Eu Sei o que Vocês Fizeram (Agora e) na Aula Passada: o TSTView no Acompanhamento de Exercícios de Programação

Matheus Gaudencio, Leticia Farias Wanderley, Filipe Wesley Lemos, Eliane Cristina de Araújo, Jorge C. A. Figueiredo, Dalton D. S. Guerrero

[1]Laboratório de Práticas de Software
Universidade Federal de Campina Grande
Av. Aprígio Veloso, s/n, SPLab, Bodocongó
58429-900 – Campina Grande - PB – Brasil

matheusgr@copin.ufcg.edu.br,

{leticia.wanderley filipe.nunes}@ccc.ufcg.edu.br,

{eliane, abrantes, dalton}@dsc.ufcg.edu.br

Abstract. *In introductory programming practice classes, the teacher needs to have a global view of the performance and progress of the class in planned activities without losing sight of the individual progress and difficulties of each student. In this paper we present our experience with a toolset we developed to enable continuous monitoring of the activities of students in the laboratory. Our first experiences shows that the toolset makes it possible to detect more easily the difficulties of individual students, and problems in planned activities for the classroom. So, it has allowed us to act more precisely in solving problems of teaching and learning.*

Resumo. *Em aulas práticas de programação introdutória, o professor precisa ter uma visão global do desempenho e do andamento da turma nas atividades planejadas sem perder a visão individual do progresso e das dificuldades de cada estudante. Neste artigo apresentamos nossa experiência com um ferramental que desenvolvemos para viabilizar o contínuo monitoramento das atividades dos estudantes em laboratório. Nossas primeiras experiências demonstram que o ferramental torna possível detectar com mais facilidade as dificuldades individuais dos alunos, e os problemas nas atividades planejadas para a aula. Assim, tem nos permitido atuar de forma mais precisa e ágil na solução de problemas de ensino e aprendizagem.*

1. Introdução

Um dos problemas que o professor enfrenta no seu dia a dia é o de conseguir acompanhar efetivamente as atividades dos estudantes. Idealmente, os professores e demais mediadores (monitores, tutores) devem ter uma visão global do desempenho e do andamento da turma nas atividades planejadas, sem deixar de ter uma visão do progresso e das dificuldades individuais dos estudantes. É fato conhecido que mediadores capazes de perceber rapidamente dificuldades dos alunos têm maior chance de interceder em tempo hábil para maximizar a oportunidade de aprendizagem oferecida numa sessão de estudo.

DOI: 10.5753/CBIE.SBIE.2013.204

Neste artigo apresentamos uma das ferramentas que desenvolvemos para auxiliar o acompanhamento das atividades dos estudantes de uma disciplina de introdução à programação e o relato de seu uso ao longo de um semestre. Na disciplina, estimulamos os estudantes a resolver muitos exercícios de programação. Aliada a um planejamento detalhado das atividades em cada sessão de estudo, a ferramenta que apresentamos dá aos mediadores uma visão centralizada das atividades, com uma perspectiva global do desempenho da turma e uma visão individual de cada estudante.

O ferramental que utilizamos na disciplina combina dois serviços: 1) um serviço de coleta e testes automáticos dos programas submetidos pelos estudantes; 2) uma interface de visualização dos dados que facilita o acompanhamento das atividades dos alunos. O serviço de coleta e testes automáticos permite que o estudante submeta suas soluções candidatas para testes, de forma semelhante à usada em juízes online para maratonas de programação. Ao submeter uma solução, o estudante recebe feedback na forma de resultados de testes sobre seu código. Embora o estudante não receba os casos de teste propriamente ditos, o estudante sabe que tipos de erros ele cometeu e o número de casos de teste em que falhou.

O segundo serviço é o TSTView, que é o foco deste trabalho. O TSTView permite visualizar o desempenho dos estudantes em um dado período de tempo e/ou em qualquer subconjunto dos exercícios. A observação dos resultados dos testes realizados sobre todas as soluções enviadas para cada exercício permite, por exemplo, que o professor monitore em tempo real o desenvolvimento dos estudantes frente aos exercícios planejados.

