

**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	5
1.0	INTRODUÇÃO.....6
2.0	CONCEITOS E TECNOLOGIAS ESTUDADAS.....6
2.1	<i>Moodle</i>6
2.2	<i>Exercício Programa</i>6
2.3	<i>Java</i>7
2.4	<i>JavaCC</i>7
2.5	<i>XML</i>7
2.6	<i>FindBugs</i>7
3.0	ATIVIDADES REALIZADAS.....7
3.1	<i>Compilador</i>7
3.1.1	<i>GCC.XML</i>8
3.2	<i>Testador</i>8
3.2.1	<i>Segurança</i>9
3.3	<i>Corretor</i>10
3.4	<i>Integração com o Moodle</i>10
3.4.1	<i>Interfaces</i>10
3.4.2	<i>Estrutura do Módulo</i>13
3.4.2.1	<i>Envio de arquivos</i>13
3.4.2.2	<i>Armazenamento do gabarito</i>13
3.4.2.3	<i>Execução da correção</i>13
3.4.2.4	<i>Criação das Interfaces dinâmicas</i>13
3.4.3	<i>Integração PHP X Java</i>13
3.4.3.1	<i>Applet Java</i>14
3.4.3.2	<i>Extensões Java do PHP</i>14
3.4.3.3	<i>Função shell_exec</i>14
3.4.4	<i>Segurança do Módulo</i>14
4.0	CONCLUSÕES.....15
5.0	TRABALHOS FUTUROS.....15
PARTE II	16
6.1	DESAFIOS E DIFICULDADES.....17
6.1.1	<i>Método de correção</i>17
6.1.2	<i>Planejamento do projeto</i>17
6.1.3	<i>Linguagem PHP</i>17
6.1.4	<i>Objetivo Inicial x Tempo</i>18
6.2	DISCIPLINAS RELEVANTES.....19

6.2.1	<i>Princípios de Desenvolvimento de Algoritmos</i>	19
6.2.2	<i>Laboratório de Programação I e II</i>	19
6.2.3	<i>Laboratório de Programação Extrema</i>	19
6.2.4	<i>Conceitos Fundamentais de Linguagens de Programação</i>	19
6.2.5	<i>Sistemas de Bancos de Dados</i>	19
6.3	INTERAÇÃO COM O ORIENTADOR.....	20
6.4	APLICAÇÃO DOS CONCEITOS VISTOS EM APLICAÇÕES REAIS	20
6.5	OUTROS PROJETOS	20

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 Management 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 ferramenta escrita em Java que busca por padrões de bugs em códigos-fonte Java. É uma ferramenta muito importante pois muitos bugs passam despercebidos 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 dividido 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

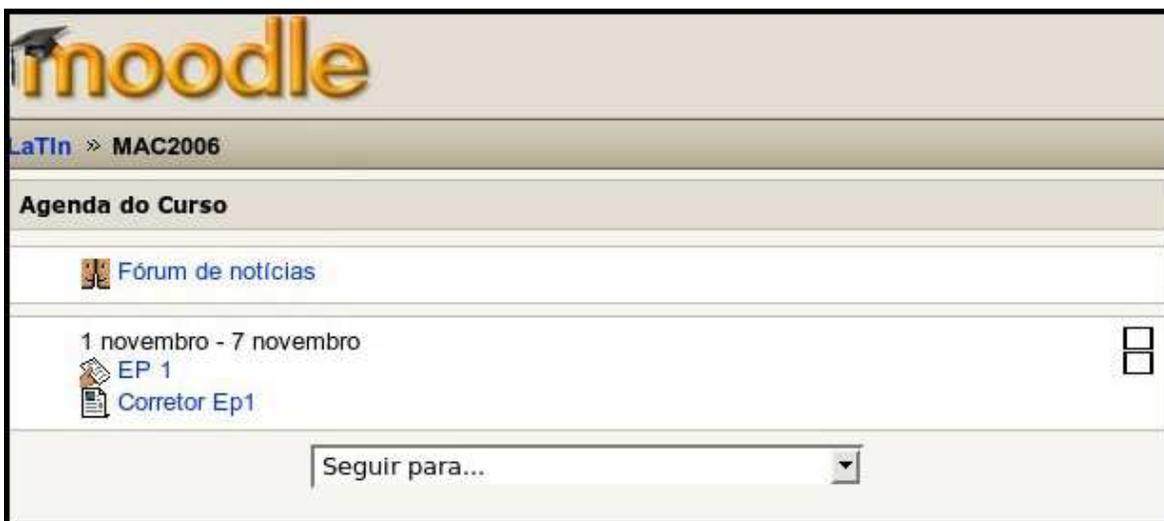
Para a integração com o Moodle, desenvolvemos um módulo de atividades (activities module) seguindo o padrão estabelecido pelo o sistema. Este módulo foi desenvolvido utilizando a linguagem PHP, norma estabelecida pelo sistema, e um professor pode adicioná-lo a uma matéria da mesma forma que é feito com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



