Sistema de apoio a atividades de laboratório de programação com suporte ao balanceamento de carga e controle de plágio

Allyson Bonetti França[1], José Marques Soares[2]

[1,2]Departamento de Engenharia de Teleinformática (DETI) Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

allysonbonetti@gmail.com, marques@ufc.br

Abstract. *This paper proposes to automate the evaluation of programs in C, Cpp and Java-based process used in the programming marathons. The system has been adapted and integrated through a service-oriented architecture to a virtual learning environment offering support balanceamnto load, since the compilation and execution of programs on a shared remote server may require considerable computational resources, and the control of plagiarism in order to monitor source code similar submissions by students.*

Resumo. *Visando contribuir com as condições de ensino e aprendizagem em laboratórios de disciplinas de programação, este trabalho propõe a automatização de avaliações de programas nas linguagens C, C++ e Java com base no processo adotado em maratonas de programação. O sistema desenvolvido foi adaptado e integrado, através de uma arquitetura orientada a serviços a um ambiente virtual de aprendizagem oferecendo suporte ao balanceamento de carga, uma vez que a compilação e execução de programas em um servidor remoto compartilhado pode requerer muitos recursos computacionais, e ao controle de plágio, a fim de monitorar submissões de códigos fonte semelhantes por parte dos alunos.*

1. Introdução

Em disciplinas voltadas ao aprendizado de técnicas de programação, embora o uso das ferramentas comumente encontradas em ambientes virtuais possa mitigar os problemas de natureza organizacional em práticas laboratoriais, como, disponibilização de notas de aulas, exercícios, manuais de instalação de software, referências, apostilas de programação, entre outros, estas não são suficientes para solucionar a dificuldade de acompanhamento e *feedback* necessários aos alunos.

Nessas práticas, para o professor retirar uma dúvida de um aluno, é necessário, no mínimo, que ele se desloque até o aluno, observe a execução do programa e verifique o seu resultado. Além disso, em caso de erro, muitos alunos assumem posturas passivas e aguardam que o professor o descubra sozinho. Em uma turma numerosa, essa atividade de simples verificação pode tornar o tempo de aula insuficiente.

Uma maneira de reduzir significativamente esse trabalho é permitir que o próprio aluno valide o resultado de seu programa em um procedimento automatizado, semelhante ao que é realizado em maratonas de programação.

O objetivo é fornecer ao professor uma ferramenta que permita o gerenciamento de seus recursos didáticos e que lhe dê apoio ao acompanhamento das práticas laboratoriais com suporte ao controle de plágio e correção automática dos códigos fonte submetidos. Adicionalmente, objetiva-se permitir ao aluno um *feedback* mais rápido, que o incentive a um comportamento mais autônomo e reduza a dependência do professor para os casos em que isto for possível, como é o caso, por exemplo, de falta de correspondência de resultados ou erros de compilação.

O ambiente para apoio a práticas de laboratórios de programação foi concebido como extensão ao sistema BOCA [De Campos 2004]. Desenvolvido na Universidade de São Paulo (USP), o BOCA é usado em maratonas de programação para suporte *online* durante competições, gerenciando times de alunos e juízes, permitindo a proposição de problemas de programação bem como a submissão e avaliação automática de soluções. Para atender a necessidades específicas de aulas de laboratório de programação em turmas regulares e ser integrado a um ambiente virtual de aprendizagem (AVA), como o ambiente Moodle [Moodle 2011], ele precisou ser estendido.

O modelo de integração desenvolvido se apóia no conceito de Arquiteturas Orientadas a Serviços (*Service Oriented Architecture* – SOA), permitindo oferecer funcionalidades aos usuários (alunos e professores) de forma complementar em uma interface única e coesa.

O Moodle foi adotado por ser um ambiente extensível e completo em termos de recursos para gerenciamento de atividades educacionais, apresentando-se como ambiente propício para integrar ferramentas que dêem suporte ao processo de ensino e aprendizagem em disciplinas de programação.

As extensões adicionadas ao BOCA incluem a adaptação de algumas características específicas para o trabalho em laboratórios, bem como a exposição de funcionalidades em forma de serviços usando Web Services (WS). Além disso, utiliza-se uma infraestrutura para prover o balanceamento de carga entre diversos servidores, visto que alguns programas propostos podem apresentar uma carga computacional considerável para um único servidor, levando-se em conta a complexidade da solução, o número de alunos e a quantidade de turmas com trabalhos concorrentes.