Esse ferramental tem permitido detectar com mais facilidade e eficiência, dificuldades individuais dos estudantes, dificuldades comuns a grupos de estudantes e dificuldades gerais da turma. E, por consequência, tem nos permitido atuar de forma mais precisa em cada uma dessas situações: dificuldades individuais requerem abordagem individual, dificuldades de grupo provavelmente requerem abordagem coletiva e, finalmente, dificuldades de toda a turma requerem um reposicionamento do professor onde talvez seja necessário repensar as atividades planejadas para as aulas práticas ou a abordagem dos conteúdos nas aulas teóricas.

Há vários cenários vislumbrados quando acompanhamos os dados mostrados no TSTView. Pode-se assumir, por exemplo, que sequências de respostas enviadas que não passam nos testes indicam dificuldades enfrentadas pelo estudante em determinada questão ou conteúdo. Da mesma forma, pode-se assumir que estudantes que rapidamente são capazes de enviar soluções que passam nos testes dominam minimamente os conhecimentos e habilidades requeridas pelas questões. Por outro lado, exercícios em que grupos grandes de estudantes (em alguns casos toda a turma) tem dificuldades podem indicar problemas na elaboração da questão ou de seus testes ou, ainda, no próprio curso em termos da exposição a conteúdos e/ou das habilidades necessárias. Em todos os casos mencionados, é a informação produzida de forma instantânea que permite ao professor e mediadores a decisão embasada sobre como atuar junto aos estudantes e no curso.

Neste artigo, apresentamos nossa experiência na construção e no uso da ferramenta TSTView. Na Seção 2 apresentaremos o referencial teórico que corrobora com nosso planejamento metodológico em sala de aula e na aplicação da ferramenta no ensino de programação. Na Seção 3 apresentamos o TSTView, em seguida discutimos, na Seção

4, a nossa experiência de uso na disciplina de Programação 1 no acompanhamento dos alunos. Por fim, apresentamos as conclusões e direcionamentos para trabalhos futuros.

2. Referêncial Teórico

O curso em que a ferramenta foi aplicada faz uso de Python como linguagem de programação. De acordo com Fangohr, Python é indicado para disciplinas introdutórias por ser intuitiva e de fácil uso [Fangohr 2004]. Ainda, dentro da metodologia de ensino da disciplina, focamos no uso de exercícios para a atividade de prática do estudante, uma atitude que é comum no planejamento de cursos de programação [Chamillard and Braun 2000]. Esta é uma prática que apresenta sucesso nas disciplinas de programação. Korhonen encontrou uma boa correlação entre as notas de exercícios e o exame final da disciplina [Korhonen et al. 2002]. Este mesmo resultado é corroborado por Eliane et al. para uma disciplina oferecida em Python [de Araujo et al. 2013].

Considerando que a atividade prática através de exercícios é comum no estado da arte, há uma vasta discussão sobre ferramentas para o auxílio do ensino de programação [Aureliano and Tedesco 2012], e, especificamente, sobre a construção de sistemas de recepção de questões e feedback através de testes automáticos [Ala-Mutka 2005]. Ihantola [Ihantola et al. 2010] apresenta uma revisão sobre diferentes ferramentas com este objetivo. Entretanto, é inexistente a discussão específica sobre a visualização e acompanhamento dos dados obtidos e o seu impacto prático sobre a sala de aula, que é o propósito deste trabalho.

Este artigo apresenta, além da ferramenta, a discussão sobre a experiência de seu uso ao longo de um semestre para o acompanhamento das atividades dos alunos em uma turma de programação introdutória. Neste sentido, Borges sugere o acompanhamento individualizado dos alunos [Borges, M. A. F. 2000]. Um dos aspectos que exploramos é justamente a intervenção no aluno de forma individualizada, sem no entanto perder a visão global da turma.

3. TSTView

O TSTView é uma ferramenta que sumariza os dados relativos às sumissões das resoluções dos exercícios de programação realizados pelos estudantes. Cada resolução de exercício é um programa Python submetido a um testador automático que aplica os testes criados pelos professores para as questões propostas. Os dados de controle e os resultados dos testes aplicados às submissões compõem a base de dados para o TSTView. A ferramenta agrupa e sumariza tais informações e as apresenta em diferentes perspectivas para professores e alunos.