The screenshot shows the Moodle interface for submitting a task. At the top, the Moodle logo is displayed. Below it, the breadcrumb navigation shows 'LaTin >> MAC2006'. The main heading is 'Enviar um arquivo'. There is a text input field containing the file path '/home/matuki/ep1.c'. To the right of the input field is a 'Browse...' button. Below the input field is a 'Corrigir Tarefa' button.

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.

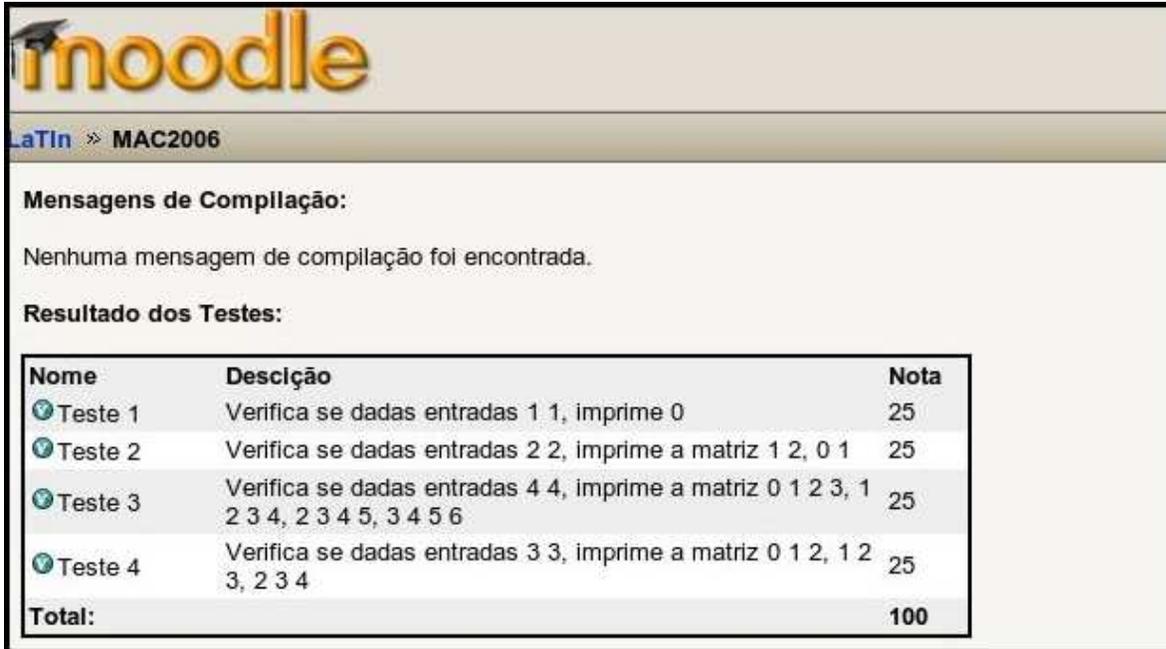
Exemplo 3: Envio de Gabarito



The screenshot shows the Moodle interface for submitting an answer key. At the top, the Moodle logo is displayed. Below it, the breadcrumb navigation shows 'LaTin >> MAC2006'. The main heading is 'Enviar um arquivo'. There is an empty text input field. To the right of the input field is a 'Browse...' button. Below the input field is a 'Corrigir Tarefa' button. The next heading is 'Submeter Gabarito'. Below it is another text input field containing the file path '/home/matuki/gabarito.xml'. To the right of this input field is another 'Browse...' button. Below this input field is an 'Enviar este arquivo' button.

