajar e as reuniões ficaram apenas pela troca de e-mails. Depois que ele foi viajar, vi como era importante as reuniões, pois ajuda bastante a tirar duvidas e ter idéias novas, o que por e-mail era bem mais difícil.

O projeto em si foi legal de se fazer, mas pela falta de tempo, ficou muito aquém do que eu gostaria e como agente rarmente conseguia reunir os três integrantes (todos trabalham), algumas partes foram feitas sem muita discussão.

Criar algo que foca mais no aluno do que no sistema foi um bom desafio, pois sempre tínhamos que parar para pensar como ele deveria ser avaliado, e se esta avaliação estava correta.

Disciplinas mais Relevantes

MAC0110 – Disciplina fundamental para aprender o básico da programação. Apesar de entrar no BCC sabendo alguma coisa (pois fiz técnico em Processamento de Dados), a professora Nami sempre tinha algo a mais para ensinar e corrigir.

MAC0122 – Provavelmente a disciplina mais importante e significativa que tive durante todos esses anos. O professor Paulo Feofiloff foi um excelente professor que me ensinou como ser um bom programador, e a importância de algoritmos claros e eficientes.

MAC0316 – Me ajudou a ver como as linguagens funcionam e a clareza e organização quase sempre é melhor que pura eficiência, além de ser divertido criar algorimos recursivos.

MAC0441 – Programação orientado a objetos é um paradigma interessante e é muito mais fácil que programação procedural. Criar sistemas orientado a objetos torna os programas de mais fácil entendimento e manutenção.

Conclusões Finais

Fazer o projeto foi uma experiência muito boa, e apesar do tempo ser curto, me ajudou a aplicar muitos dos conceitos vistos em aula, principalmente na área de programação. Criar a arquitetura do sistema foi um bom desafio, pois não fazia a mínima idéia de como iria ficar no final, e eu realmente gostei do resultado final, além de melhorar na aplicação de orientação a objetos que, pela simplicidade e praticidade, é um aspecto muito importante na minha formação.

A importância de um orientador que ajude e guie na hora das dúvidas foi muito importante para não me perder e desanimar.

O Instituto me deu uma base sólida sobre muitos aspectos de tecnologia e como estudar sozinho qualquer tema que me interesse ou seja necessário aprender.

Bibliografia

[1] Derek S. Morris – Automatic grading of student’s programming assignments: An interactive process and suite of programs.

[2] Urs von Matt – Kassandra: The automatic grading system.

[3] José Paulo Leal, Nelma Moreira - Automatic grading of programming exercises.

[4] David Hovemeyer and William Pugh - Finding Bugs is Easy - http://findbugs.sourceforge.net/

[5] Java Compiler Compiler - https://javacc.dev.java.net/

[6] www.moodle.org