Um sistema orientado a serviços para suporte a atividades de laboratório em disciplinas de técnicas de programação com integração ao ambiente Moodle

Allyson Bonetti França - DETI/CT - Universidade Federal do Ceará,
allysonbonetti@gmail.com
José Marques Soares - DETI/CT - Universidade Federal do Ceará,
marques@ufc.br
Danielo Gonçalves Gomes - DETI/CT - Universidade Federal do Ceará,
danielo@ufc.br
Giovanni Cordeiro Barroso - Departamento de Física - Universidade Federal do Ceará,
gcb@fisica.ufc.br

Resumo. As disciplinas de técnicas de programação ministradas em cursos de computação e engenharias costumam ser muito numerosas, exigindo tempo e empenho de professores e monitores que, muitas vezes, não conseguem realizar um acompanhamento eficiente dos alunos. Visando contribuir com as condições de ensino e aprendizagem desses cursos, tanto na modalidade presencial como a distância, este trabalho apresenta o desenvolvimento da automatização de avaliações de programas nas linguagens C, C++ e Java. O ambiente integra uma versão estendida de uma ferramenta Web utilizada em maratonas de programação, por meio de uma arquitetura orientada a serviços, ao Ambiente Virtual de Aprendizagem Moodle.

Palavras-chave: serviços web, avaliações automatizadas, laboratório de programação e arquitetura orientada a serviços.

Abstract. The disciplines of programming techniques taught in courses in computing and engineering tend to be very numerous, requiring time and commitment of teachers and monitors that often can not perform an effective monitoring of students. To contribute to the conditions of teaching and learning of these courses, both in the classroom as the distance mode, this paper presents the development of the automation of program evaluations in C, C + + and Java. The environment integrates an extended version of a tool used in Web programming marathons, through a service-oriented architecture, and the virtual learning environment Moodle.

Keywords: web services, automated evaluations, laboratory programming and service-oriented architecture.

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 que é 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 utilizam SOA para integração entre 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 usuários 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

Alguns trabalhos propõem a construção de ambientes virtuais de apoio a atividades educacionais sob o paradigma da orientação a serviços e visam disponibilizar funcionalidades através de interfaces consistentes, coesas e fracamente acopladas, permitindo maior interoperabilidade e flexibilidade de manutenção (Moura 2005). Outros buscam promover o reaproveitamento de sistemas legados para integração com outros sistemas e composição de novas aplicações (Dan 2008).

Oliveira (Oliveira 2009) propõe a integração de ferramentas educacionais que compõem o projeto EPT Virtual e apresenta um ambiente orientado a serviços para expor funcionalidades do InterRed, um repositório de conteúdos pedagógicos digitais especialmente concebidos para a Rede Federal de Educação Profissional e Tecnológica (Silva 2008).

Qinghua (Qinghua 2008) adota um modelo baseado em Web Service chamado SOELA (*Service-Oriented E-Learning reference Architecture*) aplicado a uma ferramenta para educação a distância, o SOELS. O modelo é dividido em camadas e utiliza um E-learning Service Bus (ELSB) para orquestrar as competências disponíveis no ambiente. O SOELS implementa processos de negócios através da comunicação e colaboração entre serviços.

Tavares (Tavares 2010) apresenta uma arquitetura baseada no uso de Web-Services e Proxy para a integração do servidor WWW *Interactive Multipourpose Server* (WIMS) ao Moodle. O WIMS é uma ferramenta para a proposição de avaliações interativas capaz de rastrear informações relevantes sobre o comportamento do aluno durante a realização de exercícios. A arquitetura proposta por Tavares pode ser generalizada de maneira a permitir a integração de outras aplicações Web onde não se tem conhecimento de sua arquitetura ou da linguagem que foi desenvolvida e que a interação com a interface é o propósito da integração.

Em um contexto mais aproximado ao trabalho aqui apresentado, algumas iniciativas foram realizadas no sentido de integrar recursos de apoio a disciplinas de programação no 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 realizada 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, configurações essas, já disponibilizadas automaticamente no BOCA. A arquitetura utilizada pelo VPL não permite a adição de novas ferramentas ou o balanceamento de carga.

