Sistematização da Aprendizagem de Programação em Grupo

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

[1]Programa de Pós-Graduação em Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Gal. Rodrigo Otávio, 3000 – Campus, Setor Norte – Manaus – AM - Brasil

hugo@inf.puc-rio.br, thais@dcc.ufam.edu.br

Abstract. *The thesis reported here deals with devising structuring elements that may broaden intervention opportunities from the teacher in a context of group programming learning. Based on a set of case studies with freshmen in computing courses a systematization for practices, methods and technologies was developed producing an approach for supporting group programming based in three investigation paths: pedagogical assumptions, CSCL environments and collaboration methods. Within this context we proposed a group programming progressive learning scheme and a new way to analyze conversation logs inspired on the speech acts theory.*

Resumo. *A tese aqui relatada trata da concepção de elementos estruturantes para ampliar as oportunidades de intervenção pelo professor em um contexto de aprendizagem de programação em grupo. A partir de uma série de estudos de caso com turmas de calouros em cursos de computação, foi desenvolvida a sistematização de práticas, metodologias e tecnologias para apoiar a aprendizagem de programação em grupo, baseada em três frentes de investigação, a saber: pressupostos pedagógicos, ferramentas LMS e métodos de colaboração. Nesse contexto foi proposto um esquema progressivo de aprendizagem de programação em grupo e uma maneira de analisar logs de conversas inspirada na teoria de atos de fala.*

1. Introdução

Aprendizagem de programação é uma atividade cognitiva que requer alto nível de raciocínio abstrato [Weinberg, 1971]. Essa necessidade de abstração frequentemente provoca bloqueios nos alunos de graduação em computação durante a disciplina introdutória de programação, transformando a aprendizagem no tema em uma experiência penosa e desmotivadora. Para amenizar o problema, diferentes abordagens têm sido utilizadas e um elemento comum a todas elas é a busca pelo desenvolvimento de habilidades para a solução de problemas, o que induz o aluno a intuitivamente entender estratégias como divisão e conquista, além de ilustrar a representação de uma solução na linguagem de programação adotada na disciplina.

Abordagens mais recentes procuram aplicar conceitos e técnicas utilizados em aprendizagem colaborativa à aprendizagem de programação. Nesse caso, a principal motivação para incorporar a colaboração na aprendizagem é que o mercado de trabalho exige profissionais aptos a trabalharem em equipes e essa habilidade é formada durante os cursos de graduação. A construção dessa habilidade exige um planejamento mais profundo das disciplinas introdutórias. No plano de curso deve estar prevista a utilização de métodos de aprendizagem colaborativa além das estratégias de aprendizagem e práticas pedagógicas usuais. O pressuposto desses métodos colaborativos é de que os alunos aprendem mais quando conseguem interagir com seus pares, em busca de uma solução para um dado problema [Delval, 2002].

Em disciplinas introdutórias de programação em nível de graduação, o uso de *groupware* baseado na Internet representa uma oportunidade para introduzir boas práticas no processo de aprendizagem dos alunos, especialmente o registro de todas as interações entre os alunos em seus grupos enquanto resolvem exercícios, além do registro da evolução dos códigos de cada aluno. O registro das atividades é um requisito essencial para acompanhar o progresso da aprendizagem, tornando possível para o professor fornecer *feedback* apropriado ao longo do processo, especialmente na realização dos exercícios.

Os alunos que trabalham em grupos apoiados por LMSs (Sistemas de Gerenciamento da Aprendizagem) geralmente produzem extensos *logs* de conversas incluindo fragmentos de código, rastros dos processos de tomada de decisão, da metodologia de trabalho e do uso de seus próprios padrões de desenvolvimento, os quais devem ser filtrados e categorizados para futuras análises dando subsídios para a identificação "*just in time*" de dificuldades dos estudantes e à oportuna intervenção do professor durante o processo.

A investigação aqui relatada trata da concepção de elementos estruturantes para ampliar as oportunidades de intervenção pelo professor em um contexto de aprendizagem de programação em grupo. Para isso, investigamos tecnologias de apoio à aprendizagem de programação, enfatizando técnicas de programação em grupo, visando a inserção da colaboração em contextos de aprendizagem de programação.

