Sistema de apoio a atividades de laboratório de programação via Moodle com suporte ao balanceamento de carga

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. *To contribute to the conditions of teaching and learning computer programming, this paper presents an environment that integrates Moodle with a web-based tool used to support programming competitions. The tool allows the registration of the result of the compilation and implementation of the proposed problems in C, Cpp and Java, tracking the student's submissions, and allows the evaluation of the teacher by the Moodle interface. Whereas the compilation and execution on a shared remote server may require many computational resources, the environment developed to support load balancing.*

Resumo. *Visando contribuir com as condições de ensino e aprendizagem em laboratórios de disciplinas de programação, este trabalho apresenta um ambiente que integra o Moodle a uma ferramenta Web utilizada no apoio a competições de programação. A ferramenta permite o registro do resultado da compilação e da execução de problemas propostos nas linguagens C, C++ e Java, rastreando as submissões dos alunos, e possibilita a avaliação do professor via interface do Moodle. Considerando que a compilação e execução de programas em um servidor remoto compartilhado pode requerer muitos recursos computacionais, o ambiente desenvolvido oferece suporte ao balanceamento de carga.*

1. Introdução

Disciplinas de técnicas de programação em cursos de computação e engenharia são, em geral, muito numerosas, exigindo bastante do professor e dos monitores que, muitas vezes, não conseguem realizar um acompanhamento individual dos alunos de maneira eficiente. Isto pode provocar desestímulo, impelindo a turma, por vezes, à dispersão em aulas de laboratório, situação dificilmente controlável pelo professor.

Para organizar o trabalho, professores fazem uso de ferramentas de apoio. Aplicações Web, por exemplo, podem ser usadas para disponibilização de notas de aula, proposição e submissão de trabalhos e registro de notas. Embora o uso de tais ferramentas possa mitigar os problemas de natureza organizacional em práticas laboratoriais, não são suficientes para solucionar a dificuldade de acompanhamento e *feedback*. Como forma de ilustrar situações comuns em laboratórios de programação,

uma questão frequentemente colocada por alunos é: *Professor, o meu programa está correto?*

Embora essa seja uma pergunta de resposta simples (sim ou não), para respondê-la, é necessário que o professor se desloque até o aluno, observe a execução do programa e verifique o seu resultado. Em caso de erro, muitos alunos assumem posturas passivas e aguardam que o professor descubra o erro. Em uma turma de 60 alunos, por exemplo, 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 semelhante ao realizado em olimpíadas de programação.

Visando contribuir com as condições de ensino e aprendizagem de cursos de programação, é apresentado neste trabalho um ambiente que permite a automatização de avaliações de programas propostos pelo professor para desenvolvimento nas linguagens de programação C, C++ e Java. O objetivo é, por um lado, fornecer ao professor uma ferramenta que permita o gerenciamento de seus recursos didáticos e que lhe dê apoio ao acompanhamento das práticas laboratoriais. Por outro lado, objetiva-se permitir ao aluno um *feedback* mais rápido, que o incentive a um comportamento mais autônomo.

Adicionalmente, é definido um modelo de integração desse ambiente, que é voltado especificamente para a avaliação de programas, aos chamados ambientes virtuais de aprendizagem (AVA). A integração permite oferecer as funcionalidades disponíveis em cada ferramenta aos usuários (alunos e professores) de forma complementar e através de uma interface única e coesa. Para a composição e avaliação do modelo de integração, adotou-se uma metodologia que se apóia no conceito de arquitetura orientada a serviços (Service Oriented Architecture – SOA) [Papazoglou 2007].

O ambiente desenvolvido para apoio a laboratórios de programação foi concebido como extensão do sistema BOCA [De Campos 2004]. Desenvolvido na Universidade de São Paulo (USP), este sistema é usado em maratonas de programação para submissão e avaliação automática de problemas e precisou ser adaptado para atender a necessidades específicas do ambiente. As extensões incluem a adaptação de algumas funcionalidades específicas para o trabalho em laboratórios, bem como a exposição de funcionalidades em forma de serviços. Além disso, foi incluída 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. O sistema estendido é denominado neste trabalho BOCA-LAB.

