Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

Aprendendo a Programar em Grupo

Thais Castro[1,2], Alberto Castro[2], Hugo Fuks[1]

[1]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
R. M. S. Vicente, 225 - Gávea - 22453-900 - Rio de Janeiro, RJ, Brasil

[2]Departamento de Ciência da Computação, Universidade Federal do Amazonas
Av. Gal. R. O. J. Ramos, 3000 - 69077-300 - Coroado - Manaus, AM, Brasil

{tcastro,hugo}@inf.puc-rio.br, alberto@ufam.edu.br

Abstract. *The technical literature in cognitive science informs that working in groups reaps more benefits than working alone. We are seeing a variety of innovative group programming methodologies like the agile methods. This work examines the findings of a pilot study carried out during the first academic semester of 2007. The subjects were 110 freshmen computing and engineering students taking their first programming course at the Federal University of Amazonas. Based on these findings, it is proposed a programming progression learning scheme, from individual to group programming. Such transition is necessary for students are not used to work in group. In order to evaluate such progression learning scheme a case study is outlined.*

Resumo. *A literatura técnica em ciência cognitiva informa que o trabalho em grupo proporciona maiores benefícios que o trabalho individual. Há inúmeras metodologias inovadoras para programação em grupo, como por exemplo, os métodos ágeis. Este trabalho examina os resultados de um estudo piloto aplicado no primeiro semestre acadêmico de 2007. Os sujeitos foram 110 calouros de computação e engenharia matriculados em seu primeiro curso de programação. Baseado nestes resultados, um esquema de progressão de aprendizagem de programação em grupo é proposto, o qual prevê uma transição gradual da prática individual para a programação em grupo. Tal transição é necessária, pois os estudantes não estão acostumados a trabalhar em grupo. A fim de avaliar tal esquema de progressão de aprendizagem, é proposto um estudo de caso.*

1. Introdução

A busca por conhecimento sobre as habilidades de programação é recorrente nesta área de pesquisa, iniciando por Dijkstra [Dijkstra, E. 1982] em seu trabalho "On the Teaching of Programming" e Weinberg [Weinberg, G. 1971], o qual aborda o aspecto psicológico da aprendizagem de programação. Embora haja muita pesquisa no assunto, os marcos de aprendizagem de programação não estão plenamente elicitados e é por isso que é difícil entender quais elementos contribuem para a aquisição de conhecimento em programação.

Quanto à aprendizagem, independentemente da área considerada, é sabido [Sharan, S. 1999] que colaboração promove nas pessoas o desenvolvimento de habilidades e

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

estratégias para resolução de problemas que são de extrema importância para a construção de conhecimento naquele domínio. Este é o motivo pelo qual nós pretendemos adaptar essas técnicas de colaboração ao domínio de aprendizagem de programação.

Colaboração tem se tornado uma necessidade tanto no mercado de trabalho quanto na educação. Mais especificamente em empresas desenvolvedoras de software, a demanda crescente por produtos e serviços leva a uma competição exacerbada, fazendo com que essas empresas busquem convergência em suas atividades. Isto se reflete na formação de times de desenvolvimento e na necessidade de colaboração no time e entre times. Com respeito à educação, quando os estudantes estão colaborando, eles podem ver o ponto de vista de seus pares para construir uma solução unificada para o problema em questão. É mais fácil perceber o efeito da colaboração quando se tem acesso aos registros em um ambiente virtual de aprendizagem, podendo se perceber uma diferença substancial nos refinamentos das soluções, como descrito em [Almeida, F., Castro, T. e Castro, A. 2006].

Em aprendizagem de programação é necessário se adotar um modelo ou esquema para sugerir o desenvolvimento de estratégias e o acompanhamento do desenvolvimento de programas em grupo. Tal modelo deve servir a ambos os domínios: resolução de problemas e colaboração. Como não foi possível encontrar esse tipo de modelo na literatura, a revisão bibliográfica apresentada na próxima seção é baseada em aprendizagem de programação, tanto individual quanto em grupo. Então, na seção seguinte, há a descrição de um estudo piloto para programação em grupo desenvolvido no primeiro semestre acadêmico de 2007. Finalmente, como resultado da análise desse estudo piloto, é proposto um esquema de progressão da aprendizagem de programação seguido pelas especificações de um estudo de caso para aprendizagem de programação em grupo.