Devido ao conjunto de extensões e modificações que o particulariza em relação à ferramenta original, o BOCA, o ambiente de compilação e execução de problemas de programação adaptados para a prática de laboratório que é apresentado neste trabalho é denominado BOCA-LAB [Bonetti 2011a][Bonetti 2011b].

O texto está disposto da seguinte forma: a seção 2 aborda os trabalhos relacionados, apresentando soluções que buscam a automatização na correção e avaliação de códigos fontes em sistemas de cunho educacional; na seção 3 é mostrada a arquitetura de integração. A interação entre os usuários e a arquitetura é explicada na seção 4; na seção 5 é descrito o processo de detecção de plágio. A seção 6 descreve a avaliação do ambiente e, por último, são apresentadas, na seção 7, as conclusões do trabalho.

2. Trabalhos Relacionados

O uso de ambientes virtuais para dar suporte a atividades de programação vem sendo explorado em alguns trabalhos [Kantzavelou 2005][Ng 2005][Wang 2008], porém, a

maioria das plataformas não oferecem um ambiente envolvendo outros recursos educacionais, como ferramentas de discussão síncronas e assíncronas, suporte à gestão de conteúdo, entre outros recursos importantes. Outra limitação de algumas das plataformas é restringir o suporte a apenas um tipo de linguagem de programação.

Em um contexto aproximado ao trabalho apresentado neste artigo, algumas iniciativas foram realizadas no sentido de integrar recursos de apoio a disciplinas de programação ao ambiente Moodle, como o Virtual Programming Lab (VPL) [VPL 2011] e o Onlinejudge [Onlinejudge 2011].

O VPL [VPL 2011] é uma ferramenta de código aberto que permite o desenvolvimento remoto de programas através de um módulo acoplado ao Moodle. A edição do código é feita através de um applet e a compilação e execução do código é realizada em um servidor remoto. É possível efetuar a compilação em várias linguagens de programação e, para a correção e compilação de códigos fonte, este módulo necessita, a cada atividade cadastrada pelo professor, da configuração de como serão os processos de compilação e correção automática.

A arquitetura utilizada pelo VPL centraliza as atividades de compilação e execução do código, não prevendo o balanceamento de carga. Manter um único servidor nessas circunstâncias pode comprometer o desempenho do sistema, visto que é possível prever, para um mesmo ambiente virtual, a existência de várias turmas de programação, e cada uma contendo dezenas de alunos. É necessário, portanto, considerar que os alunos das diversas turmas podem submeter simultaneamente programas para compilação e execução, eventualmente sobrecarregando o servidor.

O Onlinejudge [Onlinejudge 2011], também desenvolvido para gerenciar a submissão de códigos fonte adicionado ao Moodle, pode ser integrado com o uso de WS a uma aplicação denominada IDE ONE [Sphere 2011]. Essa aplicação permite escrever códigos fonte em aproximadamente 40 linguagens de programação diferentes. O Onlinejudge também pode ser executado sem a integração com a IDE ONE, dando suporte, nesse caso, apenas às linguagens C e C++. Entretanto, como se trata de uma aplicação comercial e de código fechado, o modelo de integração permite a submissão de apenas 1000 códigos fonte por mês em uma conta gratuita e não aceita a submissão de vários códigos fonte por vez.

Neste trabalho, em contribuição aos que foram apresentados nesta seção, propõe-se um ambiente de auxílio à compilação e execução remota de programas que, além de ser uma ferramenta livre e de código aberto, seja capaz de reunir as seguintes características:

i. ser integrado a um ambiente virtual de aprendizagem, permitindo o seu uso e o acompanhamento de resultados através da mesma interface de outras ferramentas disponíveis no ambiente virtual;

ii. dar suporte ao uso de diversas linguagens de programação;

iii. permitir a gestão de múltiplos servidores e executar o balanceamento de carga entre os servidores disponíveis;

iv. dar suporte ao controle de plágio entre os códigos enviados pelos alunos.