O TSTView tem dois módulos principais, o módulo do professor e o módulo do aluno. O primeiro mostra os dados recentes dos alunos que estão realizando alguma atividade no momento da consulta, dando uma visão mais geral da turma. Já o segundo, apresenta as informações em forma de um relatório histórico individual para cada aluno. Cada módulo requer acesso controlado mediante login que é efetuado através da plataforma Google App Engine, onde o sistema executa. Esta plataforma proporciona robustez quanto aspectos de segurança, desempenho e disponibilidade para a nossa aplicação.

3.1. Módulo do Professor

O módulo do professor permite uma visão geral das atividades realizadas pelos alunos. A aplicação disponibiliza filtros que permitem o foco da avaliação dos dados sobre um ou mais alunos, sobre turmas específicas, sobre determinados horários e datas, e sobre uma ou mais questões.

A página principal do módulo apresenta uma tabela que associa um aluno ao seu bloco de questões submetidas, em um dado intervalo de tempo. Por padrão, são apresentadas as questões produzidas no dia da consulta, a partir das 00:00 até a hora atual. A Figura 1 mostra a interface do módulo do professor. É apresentada uma lista com os alunos que estão em atividade no momento, quais questões foram submetidas, a corretude e a quantidade de submissões.

Figura 1. Módulo do Professor.

A interface ainda permite que o código de cada submissão seja visualizado na própria página do módulo, facilitando o entendimento das soluções dos alunos e tornando possível uma ajuda imediata em sala de aula ou através de um email com comentários para o aluno. A interface para visualização de código e envio de mensagem é exibida na Figura 2.

Uma das formas de filtro fornecidas pela ferramenta é a análise por questões, muito usada na prática quando da aplicação de minitestes nas aulas de laboratório. Para tanto, é criada uma tabela com as colunas representando as questões e as linhas representando os alunos, como mostra a Figura 3. Nesta visão, cada célula da tabela apresenta o número de submissões realizadas pelo aluno para aquela questão e a cor da célula indica se alguma das submissões enviadas pelo aluno obteve sucesso (verde) ou não (vermelho).

3.2. Módulo do aluno

O módulo do aluno consiste em uma sumarização geral de todas as resoluções de exercícios enviadas por um estudante. Com os dados, é possível analisar as questões com

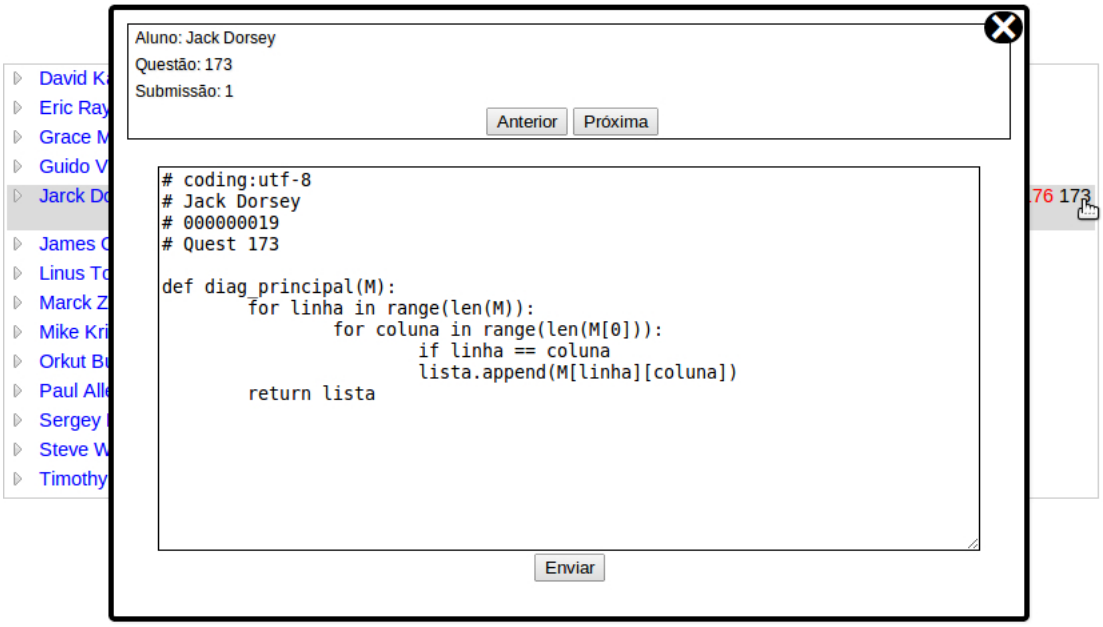

Figura 2. Visualização e edição do código da submissão.