2. Aprendizagem de Programação em Grupo

A aprendizagem de conceitos e métodos para a construção de programas de computador não é trivial, à medida que requer o uso de habilidades de alto nível e raciocínio muito abstrato. Em [Dijkstra, E. 1982] é enfatizado que programação envolve mais raciocínio que qualquer outra habilidade. Porém, programação também é uma tarefa de engenharia, uma vez que lida com a produção de artefatos que precisam satisfazer requisitos de qualidade e serem submetidos a uma verificação.

Em [Eckerdal, A. e Berglund, A. 2005] é sugerido que os estudantes sejam expostos a técnicas de resolução de problemas em cursos introdutórios de programação. O que normalmente ocorre é que os estudantes encontram muita dificuldade em aplicar suas habilidades previamente adquiridas. Conseqüentemente, isto se torna uma fonte de medo e frustração, resultando em evasão do curso. O trabalho descrito em [Clancy, M. et al 2003] apresenta um esforço de desenvolver um novo curso de programação baseado em sessões de laboratório, em vez de aulas expositivas. Neste curso, uma série de atividades foi planejada como discussões on-line, exercícios de programação em duplas (pair programming), leitura de textos pela Internet, anotações de reflexão, entradas em diário e colaborações utilizando o processo de revisão por pares para criticar as respostas dos colegas a um dado tópico.

O artigo anterior [Clancy, M. et al 2003] trata da mudança de uma metodologia envolvendo aulas teóricas e práticas para outra que utiliza somente aulas práticas, com atividades distintas bem distribuídas entre as sessões. Apesar da preocupação com o

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

desenvolvimento de questionários para criar o hábito na reflexão nos estudantes, tal metodologia se mostrou ineficaz para detectar confusão na apreensão de conceitos. Analisando-se os exercícios de programação em pares, nota-se que não há evidências de uma melhoria no desempenho dos estudantes resultante da aplicação desta técnica, uma vez que não houve registro das atividades das duplas ou mesmo um grupo de controle.

Seguindo a linha de avaliação da aprendizagem, o trabalho apresentado em [Chamillard, A. e Braun, K. 2000] descreve uma combinação de algumas técnicas de avaliação em um curso de introdução à computação e demonstra por meio de análises estatística as diferenças e relacionamentos entre essas técnicas. O curso foi planejado seguindo a premissa que antes de aprenderem a programar, os estudantes devem ser capazes de resolver problemas. Portanto, primeiramente os estudantes resolvem problemas sem o uso de uma linguagem de programação e, posteriormente, aprendem como utilizá-la para representar soluções.

Após a fase inicial de resolução de problemas, o curso continua com seis sessões de laboratório onde os estudantes devem resolver problemas individualmente, sendo permitido que consultem seus pares sempre que acharem necessário. É importante mencionar que a complexidade desses problemas aumenta conforme as sessões avançam. Uma vez que esta fase acaba, pequenos grupos (2 a 4 integrantes) propõem um estudo de caso consistindo no desenvolvimento de um programa (normalmente um jogo). Uma avaliação da aprendizagem é conduzida comparando estatisticamente o desempenho dos estudantes nas sessões de laboratório, que envolvem a concepção do estudo de caso e as práticas individuais controladas e sem consultas (como os testes) desenvolvidas duas vezes ao longo do curso.

A contribuição do trabalho supracitado é a análise estatística das correlações entre os mecanismos de avaliação utilizados. Mesmo considerando a importância de saber se os estudantes estão sendo avaliados pelos métodos mais adequados, considerando aprendizagem, outros fatos são desconsiderados. Por exemplo, não há controle sobre o desenvolvimento de cada trabalho, tornando impossível de averiguar se uma dada tarefa foi desenvolvida por somente um estudante, o que causaria um erro nas correlações.