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 a partir do momento de envio do arquivo XML.

Exemplo 4: Saída da correção

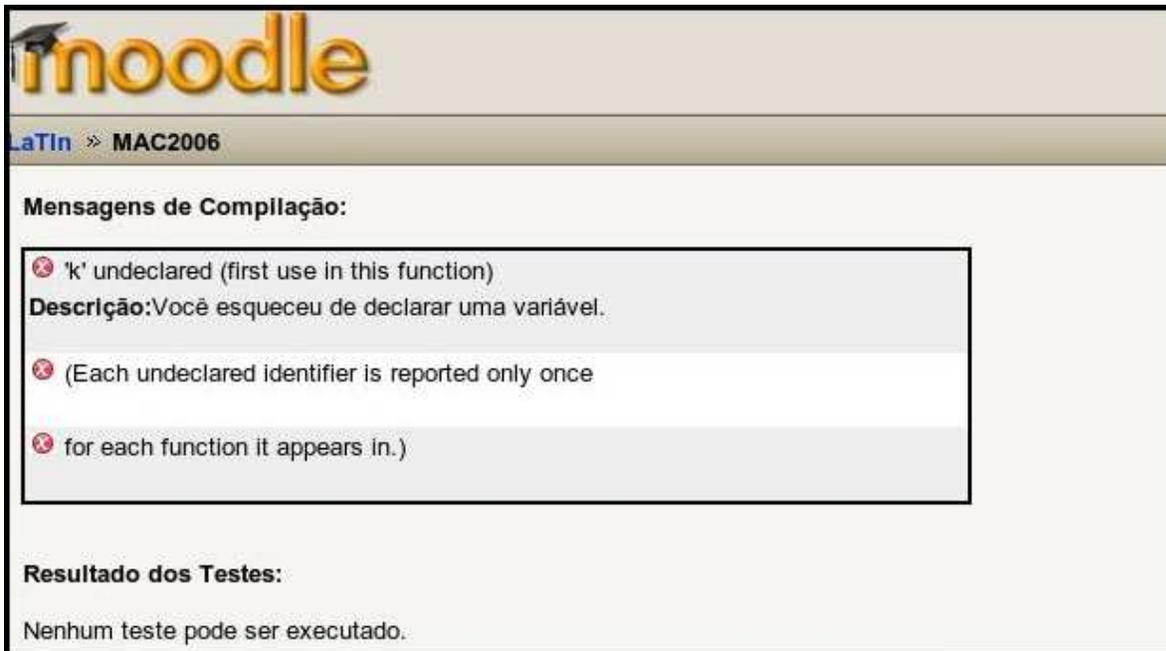


The screenshot shows the Moodle interface for a course named 'LaTin' and a task named 'MAC2006'. Under the heading 'Mensagens de Compilação:', there is a message: 'Nenhuma mensagem de compilação foi encontrada.' Below this, under 'Resultado dos Testes:', there is a table with the following data:

Nome	Descrição	Nota
✓ Teste 1	Verifica se dadas entradas 1 1, imprime 0	25
✓ Teste 2	Verifica se dadas entradas 2 2, imprime a matriz 1 2, 0 1	25
✓ Teste 3	Verifica se dadas entradas 4 4, imprime a matriz 0 1 2 3, 1 2 3 4, 2 3 4 5, 3 4 5 6	25
✓ Teste 4	Verifica se dadas entradas 3 3, imprime a matriz 0 1 2, 1 2 3, 2 3 4	25
Total:		100

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



The screenshot shows the Moodle interface for a course named 'LaTin' and a task named 'MAC2006'. Under the heading 'Mensagens de Compilação:', there are three error messages, each with a red 'x' icon:

- 'k' undeclared (first use in this function)
Descrição: Você esqueceu de declarar uma variável.
- (Each undeclared identifier is reported only once
- for each function it appears in.)