	177	181
Ada Lovelace	1	2
Alan Turing	1	2
Bill Gates	2	1
David Karp	1	1
Eric Raymond		
Grace Murray Hoper		
Guido van Rossum		
Timothy John Berners-Lee	4	2

Figura 3. Visão por questões x alunos.

erros e não resolvidas, dando ao aluno e professor a possibilidade de diálogo e reflexão baseado no histórico de atividades do estudante.

A página principal deste módulo apresenta um cabeçalho com informações gerais do aluno. Em sequência, mostra informações do progresso do aluno com dados sobre as questões submetidas com sucesso, questões submetidas com erro ou questões ainda não tentadas. Além disso, é apresentado um sumário, à direita, sobre questões mais submetidas pela turma e pelo aluno. Estas informações estão apresentadas na Figura 4.

O sumário reflete uma análise superficial e quantitativa das questões que o aluno ainda não submeteu com sucesso. Estes dados numéricos fazem um contraste entre as questões produzidas pelo aluno e as ”questões mais submetidas”pela turma considerando vários aspectos. Esta informação não precisa ser vista como sugestão de quais questões o aluno deve fazer, mas pode ser usada para que, junto com o professor, o aluno reflita sobre as questões ainda não realizadas com sucesso. A Tabela 1 apresenta o significado de cada código usado no sumário. O aluno, pode avaliar quais as questões que ele mais

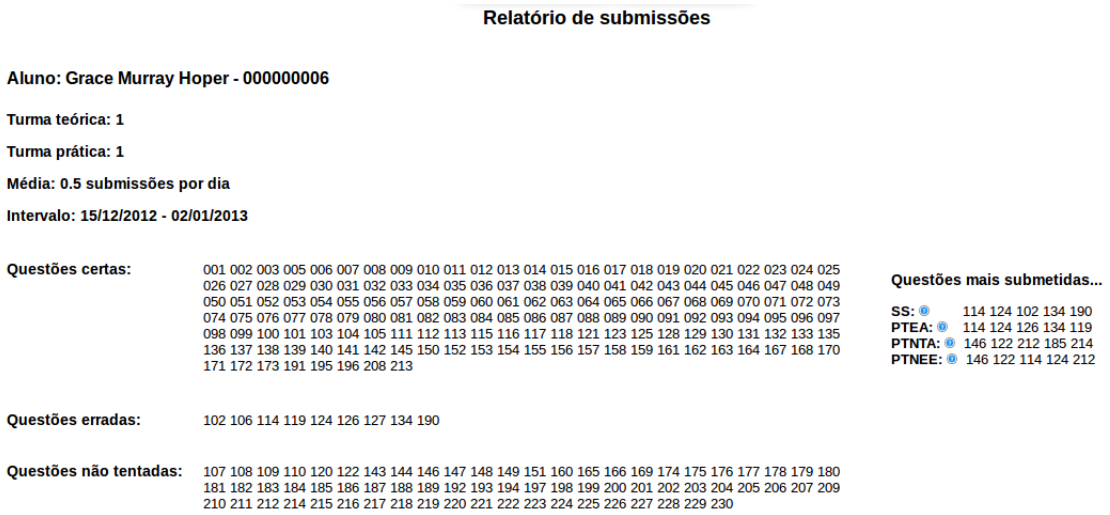

Figura 4. Módulo do Aluno.

tem tentado ainda sem uma submissão com sucesso (SS) ou ainda avaliar que questões o próprio aluno não submeteu, ou submeteu com erro, que a turma como um todo tem mais submetido com sucesso (PTEA, PTNTA e PTNEE).