Em geral, nos cursos de programação introdutória, as turmas são grandes e o professor, mesmo registrando todas as interações, não consegue, sem auxílio de mecanismos de filtragem e classificação de conteúdos, acompanhar o que se passa em cada grupo para poder intervir durante a realização dos exercícios. Daí surge a questão de pesquisa: como ampliar oportunidades de intervenção em um processo de aprendizagem de programação em grupo?

O objetivo desta tese é sistematizar práticas, metodologias e tecnologias em uma abordagem para apoiar a aprendizagem de programação em grupo. Para isso, três frentes de investigação devem ser abertas – no Pressuposto Pedagógico, nas Ferramentas LMS e no Método de Colaboração.

O pressuposto pedagógico da tese é a teoria de desenvolvimento cognitivo de Piaget, aliada a práticas vigentes de ensino de programação e técnicas conhecidas de programação em grupo. As práticas pedagógicas relacionadas na revisão bibliográfica desta tese se baseiam em concepções pedagógicas diversas, tendo em comum o uso de

software de apoio para a realização de exercícios. Essas ferramentas computacionais são úteis para monitorar e intervir durante o processo de aprendizagem. As técnicas de programação em grupo investigadas são usadas em empresas de desenvolvimento de software e muitas vezes adaptadas para serem utilizadas em disciplinas introdutórias de programação em cursos de graduação cujo cerne é computação.

Segundo a investigação feita nesta tese, os LMSs tem sido bastante utilizados para apoiar a aprendizagem presencial na graduação, confirmado por pesquisas em máquinas de busca na web e exemplo prático na instituição onde os estudos de caso desta tese foram aplicados, a Universidade Federal do Amazonas (UFAM). Ainda de acordo com a teoria de Piaget, esta pesquisa assume que os ambientes LMSs aliados a técnicas colaborativas (o que chamamos ambientes CSCL) incentivam a colaboração e regulam as práticas desejadas. Portanto, o segundo pilar desta tese é baseado em ambientes CSCL e tecnologias que apoiem a adoção de práticas desejadas com possibilidades de integração a esses ambientes. Para isso, foram investigadas linguagens de agentes e adoção do LCC para descrever agentes responsáveis pela identificação de padrões de interação gerados a partir de um esquema proposto nesta tese.

A intervenção é vista neste trabalho como o ponto crucial na aprendizagem de programação em grupo. Neste sentido, foi proposto um esquema progressivo de aprendizagem de programação, que auxilia os alunos a gradativamente adotarem práticas colaborativas na resolução de exercícios de programação enquanto o professor observa como eles estão incorporando práticas colaborativas às práticas individuais anteriormente identificadas através de uma análise baseada na evolução dos códigos individuais dos alunos, desenvolvida no contexto desta pesquisa. Esse esquema pode, conforme verificado com uma linguagem de representação do conhecimento, ser generalizado para outros contextos de aprendizagem.

2. Aprendizagem de Programação

O que realmente é necessário para se aprender e facilitar a aprendizagem de programação ainda não é conhecido. Até o desfecho desta pesquisa não foram encontrados trabalhos na literatura que tenham tido êxito em estabelecer métodos irrefutáveis e técnicas para aprendizagem de programação. Por outro lado, há iniciativas cujo principal objetivo é criar e manter o interesse dos alunos na disciplina utilizando abordagens inicialmente baseadas nos processos de resolução de problemas que os alunos iniciantes em cursos de graduação em computação já foram expostos durante a educação básica, necessárias à apreensão dos conceitos. Outros ainda valorizam as práticas colaborativas, como a técnica de programação em pares, parte da metodologia dos métodos ágeis de programação, como meio de envolver os alunos em projetos de interesse coletivo, proporcionando uma vivência em grupo, necessária no mercado de trabalho.

Buscar compreender quais processos cognitivos estão envolvidos na aquisição das habilidades para programação possibilita que as práticas utilizadas por cada aluno se tornem evidentes e o conhecimento gerado possa ser reutilizado em outros cursos de programação introdutória. Nesta seção apresentamos uma caracterização de diferentes maneiras de agrupar as modificações observadas na evolução dos códigos de alunos de

Ciência da Computação e Engenharia da Computação cursando a disciplina de Introdução à Computação na UFAM.