Below this, under 'Resultado dos Testes:', there is a message: 'Nenhum teste pode ser executado.'

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 obrigatoriamente 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 nocivos), 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 considerada 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 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

6.1 Desafios e Dificuldades

6.1.1 Método de correção

Logo ao iniciar o estudo para o projeto deparei com o maior desafio de todos: Encontrar o método mais apropriado para executar a correção de um exercício programa. Muitas dúvidas e possibilidades surgiram quando comecei a discutir com o professor orientador e os integrantes do grupo sobre o assunto, por exemplo:

- i) Como criar testes de entrada/saída para avaliar um programa?
- ii) Seria possível executar algum tipo de correção a nível de código?
- iii) Quais responsabilidades devemos assumir e quais devemos delegar para o professor ou o monitor ao corrigir um programa?

Ao estudar os projetos já existentes sobre o assunto (Referências bibliográficas [1] [2] [3]) pude tirar conclusões e analisar as possibilidades, podendo assim formar uma idéia mais clara do que precisava desenvolver.

6.1.2 Planejamento do projeto

Quando chegamos na etapa de planejamento do projeto, eu ainda não tinha uma visão completa das funcionalidades que deveríamos implementar, tínhamos somente uma listagem simples que criamos a partir do problema que foi apresentado pelo cliente (orientador).

Neste sentido surgiu uma dificuldade que era planejar um projeto que ainda não compreendia por completo e esta dificuldade foi amplificada pelo fato de que nenhum dos integrantes tinha uma grande experiência em planejamento de projetos.

Tentei então lembrar e executar algumas das técnicas que aprendi na matéria de Programação Extrema, nos reunimos então e executamos os passos do "Planning Game", conseguindo obter a partir de uma listagem simples de requisitos um planejamento do sistema Orientado a Objetos que deveríamos desenvolver.

6.1.3 Linguagem PHP

Na última etapa do projeto, que compreendeu o desenvolvimento do módulo para o moodle, tive que enfrentar um desafio de criar uma aplicação web em uma linguagem que eu não conhecia muito bem, assegurando um nível de segurança e estabilidade avançado e levando em conta que como foi a última parte do projeto o tempo que restou era muito limitado.

Para a minha surpresa, o fato de não conhecer a linguagem PHP não foi de forma alguma uma dificuldade enfrentada. Acredito que neste caso a minha experiência com linguagens orientadas a objeto (Java) e também com linguagens estruturadas (C) obtidas ao longo do curso fizeram com que eu adquirisse uma grande facilidade de aprendizado ao encontrar novas linguagens.

Porém a segurança e a estabilidade de uma aplicação PHP não foram imediatas. No caso da estabilidade, acredito que o estudo das classes já prontas no sistema Moodle me ajudaram muito, fornecendo um exemplo de como eu deveria proceder. Já no caso da segurança, felizmente já havia estudado sobre o assunto quando era administrador da Rede Linux do IME. Nesta época, eu era responsável pelo servidor apache da Rede que disponibilizava suporte ao PHP, quando sofremos uma ataque devido à uma aplicação PHP mal escrita de um usuário.

6.1.4 Objetivo Inicial x Tempo

O meu objetivo quando iniciamos o projeto era finalizar uma primeira versão de uma ferramenta que pudesse ser efetivamente utilizada nas matérias de Introdução a Computação, implementando testes de entrada/saída tanto no programa principal quanto separadamente em cada função, executando uma verificação eficiente em relação à segurança do código e muitas outras funcionalidades.

Porém este objetivo entrou em conflito com o tempo de desenvolvimento que deveríamos cumprir, ou seja, percebi que não seria possível implementar todas as funcionalidades que desejava até o final do ano.

Diante deste problema assumi que deveríamos verificar quais eram as funcionalidades que seriam mais importantes e que tornariam o resultado final atingível e utilizável por um professor que desejasse incluir o programa em seu curso. Acredito que este objetivo final, moldado segundo a nossa realidade, foi alcançado.