Novamente utilizando modelos de avaliação para ensinar programação, o trabalho descrito em [Lister, R. e Leaney, J. 2003] utiliza a pré-avaliação dos estudantes como uma base para categorizá-los em estágios de aprendizagem conforme a definição na taxonomia de Bloom [Krathwohl, D. 2002]. A partir daí, o curso é formatado de modo que possibilite oferecer atividades diferenciadas para os estudantes nos diferentes estágios de treinamento.

Em um esforço para identificar aspectos que facilitem a aprendizagem de programação, o trabalho descrito em [Eckerdal, A. e Berglund, A. 2005] afirma que através das respostas dos estudantes a questões como “o que é programação?” é possível determinar uma ordem para apresentação dos paradigmas de programação. Os autores partem do princípio que os estudantes precisam conhecer o que aprendizagem de programação realmente é a fim de que possam aprender. A maioria das respostas sintetizadas no trabalho sugere que aqueles estudantes precisam primeiramente ser expostos a processos de raciocínio mais estruturados antes de serem apresentadas as abstrações de objetos da programação orientada a objetos.

O que é realmente necessário para facilitar a aprendizagem de programação ainda é uma questão em aberto. Apesar de os trabalhos acima mencionados procurarem encontrar uma resposta para essa questão, não há trabalhos na literatura examinada que tenham sido bem sucedidos em estabelecer métodos e técnicas comprovadamente

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

eficazes para aprendizagem de programação em grupo. Por outro lado, há iniciativas cujo foco é criar e manter o interesse dos estudantes no curso utilizando conceitos de extreme programming, já amplamente utilizados por times de desenvolvimento nas empresas de software.

3. Programação em Grupo: Estudo Piloto

No primeiro semestre acadêmico de 2007, um estudo piloto [Castro, T. et al 2008] foi desenvolvido com duas turmas de estudantes cursando sua primeira disciplina introdutória de programação. Esses estudantes eram calouros dos cursos de Ciência e Engenharia da Computação da Universidade Federal do Amazonas (UFAM).

O objetivo desse estudo piloto era observar como os estudantes desenvolvem soluções para problemas complexos através da distribuição de tarefas, negociação, composição de soluções parciais e refinamentos sucessivos. Isto foi atingido por meio de um trabalho em grupo (cerca de cinco integrantes) os quais foram também responsáveis por registrar as atividades desenvolvidas ao longo dos diversos estágios do trabalho, utilizando um ambiente para controle e análise de versões de código, o AAEP [A., Castro, T. e Castro A. 2006]. Ao final do curso, como atividade final, foi proposta uma atividade em grupo. Sua solução (ou conjuntos de soluções) precisava ser acrescida do registro das interações entre os integrantes da equipe. Então, partiu-se para a análise e a interpretação desses dados, baseados nos processos de classificação, codificação e tabulação.

Conforme descrito em [Castro, T. et al 2008], os grupos vivenciaram muitas dificuldades, muitas das quais relacionadas à codificação da solução proposta na linguagem Haskell. Embora o planejamento da solução tenha sido difícil para os estudantes em decorrência da necessidade deles em se coordenarem e interagirem, a principal dificuldade relatada estava relacionada à habilidade em usar técnicas de programação e conhecimento da linguagem específica. Isto pressupõe que aprendizagem de programação pode ser mais eficaz quando conduzida em grupo, seguindo um modelo ou esquema que facilite este processo.

Além de desenvolver código e manter o registro das interações, os estudantes também responderam um questionário, que foi analisado quantitativamente. Finalmente, eles expuseram livremente suas dificuldades gerais e as relacionadas aos estágios de desenvolvimento, que por sua vez, foi analisado qualitativamente.

Considerando a reflexão do professor neste processo, em geral o desempenho dos grupos foi satisfatório. Comparando-se os códigos implementados e seus relatórios com as anotações do professor é possível encontrar uma correlação, a qual freqüentemente indica dificuldades na composição das partes dos programas que foram desenvolvidas individualmente em módulos independentes a serem construídos e refinados como um único programa.

A análise deste estudo piloto evidencia que a programação em grupo é prazerosa para os estudantes, aumentando por esta razão seu interesse pelas atividades propostas. Isso também sugere a importância de uma transição gradual de tarefas individuais para colaborativas, uma vez que os estudantes relataram muitas dificuldades em desenvolverem seus projetos em grupo.