As versões de códigos, objeto desta análise, foram obtidas diretamente de um banco de dados local, que foi povoado durante a realização de trabalhos práticos da disciplina introdutória, conforme planejamento e execução discutida em [Almeida, Castro & Castro, 2006]. Essa disciplina introdutória foi ministrada utilizando o paradigma de linguagem funcional sendo toda a codificação em Haskell.

Para identificar alguns processos cognitivos que os alunos poderiam demonstrar através das várias versões parciais de sua codificação, propusemos uma ferramenta chamada AcKnow, discutida amplamente em [Castro, Castro and Fuks, 2008], que identifica e agrupa as modificações na evolução dos códigos em três categorias de evolução de códigos: sintáticas, semânticas e refactoring.

- **Modificações sintáticas** – exemplos: endentação, inserção e deleção de caracteres. Essas modificações visam tornar o código interpretado corretamente pelo interpretador Haskell, processo que sugere muitas correções por tentativa e erro na tentativa de encontrar uma solução.
- **Modificações semânticas** – exemplos: modificação nas estruturas de dados; mudança de tupla para lista; inclusão de uma função recursiva; e correção de bugs. Essas modificações afetam diretamente a avaliação da função, resultando em uma saída errada.
- **Refactoring** – exemplo: modificações cujo objetivo é melhorar o código, de acordo com métricas de qualidade conhecidas da engenharia de software.

O Acknow foi desenvolvido em Prolog, utilizando uma gramática de cláusulas definidas e um conjunto de regras de inferência. O objetivo dessa ferramenta é analisar e categorizar a evolução dos códigos dos alunos, fornecendo elementos para que o professor faça intervenções no processo de aprendizagem de programação de seus alunos enquanto eles ainda estão elaborando sua solução. Como entrada de dados, o AcKnow foi projetado para receber funções presentes nos códigos dos alunos indicados por uma análise quantitativa previamente realizada por uma ferramenta de controle de versões chamada AAEP [Almeida, Castro & Castro, 2006]. Essas indicações são baseadas na quantidade de versões que cada aluno fez. Quando a essa quantidade de versões de um ou mais alunos difere significativamente da distribuição normal da turma, esse(s) aluno(s) é (são) identificado(s) como caso especial, ficando marcado para visualização do professor, que envia as versões diretamente para o AcKnow. Então, baseado na análise do AcKnow para cada par de versões, o professor faz uma análise mais detalhada, fornecendo feedback diretamente aos alunos.

O padrão de modificações mais comum dentre os códigos analisados é um grande número de modificações sintáticas, algumas modificações semânticas intercaladas com outras sintáticas, seguindo em alguns casos por um refactoring. Uma aluna chamou atenção por apresentar um padrão diferente e bem eficaz. Ela efetuou uma modificação de refactoring logo após as de sintática e então refletiu e percebeu que a estrutura de dados utilizada poderia trocada em prol de uma representação mais adequada.

3. Tecnologias para Aprendizagem de Programação em Grupo

No contexto de um projeto de pesquisa da UFAM, foi desenvolvido o ColabWeb, um ambiente CSCL configurado e desenvolvido utilizando a plataforma Moodle. Este ambiente é utilizado como apoio às disciplinas presenciais ministradas por professores do Departamento de Ciência da Computação. O projeto de interface para o ColabWeb privilegiou sua funcionalidade, o que fez com que os primeiros cursos ocorressem com um aproveitamento desequilibrado de seus recursos. Em vista dessa diferença em termos de sucesso no uso daquele groupware, evidenciada neste trabalho através do gerenciamento de cursos de computação introdutória, decidiu-se fazer uma inspeção que evidenciasse o poder de comunicação do artefato de software (ColabWeb). Para isso, é utilizado o Método de Inspeção Semiótica (MIS) [De Souza *et al.*, 2008]. Vale ressaltar que um método de avaliação da comunicabilidade do ambiente CSCL que se estiver utilizando é parte integrante desta sistematização, pois tal ambiente precisa passar por constantes melhorias.

Outros métodos de avaliação aplicados a ambientes CSCL já foram utilizados em contextos de curso, como os que avaliam a participação dos alunos em fóruns, a participação dos alunos de um curso em todas as atividades do groupware utilizado e até a participação em debates. No entanto, nenhum desses trabalhos considera que a interface influencia na avaliação da participação dos alunos nas atividades propostas em um ambiente CSCL. Tal aspecto só é considerado em métodos de inspeção baseados na semiótica como é o caso do MIS.