3. Arquitetura da Integração

A arquitetura de integração desenvolvida é composta por uma estrutura com módulos distribuídos em diferentes servidores, disponibilizando serviços que permitem a operação remota. Os módulos se comunicam através do protocolo SOAP (*Simple Object Access Protocol*), enviando e recebendo mensagens no formato XML (*Extensible Markup Language*). As mensagens são criptografadas para evitar acesso indevido ao código submetido por um aluno com alguma ferramenta do tipo *sniffer*. Os módulos especificados são:

- Módulo de Integração (MI) – responsável pela comunicação entre o Moodle e os módulos da arquitetura.
- Módulo de Informação (MInfo) – responsável por serviços de localização e balanceamento de carga.
- BOCA-LAB – responsável pela compilação e gestão de códigos fonte e problemas .

A Figura 1 ilustra a arquitetura da integração, onde as áreas em cinza destacam o que foi implementado neste trabalho.

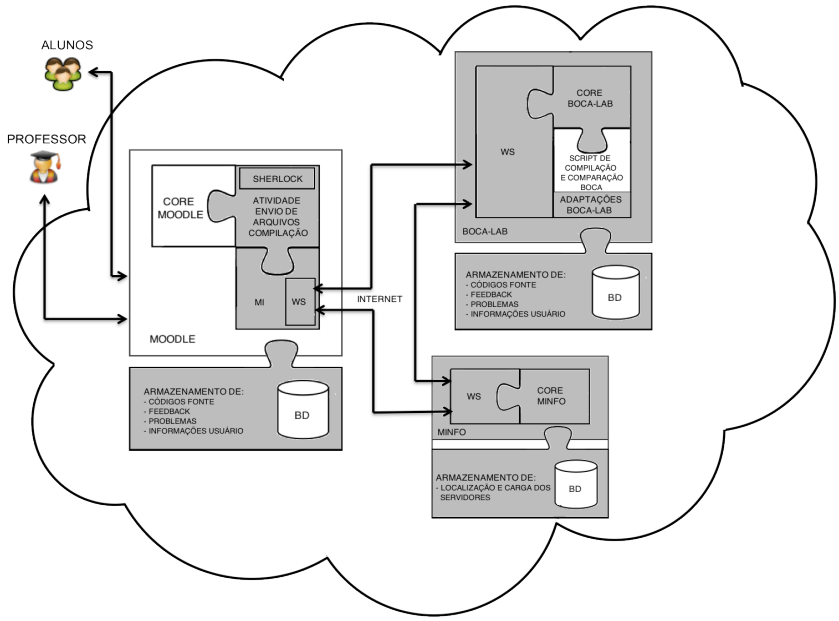

Figura 1 – Arquitetura da integração.

3.1. Módulo de integração (MI)

O MI é responsável pelas transações envolvendo toda a comunicação entre o Moodle e os demais módulos. Ele funciona como uma ponte, intermediando todas as operações iniciadas a partir do Moodle em relação aos diversos módulos da arquitetura.

O MI é responsável pelos seguintes serviços: busca de servidores que se adéquem a um determinado problema proposto pelo professor; registro dos dados necessários aos problemas computacionais pelos tutores/professores pertencentes à plataforma e; envio dos códigos fontes submetidos pelos alunos para os servidores BOCA-LAB.

É também por meio do MI que se tem acesso ao serviço que disponibiliza os *feedbacks* gerados pelo BOCA-LAB, que são exibidos aos alunos na interface do Moodle.

3.2. Módulo de informação (MInfo)

O MInfo é o módulo responsável pela disponibilização de serviços cujos objetivos são a localização e o registro de carga dos servidores BOCA-LAB na arquitetura. Para a localização dos servidores, é levada em consideração a carga de trabalho realizado pelos mesmos e a linguagem de programação requerida pelo problema.

O objetivo do armazenamento e controle da carga do servidor BOCA-LAB pelo MInfo é fornecer um suporte ao balanceamento de carga entre os diversos servidores desse tipo que são comportados pela arquitetura.

3.2.1 Controle e Balanceamento de Carga

A cada requisição feita pelo MI, o módulo MInfo procura dentre os servidores BOCA-LAB aquele que retém menor número de submissões, visando minimizar o tempo de resposta e evitar sobrecarga.