4. Um Plano de Progressão do Individual para o Colaborativo

Conforme descrito nas seções anteriores, já houve muitas tentativas de facilitar a aprendizagem de programação. Algumas iniciativas são voltadas para mais aulas

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

práticas associadas com um acompanhamento contínuo, enquanto outras procuram manter o foco em exercícios especialmente preparados. Em todo caso, todos os artigos analisados ainda revelam lacunas significativas em relação à aprendizagem de programação.

Baseados em uma experiência de 15 anos com aprendizagem de programação [Castro T. et al 2002], uso de métodos de colaboração para representar conhecimento sobre resolução de problemas [Mendonça et al 2002] [Pereira et al 2002] e [Silva et al 2002] e este estudo piloto, acreditamos que programação em grupo é uma tarefa difícil, principalmente porque os alunos não estão acostumados a trabalhar em grupos. Tal é a razão para se utilizar um modelo como guia de uma atividade tão complexa como programação. Caso este modelo não exista, faz-se necessário desenvolver algo como um plano de progressão para aprendizagem de programação, enfatizando a colaboração, como o que é proposto em seguida.

A análise de um estudo de caso descrita em [Almeida A., Castro, T. e Castro A. 2006] resultou na concepção e desenvolvimento do AAEP, uma ferramenta para acompanhamento das versões dos códigos dos alunos. O AAEP provê, conforme a necessidade do professor, a recuperação de uma comparação entre quaisquer dentre duas versões de código de um único aluno para um problema. Esta ferramenta foi projetada para atender essencialmente às demandas do professor. Conseqüentemente, o acesso às funcionalidades de gerência tais como registro de alunos e problemas e análise de soluções são serviços restritos a ele (professor). As outras funcionalidades tais como elaboração de códigos fonte de programas, edição e testes associados a comentários que caracterizam cada versão são acessíveis a ambos, professores e alunos.

Utilizou-se o AAEP [Almeida A., Castro, T. e Castro A. 2006] em aula para apoiar a programação, registrando tanto os códigos desenvolvidos individualmente quanto os desenvolvidos em grupo. Isto se tornou imprescindível na tentativa de adaptação de um modelo colaborativo de forma a atender as especificidades de aprendizagem de programação, que envolve as seguintes ações:

- Verificar se os alunos procuram códigos existentes para problemas que julgam similares, antes de iniciar a codificação da solução para um problema específico, o que seria uma possível evidência da importância da resolução de problemas e programação por exemplos;
- Tornar o processo de planejamento da solução mais explícito, revelando, assim, a importância do registro de todas as versões de código;
- Verificar se os alunos reutilizam suas próprias versões de código anteriores;
- Observar se as interações dos grupos levam a soluções em menos tempo ou mais eficientes;
- Observar as incorporações à prática, do individual para a colaboração, enfatizando as possíveis mudanças de comportamento;
- Analisar o comportamento dos alunos no grupo.

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

FASE / SEMANA	Preparação	Codificação Individual 1	Codificação Individual 2	Codificação em Grupo 1	Codificação em Grupo 2	Codificação em Time
1						
2						
3						
4						
5			Definição dos Grupos			
6						
7				Prova Parcial		
8						
9						
10						
11						
12						
13						
14						
15						Prova Final
16						Avaliação do Curso

Figura 1 - Plano de Progressão de Aprendizagem Colaborativa de Programação

A Figura 1 ilustra um plano de progressão de aprendizagem colaborativa que define uma progressão da programação individual para a em grupo, distribuída ao longo de 16 semanas na disciplina introdutória de programação. Neste cenário, as praticas se iniciam na fase de **Preparação**, envolvendo sessões de laboratório individuais, consistindo de problemas introdutórios e esclarecimentos quanto à metodologia. A segunda fase, **Codificação Individual 1**, consiste ainda na busca de soluções e registros individuais, embora para problemas um pouco mais complexos. Na fase seguinte, **Codificação Individual 2**, inicia-se o trabalho em grupo como uma preparação para soluções colaborativas. Nesta fase, os integrantes do grupo codificam individualmente e escolhem a melhor solução para ser a solução do grupo. Na próxima fase, **Codificação em Grupo 1**, a resolução de problemas se torna um pouco colaborativa: o professor determina as tarefas (divisão em módulos por funcionalidade) e os grupos determinam os atores responsáveis por cada atividade. Na fase de **Codificação em Grupo 2**, o grupo também é responsável pela divisão de tarefas. Finalmente, na fase de **Codificação em Time**, há um exercício de laboratório, no qual os alunos devem resolver um problema, em seus grupos, no estilo de uma maratona de programação, o que simula uma situação real de resolução colaborativa de problemas, conforme ocorre no mercado de trabalho em tecnologia da informação.