Tabela 1. Significado dos códigos do sumário

Código	Significado
SS - Sem Sucesso	Questões que o aluno mais tentou submeter ainda sem sucesso
PTEA - Pela Turma e Erradas pelo Aluno	Questões que a turma mais enviou, mas que o aluno ainda não conseguiu sucesso
PTNTA - Pela Turma e Não Tentadas pelo Aluno	Questões que a turma mais enviou, mas que o aluno ainda não tentou
PTNEE - Pela Turma e Não Tentadas ou Erradas	Questões que a turma mais enviou, mas que o aluno ainda não tentou ou não conseguiu obter sucesso

Em seguida, são exibidas informações específicas sobre as submissões e um gráfico do progresso do aluno ao longo do tempo, como mostra a Figura 5. Na informação sobre as submissões, o aluno pode encontrar os resultados de testes de cada submissão. O gráfico da figura mostra a quantidade de submissões, considerando as corretas e as incorretas, do aluno ao longo do tempo juntamente com a média de submissões da sua turma. Esta mesma informação pode ser apresentada, também, considerando apenas o tempo de atividade nas aulas práticas de laboratório, tendo o horário sido previamente cadastrado no sistema, ou fora dos horários de aula.

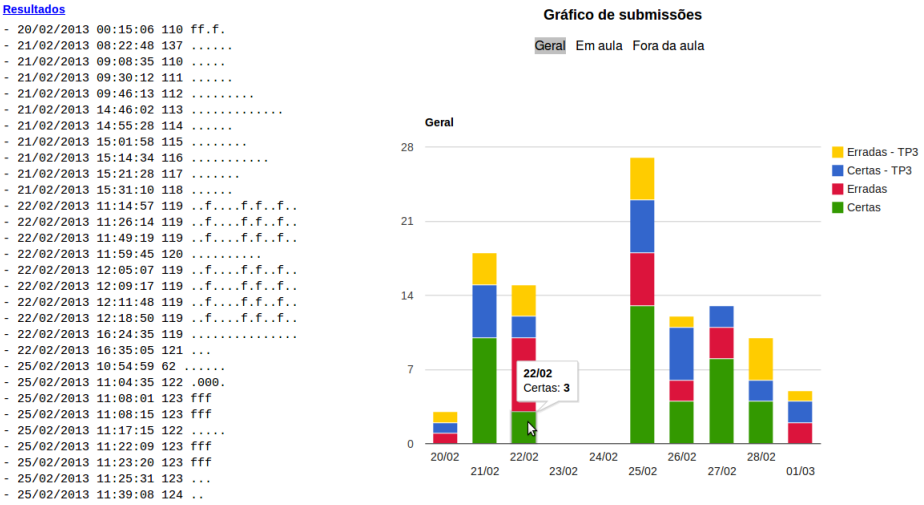

Figura 5. Gráfico e Resultados de Submissões.

4. Avaliação Prática

Nossa ferramenta foi utilizada na disciplina Programação 1, do segundo semestre do curso de Bacharelado em Ciência da Computação da Universidade Federal de Campina Grande em 2012. Os dados foram capturados entre os dias 29/11/2012 e 13/05/2013. No total, 105 alunos, entre novatos e repetentes, fizeram parte desta avaliação divididos em 5 turmas práticas. Nossa lista de exercícios foi composta de 258 questões, tendo nosso sistema recebido um total de 31.908 submissões dos alunos. Destas 258 questões, 68 representaram mini-testes presenciais aplicados para contabilização de nota para a disciplina (média de 13,6 questões por turma). Durante os mini-testes, os alunos utilizaram um ambiente protegido que permitia o acesso apenas ao enunciado, ao serviço de submissão de respostas e aos resultados de teste.

Durante o semestre, a equipe de ensino e os alunos fez uso intensivo da ferramenta de visualização para consultar informações relativas ao progresso dos estudantes na disciplina. Os dados apresentados pelo TSTView são atualizados automaticamente a cada 10 segundos. Nas 12 horas de maior atividade monitorada, o TSTView recebeu em torno de 16 mil requisições, o que representa, em média, 27 usuários a cada instante de tempo. Este e demais picos de utilização aconteceram justamente durante as aulas práticas, onde era recomendado o uso da ferramenta para acompanhmento das atividades realizadas.

Durante aulas práticas de laboratório, dedicadas principalmente à resolução de exercícios, professores e monitores utilizaram a ferramenta para identificar, principalmente, alunos com mais dificuldades. Para tanto, focamos naqueles alunos que apresentavam poucas submissões quando comparados aos demais alunos ou diversas submissões em sequência com erros. Era também dada uma maior atenção a alunos que eram identificados trabalhando em questões mais antigas da lista de exercícios, o que denotaria um atraso na apreensão do conteúdo, com relação a outros alunos da turma.