Conforme a expectativa do MIS, foram elencadas recomendações de melhorias para a comunicabilidade do ColabWeb. Essa inspeção foi descrita detalhadamente em [Castro & Fuks, 2009]. Avalia-se uma instância (ColabWeb) de uma arquitetura amplamente utilizada (Moodle), observando-se unidades, que neste caso são cursos. A partir dessas unidades generalizam-se as observações para a instância.

3.1. Resultado da Aplicação do MIS no ColabWeb

Foram encontrado problemas navegacionais, ou seja, a forma de navegação e estruturação de páginas web do ColabWeb, que é uma herança do Moodle, não é adequada.

Apesar dos diversos problemas encontrados e a sugestão de modificações, a estrutura de navegação do Moodle não permite ser modificada. No entanto, da mesma forma que as configurações, podem-se omitir os links indesejados da interface com o usuário, acrescentando outro, referenciando assim o desejado. Dessa forma, o problema que ocorre com a visualização de grupos poderia ser amenizado retirando-se o link de grupos, permanecendo somente a palavra que os descreve.

4. Um Esquema Progressivo para Aprendizagem de Programação em Grupo

Um recorte na literatura relacionada à aprendizagem de programação em grupo incluindo: o relato de uma experiência de 15 anos com aprendizagem de programação [Castro *et al*, 2002]; a especificação de padrões de colaboração [Kobbe *et al*, 2007]; e um estudo exploratório sobre aprendizagem colaborativa de programação [Castro *et al*, 2008], mostra que a programação em grupo é uma tarefa difícil em grande parte devido à inexperiência dos estudantes com o trabalho em grupo. Para desenvolver atividades

em grupo, especialmente aquelas como a programação, é necessário um modelo para orientar a atividade ou, no caso da inexistência de tal modelo, um "esquema de progressão" para a aprendizagem de programação, como proposto a seguir.

A Figura 2 ilustra o esquema progressivo de aprendizagem de programação que define uma progressão do indivíduo para o grupo na programação, em um cenário que inicia na Fase 1, com uma preparação que envolve sessões no laboratório tratando com problemas introdutórios e esclarecimentos sobre a metodologia. A Fase 2 consiste da solução individual com o respectivo registro. Na Fase 3 o trabalho em grupo começa com a decisão sobre qual seria a melhor solução individual desenvolvida. Na Fase 4 a solução do problema torna-se colaborativa: o professor define as tarefas e o grupo define os atores para cada atividade. Na Fase 5 o grupo é também responsável pela definição das tarefas. Por fim, na Fase 6 acontece uma atividade de desenvolvimento onde os grupos competem num cenário que reproduz situações reais de trabalho.

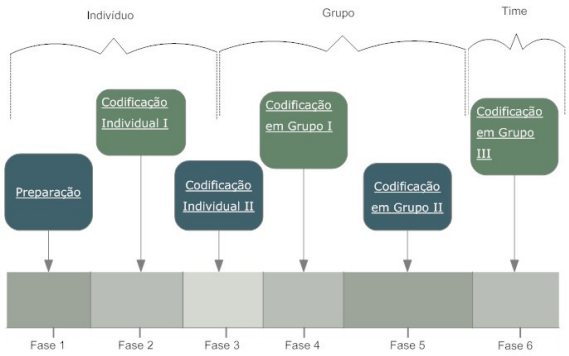

Figura 1 – Esquema Progressivo de Aprendizagem de Programação em Grupo

A opção pela divisão em 6 fases, bem como a sequenciação proposta, foi baseada na experiência com o ensino de programação usando linguagens funcionais desenvolvido naquela instituição desde 1989, bem como na análise dos relatos citados no início desta seção, considerando-se um curso de 75 horas desenvolvido ao longo de um semestre acadêmico.

5. Sistematização da Aprendizagem de Programação em Grupo

Até agora propusemos elementos de uma abordagem para aprendizagem de programação em grupo baseada em: (1) elementos da evolução de códigos para evidenciar pistas de aquisição de níveis mais elevados de abstração a partir da identificação de fragmentos de código que representam modificações na forma como o aluno "enxerga" a solução do exercício; (2) elaboração de exercícios de programação adequados ao desenvolvimento da habilidade de abstração de alunos iniciantes em cursos de graduação em computação, proporcionando a prática em grupo como sugerido nos estudos de Piaget; (3) recomendações para utilização, configuração e desenvolvimento de interfaces para ambientes CSCL de forma a proporcionar maior poder de comunicabilidade entre os alunos; e (4) um esquema progressivo de introdução de práticas colaborativas no contexto de situações de aprendizagem de programação em grupo.