Este plano foi recentemente avaliado na Universidade Federal do Amazonas, em um estudo de caso realizado na disciplina introdutória de programação, o qual é descrito a seguir. A escolha pela divisão em 6 fases foi baseada em nossa experiência de 15 anos com ensino de programação utilizando linguagens funcionais. Esta divisão é adequada para um curso de 75 horas-aula, ministrado em um semestre letivo.

As fases de resolução de problemas descritas na Figura 1 seguem esta seqüência e número de fases com base em nossa experiência com ensino de programação e como resultados de estudos de caso conduzidos especialmente para se medir a interferência do uso de métodos colaborativos na representação de conhecimento sobre resolução de problemas [Mendonça, A. et al 2002] [Pereira, V. et al 2002] [Silva, L. et al 2002]. Constatou-se que este número de fases para resolução de problemas é adequado a uma disciplina de graduação em programação introdutória de no mínimo 60 horas.

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

5. Estudo de Caso em Aprendizagem de Programação em Grupo

A fim de se definir os próximos passos desta pesquisa, um projeto de prospecção foi elaborado e aplicado para a identificação e categorização dos comportamentos em aprendizagem de programação em grupo, seguindo o plano de progressão de aprendizagem de programação em grupo definido no final da seção anterior. Dentre outras possibilidades, a aplicação deste plano de progressão pode prover subsídios para a elicitação de requisitos e posterior aplicação em ações relacionadas ao desenvolvimento de groupware para este domínio.

Considerando o que se tornou evidente nos estudos de caso anteriores, os estudantes geralmente iniciam cursos de graduação em computação sem saberem colaborar. Outra maneira de se perceber isto facilmente é examinando a maneira como os primeiros trabalhos são desenvolvidos, em que, ao invés de elaboração de síntese, há uma montagem das soluções parciais individuais, sem quaisquer reflexões do grupo.

Na disciplina introdutória de programação em questão, as aulas foram divididas em práticas e teóricas. As últimas ocorriam uma vez na semana e as práticas eram divididas entre práticas em laboratório e práticas externas. Estas práticas eram utilizadas para trabalho em grupo e, posteriormente, o registro das interações era obrigatório. Foi disponibilizada uma ferramenta de suporte à colaboração, para que fosse armazenado todo o conteúdo criado antes e no decorrer da disciplina, seguindo as diretivas encontradas em [Fuks, H. e Assis, R. 2001] e [Fuks, H., Pimentel, M. e Lucena, C. 2006], cientes dos problemas descritos em [Pimentel, M., Fuks, H. e Lucena, C. 2003]. A Tabela 1 descreve o planejamento das atividades práticas, de acordo com as fases e semanas mostradas na Figura 1. Em seguida, apresentamos alguns exercícios exemplo para cada fase. Estes exercícios foram extraídos do banco de exercícios de programação da UFAM.