6.2 Disciplinas relevantes

Dentre as disciplinas mais relevantes para o desenvolvimento do projeto posso destacar

6.2.1 Princípios de Desenvolvimento de Algoritmos

Esta disciplina foi relevante pois me ensinou a programar de forma limpa e organizada e principalmente, introduziu a análise de consumo de tempo e memória.

6.2.2 Laboratório de Programação I e II

As disciplinas de laboratório de programação foram importantes pois permitiram que eu enfrentasse o desenvolvimento sistemas de médio porte, podendo aplicar o conhecimento recém adquirido em MAC122 e MAC110. O amadurecimento em desenvolvimento de software promovido por estas disciplinas fez com que eu me tornasse um programador pronto para enfrentar problemas reais como por exemplo este trabalho de formatura.

6.2.3 Laboratório de Programação Extrema

Destaco esta disciplina como uma das mais importantes não só no desenvolvimento deste projeto de formatura mais também como no desenvolvimento de qualquer tipo de sistema orientado a objetos. Nela aprendi que seguir uma metodologia de desenvolvimento não implica em tornar a programação uma tarefa exaustiva. Utilizando muitas das técnicas ensinadas nessa disciplina pude tornar o sistema do Corretor Automático confiável e estável sem me sobrecarregar com os ditados das metodologias tradicionais.

6.2.4 Conceitos Fundamentais de Linguagens de Programação

Essa disciplina entre outras coisas fez com que a aprendizagem de novas linguagens de programação se tornasse uma tarefa fácil. No caso do projeto de formatura, o estudo dos vários tipos de linguagens promovido nesta matéria fez com que aprender a programar em PHP se tornasse uma tarefa muito rápida.

6.2.5 Sistemas de Bancos de Dados

No projeto de formatura foi necessário implementar a persistência de dados em um banco de dados MySQL, isso foi possível pois na disciplina de Bancos de Dados eu aprendi a desenvolver esse tipo de sistema.

6.3 Interação com o orientador

O orientador do nosso projeto, o professor Carlos Hitoshi Morimoto, exerceu um papel muito importante, pois sempre esteve disposto a orientar quando recorriamos à sua ajuda e também fez o papel de cliente para o projeto que estávamos desenvolvendo.

No início do projeto fizemos reuniões semanais para tentar definir o que precisava ser desenvolvido e para discutir sobre os projetos que já existiam sobre o assunto. Nesta primeira etapa ele nos apresentava artigos sobre projetos de Corretores Automáticos já desenvolvidos e durante a reunião semanal era discutido sobre vantagens e funcionalidades desses projetos.

Mesmo viajando durante o segundo semestre, nunca deixou de responder nossas questões por email e sempre que precisávamos de recursos (Exercícios Programas de anos anteriores) nos disponibilizava rapidamente.

6.4 Aplicação dos conceitos vistos em aplicações reais

Os conceitos que aprendi durante a minha graduação me tornaram um profissional diferenciado, sendo que muitos deles estão sempre presentes no meu dia-a-dia na empresa onde trabalho atualmente.

Sempre que me deparo com sistemas a serem desenvolvidos e problemas novos, inconscientemente estou aplicando todas habilidades adquiridas no curso buscando sempre um código limpo e claro, otimizado, coberto por testes e estável.

Existem também casos onde tenho que lembrar conceitos para aplicá-los, por exemplo, certa vez tive que lembrar sobre a eficiência de algoritmos de busca e ordenação para implementar num projeto da empresa onde trabalho.

6.5 Outros Projetos

Existem também outros projetos que gostaria de destacar que contribuíram muito para a minha formação. O primeiro deles foi o desenvolvimento de um sistema de leitura de telas para deficientes visuais no linux, onde tive a primeira experiência com o desenvolvimento em Linux, orientado pelo professor Alfredo Goldman.

Outro projeto que contribuiu muito para minha formação foi a Administração da Rede Linux do IME sobre orientação do professor Carlos Eduardo Ferreira, neste projeto pude aprimorar meus conhecimentos de Administração de Redes e de Linux. Tive a oportunidade e enfrentar problemas reais em um ambiente de trabalho em grupo. Considero a Rede Linux a minha primeira experiência profissional como desenvolvedor de software.