Para que os elementos enumerados acima se transformem na sistematização de uma abordagem é necessário evidenciar se, com a utilização de um ambiente CSCL (3) para apoiar a disciplina de programação introdutória, a elaboração de exercícios de programação que proporcionem a evolução do raciocínio abstrato (2), seguindo um macro-script para colaboração (4) e considerando que o aluno modifique seu raciocínio através do refinamento sucessivo dos códigos (1), as oportunidades de intervenção são ampliadas.

As intervenções têm o propósito de, através do acompanhamento durante a resolução dos exercícios, incentivarem as interações no ambiente. Essas atividades requerem a identificação e análise dos padrões presentes nas interações, que constituem o elemento final da sistematização proposta nesta tese.

Através de dois estudos de caso aplicamos o esquema progressivo e ao mesmo tempo propomos uma forma de análise das conversas. O esquema progressivo de aprendizagem de programação em grupo proposto precisava ser testado de modo a provar sua utilidade em cursos introdutórios de programação como forma de desenvolver nos alunos habilidades para colaboração em programação, apoiando o refinamento de sua habilidade de abstração.

Primeiramente, propusemos um estudo de caso para aplicarmos os elementos dessa abordagem, em forma de um estudo de caso descritivo [Yin, 2010], onde procuramos por evidências de como os grupos utilizam o esquema proposto, dentro da metodologia do curso. A partir da análise desse estudo de caso, identificamos padrões nas interações nas discussões registradas nos fóruns que nos permitiram caracterizar através do estilo de conversa, se o grupo estava interagindo de forma a favorecer a colaboração.

Um segundo estudo de caso foi então definido, sendo o seu projeto uma replicação do primeiro, com a diferença de que nesse momento procuramos explicações para confirmar uma hipótese, o que caracteriza um estudo de caso explanatório (Yin, 2010). Foram utilizadas as categorias encontradas anteriormente, as quais foram chamadas de padrões de interação.

5.1 Padrões de Interação nos logs das Conversas do ColabWeb

Ao procedermos a análise dos logs, observamos que os turnos de fala, apesar das diferenças de vocabulário e estilo, apresentavam intenções que se repetiam em todos os grupos. Dada a natureza dos diálogos, notamos uma semelhança com Atos de Fala, que, conforme descrito por Searle [1969], são classificações para os diferentes usos da linguagem, principalmente sobre a interpretação de questões, exclamações, comandos, ou seja, sobre enunciados que não são unicamente descritivos. Percebemos que são descritos para cada fala uma classificação que retrata a intenção do indivíduo, o que sugeriu sua adequação à situação observada na análise. A teoria dos atos de fala é utilizada em ambientes multiagente para descrever as interações dos agentes. Para adaptá-la ao contexto de fóruns de discussão voltados à aprendizagem de programação, achamos necessário estender o conceito do ato de fala, de forma que retratasse a continuação desses atos, em forma de resposta. Dessa forma podemos perceber se o grupo está conversando (alternando turnos, comentando falas anteriores) ou se está somente cumprindo o roteiro do esquema progressivo.

Chamamos essas classificações inspiradas em atos de fala estendidos de **padrões de interação**, devido a sua utilização em desenvolvimento de ambientes CSCL, possuindo categorias de descrição para cada padrão, o que fornece mais subsídios para verificação de consistência de cada padrão.

5.2 A Representação dos Padrões de Interação

Estereótipos são combinações de diferentes padrões de interação. Estereótipos positivos são aqueles que normalmente conduzem a um bem-sucedido esforço colaborativo de codificação, enquanto os estereótipos negativos são aqueles que não indicam esforço colaborativo, eventualmente expressando o código de um único membro ou um código incorreto.

Entretanto, estereótipos não emergem facilmente nos logs de conversas. Duas razões principais para isso são a ocorrência de quebras na conversação que não são adequadamente restauradas numa discussão on-line e a complexidade do diálogo natural. Assim, uma representação estruturada para a discussão poderia ser benéfica. Uma forma de tratar o problema de definir, casar e tratar com estereótipos detectados dentro dos logs de diálogos baseados no fórum é utilizar uma linguagem de representação formal com boa dose de expressividade.