1. **Preparação** – "Quatro amigos desejam comprar um produto que custa mais do que eles possuem. A fim de comprarem o produto, os amigos decidem que a quantia que falta seria dividida proporcionalmente à quantidade que cada um tem disponível, considerando que quem contribuiu mais poderia usufruir mais. Sabe-se que o amigo A1 contribuiu com 50% do valor total, A2 com 20%, A3 com 10% e A4 com 20%. Sabe-se, também, que o produto pode custar de 20% até 50% a mais do que a quantia que eles arrecadaram. Descreva em Haskell como você poderia resolver este problema, indicando quanto cada um deve pagar."
2. **Codificação Individual I** – "Considere 2 pontos, p1 e p2, localizados no plano cartesiano, que determinam uma reta. Descreva a equação desta reta."
3. **Codificação Individual II** – "Considere 3 pontos, p1, p2 e p3, localizados no plano cartesiano. Determine se eles constituem um triângulo e, se for o caso, determine sua área."
4. **Codificação em Grupo I** – "Considere 2 pontos, p1 e p2, localizados no plano cartesiano. Estamos interessados em identificar se os seguintes relacionamentos entre eles se aplicam: (a) é possível traçar uma reta passando por p1 e p2 e paralela ao eixo das abscissas; (b) é possível traçar uma reta passando por p1 e p2 e paralela ao eixo das ordenadas; (c) p1 e p2 estão no mesmo quadrante; (d) p1 e p2 estão em quadrantes diferentes; (e) p1 e p2 estão em quadrantes opostos; (f) p1 e p2 estão em quadrantes adjacentes."

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

Tabela 1 - Fluxo de Atividades Práticas

Fases	Semana no Curso	Descrição dos Conteúdos
Individual: **Preparação**	1 e 2	Survey inicial: os estudantes responderam um questionário disponível no ambiente de apoio às
Individual: **Codificação Individual I**	3	Definição dos níveis de habilidade: atividades práticas em laboratório utilizando o AAEP com sessões pré-definidas, usando problemas geométricos básicos (90 minutos) e análise dos resultados baseada no tempo de resolução e qualidade do código.
Grupo: **Codificação Individual II**	5	Resolução de um exercício, com anotações sobre os códigos.
Grupo: elaboração de soluções em grupo usando **Codificação em Grupo I**	6	Análise e seleção de soluções individuais; refinamentos; anotações colaborativas sobre o processo, em forma de chats, fóruns, ou possíveis conversas presenciais.
Grupo: sistematização dos processos de planejamento de solução em grupo usando **Codificação em Grupo II**	8	Problema 1 – os estudantes resolvem problemas de acordo com as fases anteriores. Os membros do
	9 e 10	Problema 2 – uma função ou um conjunto de funções para cada estudante, mas a divisão é
	11, 12 e 13	Problema 3 – os estudantes devem utilizar técnicas de modularização para a divisão de trabalho e decidem sobre quem deve desenvolver que parte, registrando o processo de tomada de decisões.
Time: competição usando desenvolvimento colaborativo (**Codificação em Time**)	14	Estilo de maratona de programação com entrega de prêmio. Ela consiste em: observação externa por tutores e monitores; uso de uma ferramenta para monitorar as atividades em grupo. Nesta etapa, os estudantes também devem registrar o processo de tomada de decisões.
Avaliação do Processo	16	Aplicação de um questionário.

5. **Codificação em Grupo II** – "Em uma clínica, assim que um paciente chega ao hospital, recebe uma senha de atendimento. Há sempre três médicos disponíveis por turno e eles recebem novos pacientes dependendo do número de pacientes que cada um já possui em sua lista de atendimento. O medico que possui menos pacientes em sua lista recebe o próximo. Usando tuplas, podemos definir a seguinte entrada: med_disp(("dr. A", 4, 23), ("dr. B", 1, 13), ("dr. C", 3, 27)), onde o segundo termo de cada tupla se refere ao número de pacientes na lista daquele medico e o terceiro termo se refere ao ultimo paciente atendido por aquele medico. Baseado nesta entrada, escreva um script em Haskell que, para um novo paciente, escolha em qual lista de atendimento ele deve ser alocado."
6. **Codificação em Time** – "Em um banco de sangue há um registro de doações que inclui o número de registro no INSS, sexo (S), idade (I), tipo sangüíneo (TS), fator RH (RH), data de coleta (DC) e a quantidade de sangue (QH) (250 ou 500 ml). O sangue é mantido em recipientes de 250 ml. Os hospitais requerem sangue (H) diariamente. Cada pedido indica as

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

> características do sangue (tipo e fator RH) e total requerido (TR). É sabido que homens e mulheres devem esperar intervalos de tempo mínimo diferentes entre doações. Para os homens são dois meses e para as mulheres, 3. As idades máximas e mínimas para doações (para ambos os sexos) são 60 e 15, respectivamente..."