Alunos nessas situações eram chamados para conversar com seu respectivo professor de forma a rever quais as principais dificuldades do processo de aprendizagem e receber uma intervenção direcionada. Durante esta conversa, o relatório individual do aluno era utilizado para discutir o histórico de submissões e a prática de estudo de

aluno ao longo do semestre. O aluno que apresentava poucos exercícios submetidos era confrontado com essa realidade, e iniciava-se um diálogo sobre um novo planejamento de ação perante a disciplina.

Uma outra função para a ferramenta foi a identificação das dificuldades gerais de uma turma. Por exemplo, se durante uma atividade, em que todos começam a fazer, ao mesmo tempo, uma determinada questão e a maioria dos alunos não consegue resolvê-la, nós realizávamos uma intervenção em grupo explicando o conteúdo requerido ou revisávamos o enunciado da questão ou os seus testes de forma torná-la acessível aos alunos.

Uma prática comum emergiu durante a aplicação de mini-testes. Originalmente propostos para serem realizados num intervalo de 30 minutos, os mini-testes recebiam uma extensão de prazo e uma orientação geral sobre o enunciado quando detectávamos que poucos alunos estavam submetendo respostas. Para avaliar o impacto desta prática, nós analisamos os dados e construímos o gráfico apresentado na Figura 6. Nesta figura, as questões de mini-teste foram ordenadas de acordo com o tempo total (considerando a extensão) dado para cada questão (valor indicado pela linha do gráfico). Nela, o eixo Y representa, para cada barra, a quantidade de alunos que submeteram cada questão divididos estes em duas categorias: i) aqueles que submeteram uma resposta nos primeiros 30 minutos do mini-teste e os que submeteram após este intervalo de tempo.

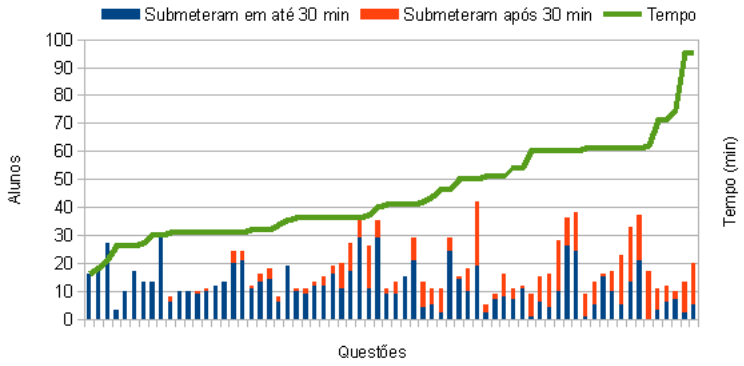

Figura 6. Tempo de Mini-testes e Submissões dos Alunos Antes e Depois de 30 Minutos de Atividade.

É possível observar que foi difícil seguir o tempo planejado de 30 minutos para a realização de cada mini-teste. Com a visualização rápida oferecida pela ferramenta do progresso dos alunos, intervimos na própria forma de avaliação respeitando o ritmo de produção dos alunos e corrigindo eventuais problemas na complexidade de um enunciado ou até mesmo com erros de especificação.

É possível ver que na mediana, os mini-testes foram realizados em 41 minutos. As questões com extensão apresentaram em média, 36,56% dos alunos submetendo apenas após a primeira meia hora de tempo, permitindo uma avaliação mais inclusiva da turma.

O TSTView mostrou-se uma ferramenta eficiente no acompanhamento das atividades e da produção dos alunos. Uma vez que proporciona uma visualização imediata do que está acontecendo durante as atividades, permite ações individuais para melhorar

o processo de aprendizagem, bem como rever o próprio plano pedagógico. Nossa experiência em sala foi positiva e nos auxiliou em diversos aspectos. Em especial, como relatado, permitiu identificar uma necessidade de atuação no processo de avaliação da disciplina.