O Onlinejudge, também desenvolvido para gerenciar a submissão de códigos fontes, 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 no navegador. O Onlinejudge também pode ser executado localmente, dando suporte, nesse caso, apenas às linguagens C e C++. 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.

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.

Expandindo-se o escopo para práticas em laboratório de disciplinas de programação, identificam-se necessidades para as quais o BOCA não se adéqua. Um exemplo é a proposição de um problema cuja solução exija a integração de um conjunto de classes desenvolvidas pelo aluno a outras preexistentes, cadastradas pelo professor. O BOCA não permite o envio de mais de um arquivo para o mesmo problema e, adicionalmente, não prevê o cadastramento prévio de parte do código.

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 da Integração Baseada em Web Services

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 adaptado a outros AVAs.

Visando dar suporte à especificação das funcionalidades do BOCA-LAB a serem expostas por meio de WS, foi realizado, durante um semestre, o acompanhamento das

atividades de laboratório da disciplina Técnicas de Programação II, do Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará. Nesta turma foram avaliadas a metodologia adotada pelo professor para proposição e correção de problemas, bem como a maneira de apresentação de *feedback* ao aluno.

4.1. A Arquitetura da Integração

A arquitetura da integração é composta por três módulos que se comunicam através do protocolo SOAP usando mensagens criptografadas no formato XML: o Módulo de Integração (MI), o Módulo de Informação (MInfo) e o Módulo de Acoplamento BOCA-LAB (MAB). A Figura 1 mostra 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.

4.1.1 Módulo de Integração (MI)

O MI é responsável pela comunicação entre o Moodle e os demais módulos. Ele expõe as seguintes operações usando WS: o acesso a serviços de busca de servidores; o registro de enunciados de problemas computacionais e; o envio dos códigos fonte em resposta aos problemas propostos para os servidores MAB.

É também por meio desse módulo que se tem acesso ao serviço que disponibiliza os *feedbacks* gerados pelo BOCA-LAB através dos servidores MAB, para serem exibidos aos alunos na interface do Moodle.

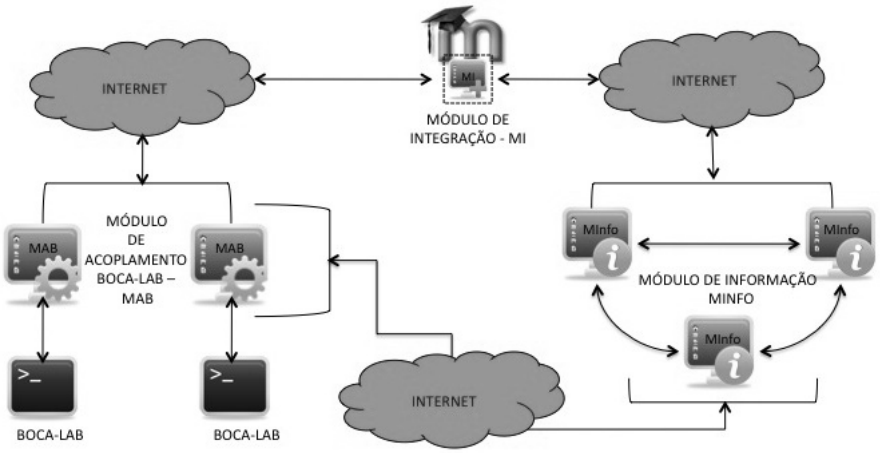

Figura 1 – Arquitetura da integração.

4.1.2 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 permite efetuar o balanceamento de carga.

Do ponto de vista educacional, o balanceamento de carga agiliza o processo de *feedback* para o aluno, evitando que um processo permaneça tempo desnecessário em filas em servidores sobrecarregados.