O BOCA-LAB foi integrado ao Ambiente Virtual de Aprendizagem (AVA) Moodle [Moodle 2011]. O Moodle forneceu a interface e o conjunto de funcionalidades necessárias à gestão e ao acompanhamento das atividades associadas ao laboratório de programação. A integração desses dois ambientes foi realizada com o uso de Web Services (WS), que se destacam como tecnologia para a implementação de SOA e vêm sendo utilizados em sistemas educacionais como o Sakai [Sakai 2011].

O texto está disposto da seguinte forma: a seção 2 aborda os trabalhos relacionados, apresentando soluções que buscam essa automatização na correção e

avaliação de códigos fontes em sistemas de cunho educacional; a seção 3 mostra as características do Moodle e do BOCA, ferramentas que compõem o ambiente de integração; na seção 4 é mostrada a arquitetura de integração. A interação entre os usuaários e a arquitetura é explicada na seção 5. A seção 6 descreve a avaliação do ambiente e, por último, são apresentadas, na seção 7, as conclusões e perspectivas do trabalho.

2. Trabalhos Relacionados

O uso de ambientes virtuais para dar suporte a atividades de programação vem sendo explorado há alguns anos.

Kantzavelou [Kantzavelou 2005] apresenta um modelo de laboratório virtual para um curso introdutório à Ciência da Computação. A arquitetura do modelo consiste em sete módulos onde cada módulo corresponde a um tópico específico da disciplina, como lógica e linguagem de programação, algoritmos, arquitetura de computadores entre outros. Cada módulo possui associado recursos como sites especializados, simuladores ou laboratórios virtuais que oferecem exercícios e problemas específicos relacionados ao tópico. Entretanto esse modelo ainda está em fase de implementação não apresentando todos os módulos da disciplina concluídos.

Ng [Ng 2005] investiga como a nova tecnologia pode auxiliar no processo de ensino e aprendizagem em disciplinas de programação, propondo um ambiente Web interativo para o ensino de linguagens de programação Java. O ambiente apresenta funcionalidades que permitem a compilação e o retorno de erros dos programas submetidos.

Wang [Wang 2008], propõe um sistema Web para o ensino da linguagem de programação C. Esse sistema é desenvolvido em .NET e oferece funcionalidades que permitem a compilação e checagem de erros dos programas submetidos.

Embora esses trabalhos tragam estudos de plataformas interativas para o ensino de linguagem de programação, eles não oferecem um ambiente envolvendo outros recursos educacionais como ferramentas de discussão síncronas e assíncronas, suporte a gestão de conteúdo entre outros recursos importantes, principalmente para disciplinas ministradas a distância. Outro limite de algumas plataformas citadas é restringir o suporte a apenas um tipo de linguagem de programação.

Em um contexto mais aproximado ao trabalho aqui apresentado, algumas iniciativas foram realizadas no sentido de integrar recursos de apoio a disciplinas de programação ao ambiente Moodle, como o VPL [VPL 2011] e o Onlinejudge [Onlinejudge 2011]. O VPL (Virtual Programming Lab) 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 com segurança em um servidor Linux remoto. É possível efetuar a compilação em várias linguagens de programação, dentre elas C, C++, PHP, Java e Fortran. 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 de códigos fonte e de correção automática. A arquitetura utilizada pelo VPL não permite a adição de novas ferramentas ou o balanceamento de carga, visto que o servidor responsável pela compilação e execução do código

submetido é único. Um único servidor para tal tarefa pode se tornar um gargalo uma vez que podemos ter em um mesmo ambiente Moodle várias turmas contendo dezenas de alunos submetendo simultaneamente.

O Onlinejudge, também desenvolvido para gerenciar a submissão de códigos fontes adicionado ao Moodle, pode ser integrado com o uso de WS a uma aplicação denominada Ideone [Sphere 2011]. Essa aplicação permite escrever códigos fonte em aproximadamente 40 linguagens de programação diferentes, sendo os mesmos executados diretamente a partir do navegador. O Onlinejudge também pode ser executado sem a integração com o Ideone, 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.

Embora todos os trabalhos citados contenham importantes contribuições para o apoio de práticas laboratoriais em turmas de programação, neste trabalho propõe-se um ambiente de auxílio a compilação e execução remota de programas que 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.