5. Conclusões

Neste artigo apresentamos nossa experiência com uma das ferramentas que desenvolvemos para o acompanhamento das atividades dos alunos na disciplina de Programação 1. Nosso ferramental foca em oferecer ao mediador do processo de aprendizagem informação útil sobre o acompanhamento das atividades dos estudantes primeiramente numa visão global mas dando oportunidade para a análise individual. A utilização do TSTView em aulas práticas no laboratório de programação permite que o mediador observe a turma em uma posição privilegiada. O TSTView mostra se as submissões dos alunos para as atividades planejadas para a aula estão erradas ou certas, de acordo com os testes automáticos. Assim, é possível observar a evolução ou não dos alunos em questões específicas e intervir mais precisamente e precocemente em grupos de alunos com dificuldades. O TSTView pode ser visto como um painel, onde as informações que trafegam entre os alunos e o sistema de automático de submissão e testes são visualizados de forma sumarizada e instantânea.

Outra possibilidade trazida pela ferramenta é a de visualizar um relatório detalhado de cada estudante e de sua produção de exercícios de programação. Esta abordagem retrospectiva, permite que analisemos os estudantes de forma detalhada e individualizada. Nesta perspectiva é possível identificar quadros típicos de alunos que estão adotando comportamentos de risco, como exemplo, realizando poucos exercícios, realizando exercícios apenas em sala de aula, realizando exercícios errados sem retomá-los, etc. Assim, a ferramenta auxilia na orientação dos alunos de forma mais embasada e na reflexão de suas atitudes frente ao curso. Ao passo que é possível confrontá-los com as suas próprias dificuldades, o TSTView também permite o contraste com a produção da turma.

Durante um semestre, utilizamos o TSTView na disciplina de Programação 1. Embora não se trate de uma avaliação formal, discutimos neste artigo a nossa experiência e impressões de uso. No processo de ensino, foi possível, ao professor, atuar de forma rápida nos alunos que mais precisavam de uma intervenção dialógica, por vezes confrontando sua própria realidade. Ao mesmo tempo, o professor pode rever práticas próprias de ensino, como, por exemplo, ser capaz de se adequar a diferentes ritmos de produção de exercícios dos alunos.

Ao longo do uso da disciplina, também foi possível obter retorno dos alunos quanto a possíveis práticas que podem ser aplicadas ao nosso ferramental de forma incentivar o seu uso pelos alunos e pelo professor. Como sugestões de trabalhos futuros, pretendemos incorporar na ferramenta, a possibilidade de interação entre os alunos que assim desejarem comparar suas produções de exercícios em grupos (ranking) definidos pelos próprios estudantes. Também queremos expandir o campo de diagnóstico utilizando técnicas de mineração de dados de forma poder sugerir ao aluno questões relevantes que possam ser realizadas pelo mesmo quando não for possível a intervenção de um mediador para sugerir tal ação.

Referências

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102.

Aureliano, V. C. O. and Tedesco, P. C. d. A. R. (2012). Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no sbie e wie. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 23.

Borges, M. A. F. (2000). Avaliação de uma Metodologia Alternativa para a Aprendizagem de Programação. In *Workshop de Educação em Computação, Congresso anual da SBC 2000*, Curitiba, Brasil. SBC.

Chamillard, A. T. and Braun, K. A. (2000). Evaluating programming ability in an introductory computer science course. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, SIGCSE '00, pages 212–216, New York, NY, USA. ACM.

de Araujo, E. C., Gaudencio, M., Menezes, A., Ferreira, I., Ribeiro, I., Fagner, A., Ponciano, L., Morais, F., Guerrero, D. S., and Figueiredo, J. A. (2013). O papel do hábito de estudo no desempenho do aluno de programação. In *Workshop de Educação em Computação, Congresso anual da SBC 2013*, Maceió, Brasil. SBC.

Fangohr, H. (2004). A comparison of c, matlab, and python as teaching languages in engineering. In Bubak, M., Albada, G., Sloot, P., and Dongarra, J., editors, *Computational Science - ICCS 2004*, volume 3039 of *Lecture Notes in Computer Science*, pages 1210–1217. Springer Berlin Heidelberg.

Ihantola, P., Ahoniemi, T., Karavirta, V., and Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA. ACM.

Korhonen, A., Malmi, L., Myllyselkä, P., and Scheinin, P. (2002). Does it make a difference if students exercise on the web or in the classroom? In *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, ITiCSE '02, pages 121–124, New York, NY, USA. ACM.