Com o balanceamento de carga, além da otimização no uso dos recursos disponíveis no ambiente, reduzindo o impacto na concorrência para compilação e execução de problemas de programação, é oferecido melhor desempenho para o retorno do *feedback* ao aluno, evitando-se que um processo permaneça tempo desnecessário em filas de servidores sobrecarregados, situação que pode ocorrer, por exemplo, em soluções propostas por ambientes como o VPL [VPL 2011] e o Onlinejudge [Onlinejudge 2011].

A Tabela 1 compara as funcionalidades do ambiente proposto neste trabalho (BOCA-LAB) e das aplicações semelhantes que foram apresentadas nesta seção.

Tabela 1 – Comparativo entre os trabalhos relacionados.

	Compilação em C, C++, Java	Envio de múltiplos arquivos	Fácil adição de novas linguagens	Suporte à integração por serviços	Compilação e correção automáticas	Configuração automática do processo de compilação	Suporte ao balanceamento de carga	Suporte ao controle de plágio
Onlinejudge*	✗	✗	✗	✗	✓	✓	✗	✗
VPL	✓	✓	✓	✗	✓	✗	✗	✓
BOCA	✓	✗	✓	✗	✓	✓	✗	✗
BOCA-LAB	✓	✓	✓	✓	✓	✓	✓	✓

* Levando em consideração a aplicação Onlinejudge sem a integração com o IDE ONE.

3.3. BOCA-LAB

O BOCA-LAB foi concebido por meio de adaptações ao sistema BOCA [De Campos 2004] visando atender a necessidades específicas, inerentes à proposição e ao acompanhamento de práticas de laboratório de programação, tendo suas funcionalidades expostas como serviços (usando WS). Assim, dentro da arquitetura desenvolvida, o BOCA-LAB é o módulo responsável pelos serviços de compilação dos códigos fonte

submetidos, execução e comparação de resultados, cadastro de informações referentes aos problemas e códigos fontes e, por fim, pelo fornecimento dos *feedbacks* gerados pela compilação e execução das soluções submetidas pelos alunos.

3.3.1. Adaptações necessárias ao BOCA

Expandindo-se o escopo para práticas em laboratório de disciplinas de programação, é possível identificar necessidades para as quais o BOCA não se adéqua em sua forma original. Por exemplo, a impossibilidade de enviar mais de um arquivo fonte como solução para o mesmo problema, assim como ocorre no Onlinejudge [Onlinejudge 2011].

Para que se possa submeter um conjunto de classes organizadas em arquivos separados ao BOCA-LAB, modificações foram realizadas para que o servidor identifique um conjunto de programas-fonte compactados, descompacte-os e compile-os separadamente, além de permitir o armazenamento de maneira adequada para posterior execução.

Outra característica original do BOCA é o agrupamento de usuários em times, não existindo informações individuais específicas armazenadas no banco de dados. Para a aplicação visada neste trabalho, sem prejuízo de outras configurações, os problemas devem ser propostos de forma individual, sendo, portanto, necessário armazenar informações dos alunos de maneira a rastrear suas atividades, bem como a permitir o rastreamento de *feedbacks* aos usuários do sistema.

Dessa maneira, adaptações foram implementadas e novas tabelas foram adicionadas ao banco de dados do sistema para armazenar informações individualizadas dos alunos, de forma a permitir o correto direcionamento do *feedback* gerado pela ferramenta. Visando ainda a sua integração com outros ambientes, as funcionalidades do BOCA-LAB foram expostas através de serviços que são destacados na próxima subseção.

3.3.2. Serviços expostos pelo BOCA-LAB

Após o estudo das funcionalidades relativas às operações do sistema BOCA e suas limitações para o contexto explorado neste trabalho, foi elencado um conjunto de operações a serem programadas e expostas por meio de uma camada de serviços de integração pelo BOCA-LAB (Figura 2).

- *Cadastro de problemas computacionais*: permite ao professor cadastrar o enunciado para um problema computacional a ser desenvolvido pelos alunos.
- *Edição da data de expiração de um problema*: permite ao professor alterar a data limite de envio de um problema.
- *Exclusão de um problema*: permite ao professor eliminar um problema previamente cadastrado.
- *Cadastro de códigos fontes*: permite ao aluno enviar seu código fonte resposta para um problema computacional proposto.
- *Importação de feedbacks*: mecanismo que permite a importação dos *feedbacks* gerados pela compilação e execução de um código fonte submetido.