3. Os Ambientes Moodle e BOCA

O Moodle (Modular Object Oriented Distance Learning) é um sistema de código aberto baseado na Pedagogia Social Construcionista [Alves 2005]. Rico em recursos educacionais, oferece alta flexibilidade para configuração e uso. Além disso, seu desenvolvimento modular permite a fácil inclusão de novos recursos que podem melhor adaptá-lo às necessidades da instituição que o utiliza. Por ser um ambiente extensível e completo em termos de recursos para gerenciamento de atividades educacionais, o Moodle apresenta-se como ambiente propício para integrar ferramentas que dêem suporte ao processo de ensino e aprendizagem em disciplinas de programação.

O BOCA é um sistema de apoio a competições de programação desenvolvido para uso em maratonas promovidas pela Sociedade Brasileira de Computação. Oferece suporte online durante a competição, gerenciando times de alunos e juízes, permitindo a proposição de problemas de programação bem como a submissão e avaliação automática de soluções. Sendo um sistema de código aberto, o BOCA pode ser adaptado ao contexto de laboratórios de programação e integrado a um AVA, como o ambiente Moodle. As características de principal interesse para a integração do BOCA ao Moodle são apresentadas nas próximas subseções.

3.1. Processo de Compilação e Correção dos Códigos Fonte Enviados ao BOCA

Para cada problema cadastrado no BOCA, são necessários um arquivo contendo um conjunto de entradas e outro contendo as respectivas saídas. Os arquivos de entrada e saída são obtidos pelo professor, através de um programa executável elaborado pelo mesmo como solução ao problema, onde as entradas enviadas para o programa e as saídas geradas são armazenadas em arquivos distintos.

Ao receber o código fonte submetido pelo time, o sistema o compila. Caso não ocorra nenhum erro, é gerado um executável e realizada a sua execução. O teste ao programa é realizado enviando as entradas contidas no arquivo de entrada cadastrado para o problema. Em seguida, o sistema efetua a comparação da saída gerada com o arquivo de saída cadastrado para o problema. Ao final das etapas de compilação e comparação, é enviado um *feedback* para o time, contendo eventuais erros encontrados no processo de compilação ou na comparação da saída.

3.2. Necessidades Inerentes à Integração

Para dar suporte à integração das funcionalidades das duas ferramentas, o sistema de armazenamento de dados, a submissão de arquivos e a compilação realizada pelo BOCA precisam ser adaptados. Em sua concepção original, o sistema BOCA só permite o envio de um único arquivo por problema computacional proposto.

O envio de mais de um programa fonte pode ser facilmente resolvido através da compactação do conjunto de arquivos usando ferramentas como arj ou zip. Entretanto, essa operação resolve apenas parcialmente o problema, tendo em vista que é necessário o servidor identificar o arquivo compactado, executar a descompactação, a compilação dos programas fontes e o armazenamento de maneira adequada dos mesmos.

Para a aplicação visada neste trabalho, os problemas devem ser propostos de forma individual, sendo necessário, portanto, adaptar o BOCA para armazenar informações de forma a identificar o aluno no Moode, rastrear as atividades do mesmo e fornecer *feedbacks*.

Para a gestão do cadastro de alunos, registro de atividades e notas, entre outros aspectos administrativos das atividades educacionais, o ambiente Moodle oferece os recursos necessários. Assim, verifica-se a complementaridade entre os ambientes a serem integrados neste trabalho, valorizando o conjunto de competências peculiares a cada um. Além das alterações propostas para o BOCA, um módulo de extensão deve ser criado no Moodle de maneira a permitir a integração entre os ambientes. Este módulo de extensão deve: permitir o acesso à funcionalidades disponibilizadas pelo BOCA; usar estruturas específicas para registro dos dados relativos aos problemas propostos; apresentar interfaces para submissão de soluções ao BOCA e para apresentação dos resultados, ambos a partir da interface do Moodle.

Na seção seguinte, é discutida a arquitetura de integração proposta e os módulos que a integram.

4. Arquitetura de Integração Baseada em Web Services com suporte ao Balanceamento de Carga

A arquitetura da integração é composta por três módulos que se comunicam através do protocolo SOAP usando mensagens criptografadas no formato XML. A Figura 1 ilustra a estrutura de comunicação destes módulos, ressaltando a coexistência de múltiplos servidores que dão suporte ao balanceamento de carga. Os módulos são detalhados nas próximas subseções.