4.1.3 Módulo de Acoplamento BOCA-LAB (MAB)

O MAB é responsável pela comunicação entre o BOCA-LAB e os demais módulos da arquitetura. É através de serviços disponibilizados por ele que os dados dos problemas e os códigos fonte são recebidos e repassados para o BOCA-LAB. Ele é responsável também pela disponibilização do serviço de recuperação de *feedback* (acessado pelo MI) e pelo controle secundário da carga de compilação do BOCA-LAB, evitando o recebimento de requisições do MI caso o servidor esteja com sua carga máxima.

O BOCA-LAB, é um *script* que executa em *background* e localmente, atrelado a cada servidor MAB. Ele é o responsável pela compilação e comparação dos códigos fonte enviados, além do fornecimento dos *feedbacks*. Ele integra as adaptações ao sistema BOCA, como o envio de múltiplos arquivos para compilação e o cadastro de informações referentes aos alunos.

5. Interação com a Integração

Em seu curso, no ambiente Moodle, o professor pode adicionar uma atividade denominada "Envio de Arquivos para Compilação" que foi implementada e agregada ao Moodle para a administração da submissão de códigos fonte e problemas. A atividade implementada é uma interface entre o usuário (professor/aluno) e o MI. A nova tarefa permite a submissão de problemas por parte do professor, e a submissão de códigos fonte para avaliação e recuperação de *feedbacks* por parte dos alunos. Configuradas as informações dessa atividade, o professor pode cadastrar seu problema através de um formulário (Figura 2) contendo: a linguagem de programação; o nome da classe principal para compilação; um arquivo contendo as entradas para o problema e; um arquivo com as saídas padronizadas. Também é necessário definir o tempo máximo de execução para o *script* submetido, evitando que programas errados ou mal intencionados fiquem executando indefinidamente. Por fim é solicitada a data limite para a submissão de soluções para a atividade.

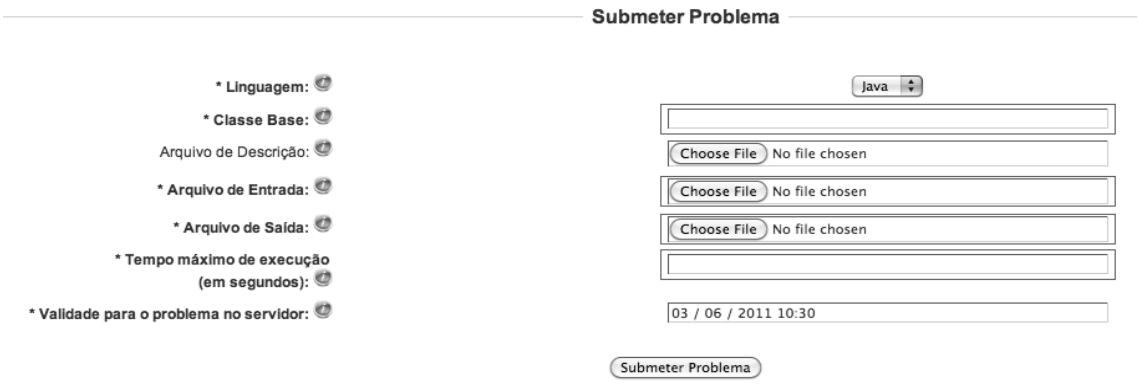

Figura 2 – Formulário de cadastro de um problema.

Após preenchimento do formulário, o MI envia os dados do problema para o MInfo que busca um MAB. Encontrado o servidor que melhor se adéqüe aos requisitos do problema, o MInfo retorna como resposta o endereço do MAB que fica registrado no MI para envio dos dados do problema. Após o envio do problema ao MAB escolhido, um

formulário é disponibilizado na interface do aluno para a submissão de seu código fonte, como mostrado na Figura 3.