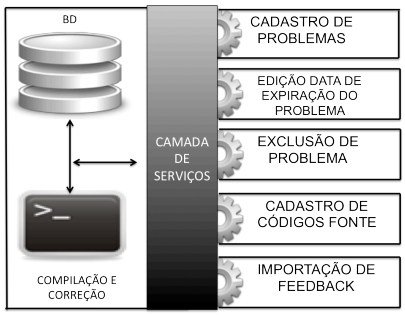

Figura 2 – BOCA-LAB e os serviços implementados e expostos através de uma camada de serviços.

É importante frisar que os Web Services que expõem as funcionalidades do BOCA-LAB podem ser integrados a qualquer cliente ou AVA desenvolvido em qualquer tecnologia, guardando, portanto, independência em relação ao Moodle, embora a arquitetura apresentada neste trabalho tenha se baseado nesse ambiente virtual.

4. Interação com o ambiente integrado ao Moodle

Em seu curso, no ambiente Moodle, o professor deve adicionar a atividade denominada "Envio de arquivos para compilação" que foi implementada e agregada ao ambiente para a administração da submissão de problemas, códigos fonte e recuperação de *feedbacks* por parte dos alunos. A atividade desenvolvida é uma interface entre o usuário (professor/aluno) e o MI. Configuradas as informações dessa atividade, o professor deve cadastrar seu problema.

Ao submeter o problema, o MI envia os dados para o MInfo que busca e retorna o endereço do BOCA-LAB que melhor se adéqua aos requisitos da atividade. Uma vez enviado o problema, o aluno pode submeter seu código fonte que é repassado ao BOCA-LAB através do MI.

A cada código fonte recebido ou processado, o servidor BOCA-LAB atualiza a informação sobre o seu estado nos servidores MInfo, permitindo, assim, uma melhor distribuição de carga pelo mesmo. Após o envio do código fonte, a interface do aluno fica bloqueada para novas submissões ao mesmo problema até ser disponibilizado o *feedback*. O *feedback* retornado ao aluno pelo BOCA-LAB é composto por uma resposta do compilador, um arquivo contendo os erros da compilação, caso ocorram, e um outro contendo a saída gerada pelo seu programa.

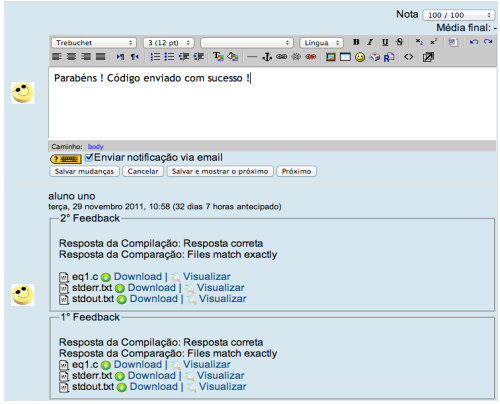

Figura 3 – Interface de atribuição de notas.

Os resultados de todas as submissões são armazenados pelo sistema e apresentados na interface do professor, como ilustrado na Figura 3, permitindo ao mesmo analisar o desempenho do aluno, facilitando a atribuição de nota.

A nota atribuída às atividades de programação é registrada juntamente com as demais notas de atividades regulares de um curso Moodle, como Fóruns, Chats e outras atividades que compõem o conjunto de elementos de avaliação de um aluno no ambiente virtual. Isto permite que as notas de programação possam ser usadas em fórmulas do Moodle para composição da média final do aluno.

5. Detecção de Plágio

Verificar o plágio em trabalhos de programação requer do professor a comparação de códigos par a par, tarefa ainda mais difícil quando a solução dos alunos for desenvolvida com o uso de mais de um código fonte.

Para dar suporte a esta atividade, o plágio pode ser detectado através de softwares especializados. Dentre os softwares detectores de plágio de distribuição livre apresentados no relatório desenvolvido por Scaife [Scaife 2007], o Sherlock, criado por Pike [Pike 2011], apresenta características que facilitam a sua integração a um AVA [Santos 2010], motivo pelo qual foi adotado na arquitetura proposta neste trabalho.