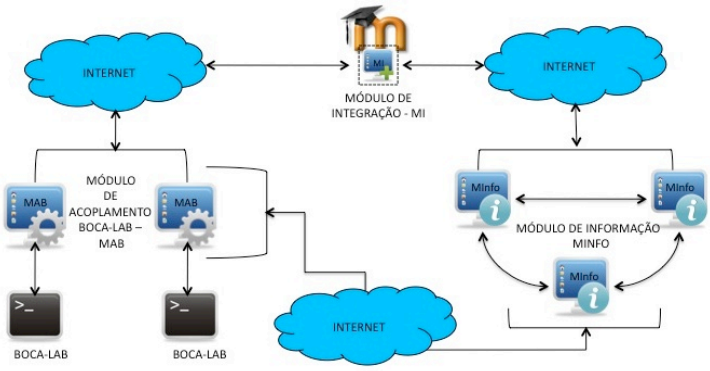

Figura 1. Arquitetura da integração.

4.1. Módulo de Integração (MI)

O MI é responsável pelo acesso ao serviço de busca de servidores; registro dos dados necessários aos problemas computacionais e; o envio e recuperação de *feedback* dos códigos fonte submetidos para os servidores MAB.

4.2. Módulo de Informação (MInfo)

O MInfo é o módulo responsável pela disponibilização dos serviços de localização e registro do estado dos servidores MAB. O armazenamento de informações sobre o estado dos servidores MAB o permite efetuar o balanceamento de carga.

4.2.1 Controle e Balanceamento de Carga

O balancemento de carga na arquitetura é realizado pelo MInfo, a cada requisição feita pelo MI, o módulo verifica procura detre os servidores MAB aquele que retém menor número de submissões, visando minimizar o tempo de resposta e evitar sobrecarga.

Além de reduzir o impacto da concorrência por recursos computacionais no mesmo servidor para compilação e execução de problemas de programação, o balanceamento de carga agiliza o processo de *feedback* para o aluno, evitando que um processo permaneça tempo desnecessário em filas em servidores sobrecarregados.

4.3. Módulo de Acoplamento BOCA-LAB (MAB)

O MAB é responsável pela disponibilização de serviços que permitem o recebimento e repasse ao BOCA-LAB dos dados dos problemas e códigos fonte; e a recuperação de *feedback*, além do controle secundário da carga de compilação no BOCA-LAB, evitando o recebimento de requisições caso o servidor esteja com sua carga máxima.

5. Interação com a Integração

Em seu curso, no ambiente Moodle, o professor deve adicionar a atividade denominada “Envio de arquivos para compilação” que foi implementada e agregada ao Moodle para a administração da submissão de problemas, códigos fonte e recuperação de *feedbacks* por parte dos alunos. A atividade implementada é uma interface entre o usuário (professor/aluno) e o MI. Configuradas as informações dessa atividade, o professor deve cadastrar seu problema através de um formulário (Figura 2).

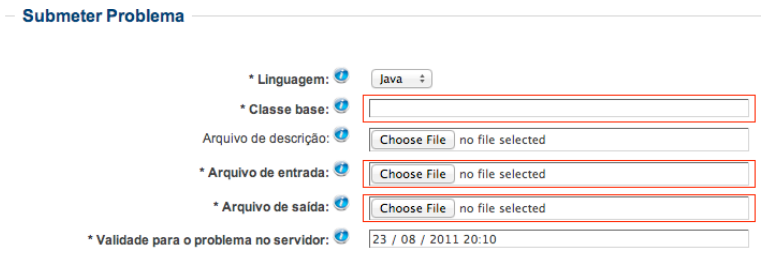

Figura 2. Formulário de cadastro de um problema.

Ao submeter o formulário, o MI envia os dados para o MInfo que busca e retorna o endereço do MAB que melhor se adéqüa aos requisitos do problema. Enviado o problema ao MAB, um formulário (Figura 3) é então disponibilizado na interface do aluno para a submissão de seu código fonte. Uma vez submetido, o código fonte é enviado ao MAB através do MI.