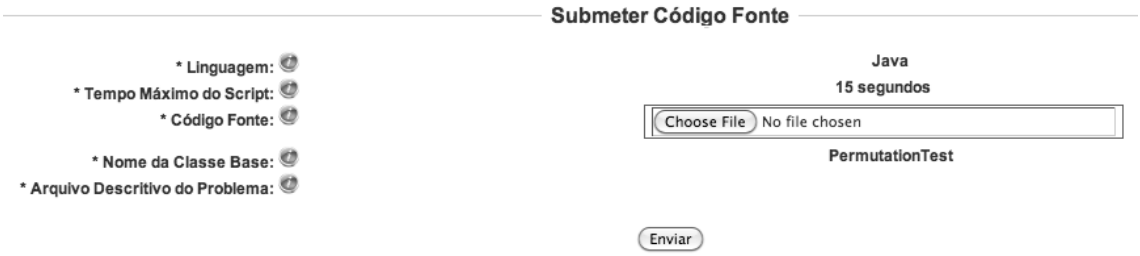

Figura 3 – Formulário de envio de código fonte.

Após o envio do código fonte pelo aluno, o MI remete o mesmo ao MAB, caso o serviço esteja *offline* ou sobrecarregado, o MI requisita ao MInfo a pesquisa de um novo servidor. Ao receber o endereço desse novo servidor MAB, o MI verifica se o mesmo já possui cadastrado em sua base o problema correspondente ao código fonte que será enviado. Caso não encontre, ele envia, além das informações do código fonte, as informações do problema para cadastramento no novo servidor. O endereço do novo servidor é, então, atualizado na base de dados do Moodle para futuras submissões para o mesmo problema.

A cada submissão, o servidor MAB executa um processo de atualização de sua carga no MInfo, de maneira que o mesmo permaneça sempre atualizado e, a cada código fonte processado, o servidor verifica localmente se sua carga está próxima de atingir o limite configurado. Caso isso ocorra, o servidor MAB atualiza a informação sobre o seu estado nos servidores MInfo, permitindo, assim, um melhor controle de carga por parte desse último. Esse mecanismo evita a sobrecarga da rede por mensagens de atualização individuais desnecessárias a cada fim de processamento de código fonte.

Após o envio do código fonte, a interface do aluno fica bloqueada para novas submissões ao mesmo problema até a compilação do código fonte já enviado e o acesso ao *feedback* da compilação pelo MI.

O endereço do servidor MAB é armazenado junto às informações do código fonte, permitindo recuperar o *feedback* da submissão. Caso o servidor esteja *offline*, e não seja possível recuperar o *feedback*, o MI reenvia o código fonte e o problema a ele associado para outro servidor a fim de obter 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.

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.

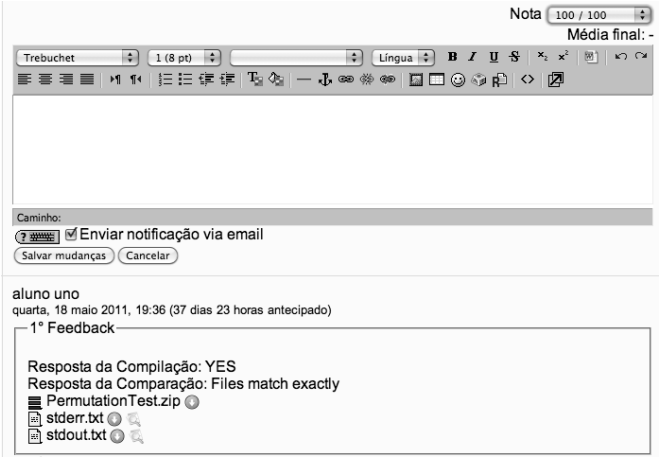

Figura 4 – Interface de atribuição de notas.

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.

6. Validação e Testes do Ambiente

Para validação do ambiente, especificou-se um cenário de testes em que participaram o professor do laboratório de programação e quatro alunos de mestrado e de iniciação científica. Foram utilizados problemas da disciplina Técnicas de Programação II. A interface de submissão de problemas, por ser bastante semelhante à interface de submissão de tarefas padrão do Moodle, não gerou dúvidas ao professor que já conhecia bem este ambiente.