O Sherlock encontra semelhanças entre documentos textuais usando assinaturas digitais e apresenta os resultados das análises em tempo real, podendo ser utilizado em dois modos de operação. No primeiro ele pode descobrir plágio em tarefas de linguagem natural e, no outro, ele pode descobrir plágio em tarefas de código fonte.

5.1. Integração do Sherlock à Arquitetura

Integrado ao ambiente apresentado, o Sherlock oferece ao professor uma ferramenta de suporte para analisar e detectar plágio nos códigos enviados pelo alunos.

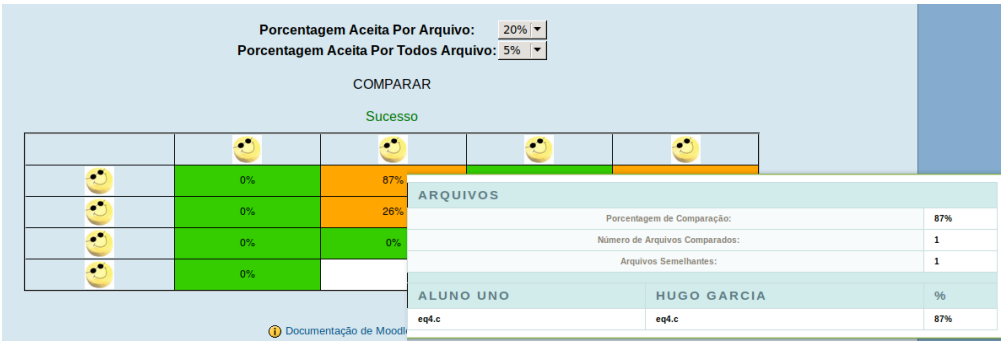

Figura 4 – Interface contendo as informações importantes para comparação de códigos.

Após ajustar os parâmetros do Sherlock para indicar o percentual de semelhança aceitável, o processo de verificação de semelhança é acionado pelo professor com um simples clique. Os resultados da comparação são visualizados pelo professor em uma tabela em que nas células é apresentado o percentual de semelhança para cada dupla de alunos da turma. Outras informações, como identificação dos alunos, número de arquivos comparados, número de arquivos semelhantes e a porcentagem de semelhança arquivo a arquivo podem ser apresentadas posicionando-se o ponteiro do mouse sobre uma célula, como ilustrado na Figura 4.

Ao clicar sobre uma célula com o resultado para uma dupla de alunos, é possível visualizar os códigos escritos pelos mesmos lado a lado, permitindo compará-los visualmente. Nessa tela é possível, ainda, atribuir notas e registrar comentários destinados ao aluno.

6. Validação e Testes do Ambiente

Duas experimentações foram realizadas no ano de 2011, ambas em cursos da Universidade Federal do Ceará. A primeira foi conduzida em duas turmas da disciplina de Técnicas de Programação para Engenharia I, do Departamento de Engenharia de Teleinformática, com a proposição de um único problema aos alunos. A segunda foi conduzida em uma turma da disciplina de Fundamentos de Programação, do Departamento de Computação, em que foram propostos 12 problemas aos alunos. Destes 12 problemas, 4 são problemas de nível básico, que foram resolvidos em laboratório com ajuda do tutor. O objetivo foi fazer com que os alunos se familiarizassem com o ambiente que foi utilizado. Os demais foram resolvidos pelos alunos e submetidos em horários livres, tendo sido estabelecido um prazo final de 15 dias.

No intuito de avaliar a relevância da proposta, uma enquete foi dirigida aos estudantes por intermédio de um questionário [Bonetti 2011b], objetivando-se registrar as impressões dos mesmos sobre o uso da ferramenta nesse tipo de disciplina e avaliar a robustez da aplicação. Os testes realizados permitiram avaliar os principais requisitos levantados, bem como reconduzir alguns problemas, que foram corrigidos durante a sua execução.

7. Conclusão

O modelo de integração utilizado visa atribuir transparência no acesso aos recursos do BOCA-LAB. Neste trabalho, o acesso aos recursos é realizado integralmente a partir da interface do Moodle, mas, devido ao fato de serem expostos como serviços, podem ser adaptados a outros AVAs bastando, para isso, construir um módulo de integração (MI) específico à plataforma para consumir os serviços exportados pelo mesmo.