A modularização de problemas e definição de funções em Haskell são conceitos envolvidos na fase de preparação. As fases de codificação individual requerem o mesmo nível de habilidade (expertise) com mais raciocínio matemático devido à natureza dos problemas geométricos. A fase seguinte, codificação em grupo I, além da expertise necessária para a resolução dos problemas geométricos, requer o uso de condicionais. Na fase de codificação em grupo II, os estudantes devem usar tuplas ou listas para resolverem um problema de complexidade média (para estudantes iniciantes em programação). Finalmente, na fase de codificação em time, os estudantes devem resolver um problema envolvendo tuplas, listas e recursão.

Da primeira fase de codificação individual em diante os estudantes precisam trabalhar em grupo. Esses grupos são formados na terceira semana de aulas práticas, após a aplicação de uma sessão de laboratório supervisionada. O requisito adotado para a formação dos grupos é que cada um seja o mais heterogêneo possível, com um mínimo de 4 integrantes. No máximo 2 devem possuir experiência formal em programação, 1 pode ter alguma experiência (por exemplo, programação em script para web) e 2 ou 3 inexperientes em programação.

Conforme mostrado nos exercícios exemplo acima, após a formação dos grupos, os exercícios aumentam de complexidade seguindo o plano de progressão e o avanço no conteúdo da disciplina. Isto possibilita uma transição gradual de trabalho baseado em práticas individuais à incorporação de práticas colaborativas de desenvolvimento de programas.

6. Conclusão

Esse artigo apresenta a descrição do planejamento de um estudo de caso, envolvendo um plano de progressão de aprendizagem para apoio à programação em grupo, aplicado no primeiro semestre acadêmico de 2008 na UFAM, baseado na análise dos resultados de um estudo piloto conduzido com alunos de graduação em computação na UFAM, conduzido durante o primeiro semestre acadêmico de 2007 [Castro, T. et al 2008].

O semestre acadêmico no qual o estudo de caso supracitado foi aplicado para avaliar o plano de progressão finalizou há pouco tempo e, conseqüentemente, os dados obtidos ainda estão sendo analisados. Mesmo sem ter analisado todo o material digital, é possível identificar uma boa aceitabilidade e rendimento nas sessões de laboratório, que acreditamos tenha motivado os estudantes a cumprirem as atividades planejadas.

Ao analisarmos todo o conteúdo digital produzido durante essa disciplina introdutória, será factível levantar as estratégias colaborativas utilizadas pelos grupos nas codificações. Após esta análise, nós continuaremos com esta investigação com vistas a formalizar um modelo de colaboração para aprendizagem de programação em grupo.

Agradecimentos

Este projeto é parcialmente financiado pelo Ministério de Ciência e Tecnologia através da concessão de despesas do CTInfo nº 550865/2007-1. Hugo Fuks recebe uma bolsa de

pesquisa do CNPq nº 301917/2005-1 e também financiamento da FAPERJ, no projeto "Cientistas do Nosso Estado". Esta pesquisa também recebe recursos destinados ao ColabWeb – Proc.553329/2005-7, CNPq/CT-Amazônia n.27/2005.

Referências

Almeida Neto, F. A. ; Castro, T. ; Castro, A. N. "Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação" (Using the Piagettian Clinic Method for Keeping Track of Programming Learning). In: *Simpósio Brasileiro de Informática na Educação*. Brasília: Gráfica e Editora Positiva Ltda, v. 17. p. 184-193. 2006.

Castro, T. H. C. ; Castro Jr, A. N. ; Menezes, C. S. ; Boeres, M. C. S. ; Rauber, M. C. P. V. "Utilizando Programação Funcional em Disciplinas Introdutórias da Computação" (Using Functional Programming in Computing Introductory Courses). In: *XXII Congresso da Sociedade Brasileira de Computação - X Workshop Sobre Educação em Computação, 2002, Florianópolis-SC. XXII Congresso da SBC*. Porto Alegre : Sociedade Brasileira de Computação, 2002.