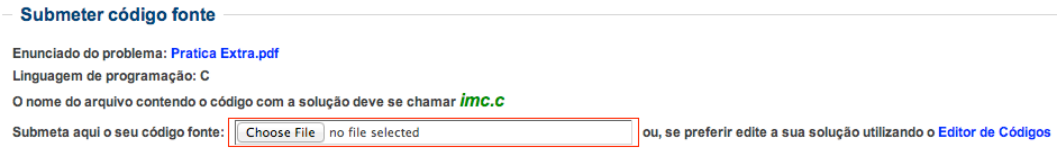

Figura 3. Formulário de envio de código fonte.

A cada código fonte recebido ou processado, o servidor MAB atualiza a informação sobre o seu estado nos servidores MInfo, permitindo, assim, uma melhor distribuição de carga pelo mesmo. Essa atualização, só é feita caso a carga máxima do servidor esteja próxima de atingir o limite configurado.

Após o envio do código fonte, a interface do aluno fica bloqueada para novas submissões ao mesmo problema até ser disponibilizado o seu *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 programa.

Os tipos de resposta retornados pelo compilador são mostrados pela Tabela 1.

Tabela 1. Tipo de *feedbacks* retornados pelo BOCA-LAB.

Reposta	Descrição
YES	Programa foi aceito sem erros.
NO: Incorrect Output	Saída dos testes incorreta.
NO: Time-limit Exceeded	O programa excedeu o tempo estipulado.
NO: Runtime Error	Durante o teste ocorreu um erro de execução.
NO: Compilation Error	Programa possui erros de sintaxe.

Os resultados de todas as submissões são armazenados pelo sistema e apresentados na interface do professor, como apresentado na Figura 4, permitindo ao mesmo analisar o desempenho do aluno, facilitando assim, a atribuição da nota.

A nota atribuída às atividades de programação figuram junto ao conjunto de notas de atividades regulares de um curso Moodle, como Fóruns, Chats e outras atividades, compondo assim a nota final do aluno.

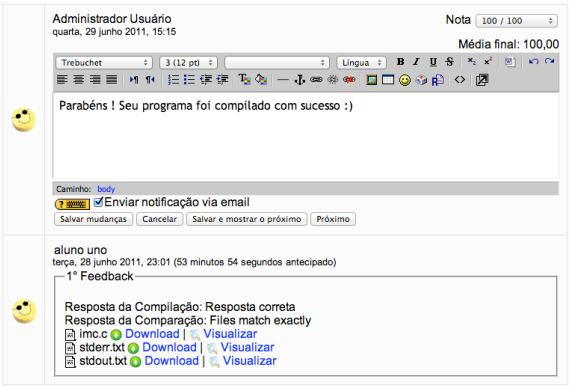

Figura 4. Interface de atribuição de notas.

6. Validação e Testes do Ambiente

Para validação da ferramenta e aprimoramento da mesma, uma experimentação foi realizada em duas turmas de Introdução à Programação, onde um determinado problema foi proposto para ser resolvido pelos alunos utilizando a ferramenta apresentada.

No intuito de avaliar a relevância da proposta, uma pesquisa foi dirigida aos estudantes, objetivando-se obter *feedback* dos mesmos sobre o uso da ferramenta nesse tipo de disciplina e avaliar a robustez da arquitetura de integração e balanceamento de carga. O modelo, os resultados e a análise da pesquisa serão discutidos nessa seção a fim de se observar a importância de se agregar a ferramenta nesse tipo de disciplina.

Para a pesquisa, um questionário web foi desenvolvido e disponibilizado através do Moodle, facilitando o acesso pelo aluno. A pesquisa foi anônima, voluntária e restrita somente aos alunos dessas duas turmas. O questionário foi composto de cinco questões envolvendo aspectos relevantes a percepção do aluno sobre a ferramenta. Com base no grau de concordância e aceitação sobre as questões, o aluno indicava o seu conceito de acordo com os ícones da Tabela 2:

Tabela 2. Ícones utilizados no questionário.

Ícone	Descrição	Ícone	Descrição
	Muito Bom		Fraco
	Bom		Não Satisfatório
	Regular		Não se aplica

As questões aplicadas foram:

Q-01. O sistema apresenta clareza em sua tela de submissão ? Sua interface está compreensível ?