Os participantes, no papel de alunos, também não apresentaram dificuldades em submeter o código, fazendo simulação de erros de compilação, de execução, loops infinitos e soluções corretas. Para simular as atividades usuais de um curso normal, foram adicionadas três outras atividades do tipo *offline* para registro das notas de avaliações escritas, além de uma atividade para entrega de um projeto final. As notas destas atividades foram simuladas.

Assim, a integração do BOCA ao Moodle propiciou um ambiente completo para a organização dos recursos da disciplina e composição das médias dos alunos por parte do professor.

7. Conclusões e Perspectivas

O ambiente BOCA-LAB, concebido como extensão do sistema BOCA, oferece recursos em forma de serviços para dar suporte a atividades de laboratório de programação. Integrado ao ambiente Moodle, as funcionalidades de compilação e correção automática de códigos fonte são disponibilizadas como tarefas do próprio AVA. A arquitetura proposta, pode ser generalizada de maneira a permitir a integração do BOCA-LAB a

diversos AVA, bastando, para isso, construir um módulo de integração (MI) específico à plataforma para consumir os serviços exportados pelo mesmo.

A integração do BOCA-LAB a um AVA como o Moodle visa diminuir consideravelmente a sobrecarga de trabalho na correção de códigos fonte por parte dos professores, bem como reduzir o tempo necessário para correção e apresentação dos resultados das atividades desenvolvidas em laboratório pelos alunos. Como conseqüência, espera-se a melhoria na qualidade do aprendizado de programação, visto que, por um lado, o tempo do professor com atividades de administração e de gestão de recursos pode ser reduzido e, por outro lado, espera-se maior disponibilidade do mesmo em termos de atenção ao aluno.

Ainda que outros fatores possam ser responsáveis por desistências ao longo do curso, pode-se considerar que, com o aumento da disponibilidade de tempo, o professor terá ampliado o seu poder de interferência em evitar evasões por desestímulo ou eventual sensação de abandono por parte do aluno.

A experimentação realizada com a turma de pós-graduação, utilizando problemas reais extraídos da disciplina de Técnicas de Programação II, permitiu validar o funcionamento da integração, bem como corrigir pequenos erros de implementação e alguns ajustes nos serviços especificados. Atualmente, uma nova experimentação está sendo realizada em uma turma de Introdução à Programação. Além disso, novos elementos estão sendo projetados para o enriquecimento da plataforma visando dar suporte a atividades ligadas ao desenvolvimento de aplicações WEB que possuem particularidades específicas, como a necessidade de uso de interfaces no lado cliente e bancos de dados no lado servidor.

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.

Dan, Asit; Johnson, Robert D.; and Carrato, Tony. (2008) “SOA Service Reuse By Design.” Proceedings of the 2nd international workshop on Systems development in SOA environments, p. 25-28. Mai, 2008.

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.

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

Moura, Simone L. de; et al. (2005) “Integrating Repositories of Learning Objects Using Web-Services to Implement Mediators and Wrappers.” Proceedings of the International Conference on Next Generation Web Services Practices. IEEE. 2005.

Oliveira, E. M. (2009). “Cooperação Inter-sistêmica em Apoio à Educação Profissional e Tecnológica”(2009). Dissertação (Mestrado)– Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará, Fortaleza/Ce (2009).

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.

Qinghua Zheng, Bo Dong, Feng Tian and Wei Chen (2008) “A Service-oriented Approach to Integration of e-learning Information and Resource Management Systems”, 978 -1-4244-1651-6/08 2008 IEEE.

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.

Tavares, Daniel Alencar B.; França, Allyson Bonetti; Soares, José Marques; Barroso, Natália Maria C.; Mota, João César M. (2010) “Integração do ambiente WIMS ao Moodle usando Arquitetura Orientada a Serviços e Compilação Automática de Médias.” RENOTE - Revista Novas Tecnologias na Educação, ISSN 1679-1916, Vol. 9 No 1 (2011), Porto Alegre - RS.

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