Os testes realizados com as turma de graduação, utilizando problemas reais, assim como o *feedback* fornecido pelos alunos em resposta à enquete realizada, permitiram avaliar o funcionamento da integração, bem como corrigir erros de implementação e realizar alguns ajustes nos serviços especificados.

Durante a enquete realizada, alguns alunos registraram dificuldades de compreensão em relação à interface para a submissão de códigos fonte, o que nos mostra a necessidade de torná-la mais intuitiva. O mesmo ocorreu com os arquivos retornados como *feedback* pelo BOCA. Usuários que não possuem suficiente conhecimento da linguagem de programação ou mesmo do funcionamento do compilador, não compreendem facilmente as informações apresentadas, sendo necessário reavaliar a forma e o conteúdo da apresentação dos erros aos alunos em função de seu nível de conhecimento.

Tendo em vista a possibilidade de plágio na submissão de soluções aos problemas propostos, o Sherlock mostrou-se eficaz na detecção destes, auxiliando o professor no diagnóstico e no controle deste tipo de atitude entre os alunos da turma.

O balanceamento de carga se mostrou eficaz no testes realizados. Na versão atual, a distribuição dos programas se baseia na quantidade de códigos fonte ainda não processados e que são armazenados nos servidores. Entretanto, o modelo foi projetado de maneira que pode ser adaptado facilmente para técnicas de balanceamento que levem em consideração outros parâmetros, como a complexidade prevista para os códigos enviados ou as características físicas dos servidores, como quantidade de memória livre, uso de CPU entre outros fatores.

Referências

Bonetti, A. et al. (2011) "Um sistema orientado a serviços para suporte a atividades de laboratório em disciplinas de técnicas de programação com integração ao ambiente Moodle." RENOTE. Revista Novas Tecnologias na Educação, v. 9, p. 1-11.

Bonetti, A.; Soares, J. M. (2011) "Sistema de apoio a atividades de laboratório de programação via Moodle com suporte ao balanceamento de carga." In: XXII Simpósio Brasileiro de Informática na Educação. Anais do XXII SBIE - XVII WIE , v. 22, Aracaju.

De Campos, C. P. ; Ferreira, C. E. (2004). "BOCA: Um Sistema de Apoio para Competições de Programação." Workshop de Educação em Computação, 2004, Salvador. Anais do Congresso da SBC, 2004.

Kantzavelou, I. (2005) "A virtual lab model for an introductory computer science course". Facta Universitatis, vol. 18, no. 2, pp.263 -274.

Moodle – "A Free, Open Source Course Management System for Online Learning." (2011) Disponível em http://moodle.org/. Acesso em 17 de Março de 2011.

Ng, S. C. et al. (2005) "A Web-Based Environment to Improve Teaching and Learning of Computer Programming in Distance Education." In: Lau, R., Li, Q., Cheung, R., Liu, W. (eds.) ICWL 2005. LNCS, vol. 3583, p. 279–290. Springer, Heidelberg.

Onlinejudge. (2011) Disponível em: https://github.com/hit-moodle/onlinejudge. Acessado em 21 de Março de 2011.

Pike, R. (2011) "The Sherlock Plagiarism Detector." Disponível em: http://sydney.edu.au/engineering/it/~scilect/sherlock/ Acesso em: 04 Dezembro 2011

Santos, F. A. O. (2010) "Criação da Ferramenta de Detecção de Plágio em Ambiente Virtual de Aprendizagem." Dissertação (Mestrado) – Departamento de Engenharia Elétrica, Universidade Federal de Itajubá, Itajubá/Mg.

Scaife, B. (2007) "Evaluation of Plagiarism Software: Plagiarism Detection Software Report for JISC Plagiarism Advisory Service" Manchester, ver. 1.5, n. 11147, set.

Sphere Research Labs – IDE ONE (2011) Disponível em http://ideone.com/. Acesso em 22 de Março de 2011.

VPL – "Virtual Programming Lab Disponíel" em: http://vpl.dis.ulpgc.es/. Acesso em: 21 Março 2011. Acesso em 21 de Marco de 2011.

Wang, J.; CHEN, L.; ZHOU, W. (2008) "Design and Implementation of an Internet-Based Platform for C Language Learning." In ICWL,187-195.