Q-02. Se você criou ou editou seu código fonte utilizando o editor de códigos da ferramenta, a interface deste é apresentada de forma clara ?

Q-03. O *feedback* apresentado com os arquivos contendo a comparação com a saída padrão (sdterr.txt) e a saída do seu programa (stdout.txt) ajudam na resolução dos possíveis erros encontrados em seu código fonte ?

Q-04. A ferramenta apresentada agrega funcionalidades importantes ao Moodle e a sua disciplina ?

Q-05. Em sua opinião, seria possível utilizar a ferramenta como suporte para acompanhar um curso ou treinamento inteiramente a distância ?

A Tabela 3 resume a freqüência de cada ícone em cada questão e a freqüência dos alunos que responderam com os ícones azul ou verde, dando as maiores notas por questão.

Tabela 3. Resultado do questionário por questão.

Questões							
Q-01	27.27%	36.36%	27.27%	9.10%	0%	0%	63.63%
Q-02	9.10%	36.36%	0%	0%	0%	54.54%	45.46%
Q-03	9.10%	18.18%	18.18%	18.18%	36.36%	0%	27.28%
Q-04	54.54%	36.36%	0%	9.10%	0%	0%	90.9%
Q-05	36.36%	45.44%	0%	9.10%	9.10%	0%	81.8%

Com base nos resultados da Tabela 3 podemos observar que a maioria dos alunos (90.9%) acham a ferramenta muito importante em sua disciplina, onde 81.8% responderam que seria possível utilizar a ferramenta como suporte para acompanhar um curso ou treinamento inteiramente a distância, onde se tem pouca ou nenhuma ajuda por parte do professor.

Dentre as questões que obtiveram uma aceitação abaixo da média estão as questões 2 e 3 com respectivamente 45.46 e 27.28%. Podemos perceber com o resultado dessas questões que a interface de *feedback* retornado ao aluno precisa ser modificada melhorando o entendimento do mesmo sobre os possíveis erros encontrados em seus programas e que a maioria dos usuários utilizaram a plataforma de desenvolvimento instalada em seu computador para criar ou editar seu programa, porém, acreditamos que a funcionalidade do editor de códigos integrado a ferramenta é muito importante, deixando o usuário independente de plataforma.

7. Conclusões e Perspectivas

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.

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 dos códigos enviados ou as características físicas dos servidores, como quantidade de memória livre, uso de CPU entre outros fatores.

A experimentação realizada com as turmas de graduação permitiu validar o funcionamento da integração, bem como ajustar a implementação dos serviços especificados. Novos problemas estão sendo adaptados e uma nova experimentação está sendo planejada. Atualmente está sendo desenvolvido um novo módulo com a função de comparar os códigos enviados pelos alunos a fim de minimizar o problema da cópia de códigos fonte.

Referências

Alves, L. and Brito, M. (2005) "O Ambiente Moodle como Apoio ao Ensino Presencial." Disponível em: www.abed.org.br/congresso2005/por/pdf/085tcc3.pdf. Acesso em: 06 janeiro 2011.

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

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., Choy, S.O., Kwan, R., Chan, S.F.: 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, pp. 279–290. Springer, Heidelberg (2005).

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

Papazoglou, Mike P.; Heuvel, Willem-Jan van den. ."(2007) "Service Oriented Architectures: approaches, technologies and research issuesThe VLDB Journal, v. 16, p. 389-415, 2007.

Sakai: Collaborative and Learning Environment for Education. Disponível em https://confluence.sakaiproject.org/display/WEBSVCS/Home. Acesso 20 de Janeiro de 2011.

Silva, C. R. O.; Soares, J. M.; Serra, A. de B. (2008) "EPT Virtual: espaço digital de apoio à pesquisa e aplicação das TICs na educação profissional e tecnológica." Revista Brasileira da Educação Profissional Tecnológica, Brasília, v. 1, n. 1, p. 118-130, jun. 2008.

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

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

Wang, J., Chen, L., Zhou, W.: Design and Implementation of an Internet-Based Platform for C Language Learning. In ICWL(2008) 187-195.