Castro, T.H.C; Fuks, H.; Spósito, M.A.F and Castro, A.N. "The Analysis of a Case Study for Group Programming Learning". In the Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies - ICALT. Santander, Spain, 2008.

Chamillard, A. T. and Braun, K. A. "Evaluating Programming Ability in an Introductory Computer Science Course". In: *SIGCSE 3/00*. ACM 1-58113-213-1/00/0003. Austin, Texas, USA, 2000.

Clancy, M.; Titterton, N.; Ryan, C.; Slotta, J. and Linn, M. "New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course". In: *SIGCSE*, ACM 1-58113-648-X/03/0002, 2003.

Dijkstra, E.. "On the Teaching of Programming, i.e. on the Teaching of Thinking". In: *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag NY. 1982.

Eckerdal, A. and Berglund, A. "What Does It Take to Learn Programming Thinking?" In: *ICER'05*, ACM 1-59593-043-4/05/0010. Seattle, Washington, USA, 2005.

Fuks, H. and Assis, R.L. "Facilitating Perception on Virtual Learningware-based Environments". In: The Journal of Systems and Information Technology, Vol 5., No. 1, ISSN 1328-7265, Edith Cowan University, pp 93-113, 2001.

Fuks, H., Pimentel, M. and Lucena, C.J.P. "R-U-Typing-2-Me? Evolving a chat tool to increase understanding in learning activities". In: International Journal of Computer-Supported Collaborative Learning, Volume 1, Issue 1. ISSN: 1556-1607 (Paper) 1556-1615 (Online). Springer: Mar 2006. pp. 117-142.

Krathwohl D. R. "A Revision of Bloom's Taxonomy: An Overview". In: Theory Into Practice, Volume 41, Issue 4. DOI 10.1207/s15430421tip4104_2. Routledge. November 2002 , pages 212 – 218.

Lister, R. and Leaney, J. "Introductory Programming, Criterion-Referencing, and Bloom". In: *SIGCSE*, ACM 158113-648-X/03/0002. Reno, Nevada, USA, 2003.

Publicado nos Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 de Outubro, Vila Velha, ES..
ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS. pp 45-54.

Mendonça, A. P. ; Castro, A. N. ; Mota, E.S. ; Silva, L. S. ; Pereira, V. L. S. "Uma Experiência com o uso de Mapas Conceituais para Apoiar o Método da Controvérsia Acadêmica" (An experience Using Conceptual Maps to Support the Academic Controversy Method). In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola, 2002, Florianópolis-SC. XXII Congresso da SBC*. Porto Alegre : Sociedade Brasileira de Computação, 2002. v. 5. p. 99-107.

Pereira, V. L. S. ; Castro, A. N. ; Mendonça, A. P. ; Silva, L. S. "Análise do método Jigsaw de aprendizagem cooperativa através da utilização de mapas conceituais" (The Analysis of the Jigsaw Cooperative Method for Learning using Concept Maps). In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola, 2002, Florianópolis-SC. XXII Congresso da SBC*. Porto Alegre : Sociedade Brasileira de Computação, 2002. v. 5. p. 181-188.

Pimentel, M., Fuks, H. and Lucena, C.J.P. "Co-text Loss in Textual Chat Tools". In: 4th International and Interdisciplinary Conference on Modeling and Using Context - CONTEXT 2003, LNAI 2680, Stanford, CA, USA, June, pp 483-490, 2003.

Sharan, S. "Handbook of Cooperative Learning Methods". *The Greenwood educators' reference collection*, ISSN 1056-2192. Praeger Publishers. ISBN 0-313-28352-4. USA. 1999.

Silva, L. S; Castro, A. N. ; Mendonça, A. P. ; Pereira, V. L. S. "Mapas Conceituais como suporte à estratégia de Investigação em Grupo: Uma experiência na Universidade" (Concept Maps as a Support for the Group Investigation Strategy: an Experience at the University). In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola, 2002, Florianópolis-SC. XXII Congresso da SBC*. Porto Alegre : Sociedade Brasileira de Computação, 2002. v. 5. p. 163-172.

Weinberg, G. M. "The Psychology of Computer Programming". *Computer Science Series*. Litton Educational Publishing, F9264-000-4. USA. 1971.