Uma linguagem baseada na lógica pra tratar com protocolos de comunicação em ambientes multiagente foi então adotada para representar os padrões de interação. O LCC (*Lightweight Coordinate Calculus*) [Robertson, 2004] é uma notação já utilizada em diferentes aplicações sob uma plataforma denominada *Open Knowledge* (www.openk.org) que ajuda a tratar o problema de coordenação. Apesar de ser uma linguagem bastante expressiva, o LCC mantém a simplicidade necessária para representar adequadamente as interações através de recursos de inferência providos pela plataforma Open Knowledge.

Tipicamente, Figura 2, cada processo inicia por um membro do grupo decidindo iniciar um tópico com uma mensagem, disparando desse modo um padrão de interação. O iniciador automaticamente torna-se o coordenador para aquele padrão de interação. Em seguida um agente virtual chamado "*broadcaster*" distribui a mensagem para todos os demais inscritos na conversa. Caso após a leitura da mensagem alguém resolver respondê-la, a pessoa torna-se automaticamente um avaliador enviando a mensagem de volta ao coordenador via *broadcaster*.

O professor analisa os padrões de interação apresentados por cada grupo e define qual sequência ele considera bem sucedida como um estereótipo. Uma base de conhecimento é então criada para cada exercício compreendendo o log de conversas do grupo, padrões de interação e estereótipos. Sempre que um estereótipo não identificado surgir durante uma conversa no fórum, o professor deve analisar em profundidade e dar *feedback* ao grupo antes que a tarefa seja concluída. Ele pode precisar incluir novos estereótipos na base de conhecimento.

Para saber quando intervir, os professores deveriam sempre: *(a)* identificar os estudantes que não estão correspondendo à participação esperada nos próprios grupos, o que implica que os alunos encontraram dificuldades em executar a tarefa solicitada; *(b)*

identificar os estereótipos negativos, buscando ajudar os alunos sempre que eles estiverem “bloqueados”.

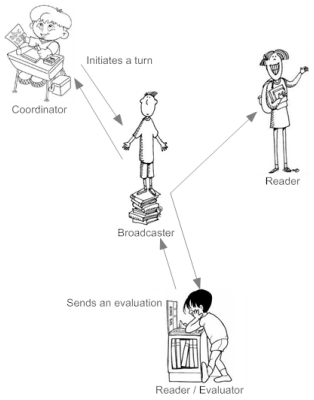

Figura 2 – Representação Formal das Conversas

As pistas para intervenção emergem sempre que: *(i)* um padrão de interação aparece repetitivamente numa discussão no fórum; *(ii)* somente um ou dois membros do grupo se mantêm trabalhando, mesmo se estiverem usando diferentes padrões de interação e *(iii)* a combinação de padrões de interação reforçar estereótipos negativos.

6. Conclusão

Este artigo descreve resumidamente a pesquisa de uma tese de doutorado que propõe a sistematização da aprendizagem de programação em grupo. Essa sistematização é fundamentada na teoria do desenvolvimento cognitivo de Jean Piaget, na utilização de ambientes CSCL para auxiliar no registro e monitoramento das atividades de aprendizagem guiadas por um esquema progressivo de aprendizagem de programação em grupo.

A partir da teoria de Piaget fazemos uma comparação com o desenvolvimento da capacidade de abstração nos adolescentes, perfil da maioria dos alunos ingressantes nos cursos de Engenharia da Computação e Ciência da Computação da UFAM, instituição onde desenvolvemos os estudos de caso desta pesquisa. O AcKnow foi desenvolvido buscando explicitar o processo cognitivo dos alunos ao resolverem problemas de programação.

Sabemos que o grupo tem um papel importante na aprendizagem para descobrirmos como os alunos se organizam em grupos e levantarmos as necessidades desenvolvemos um estudo de caso exploratório aplicado a uma turma de calouros de Ciência da Computação em 2007. Esse estudo apontou a falta de organização e de critérios de divisão de tarefas no grupo como uma das causas da dificuldade dos grupos realizarem o exercício no tempo previsto. Isso serviu de subsídio para a elaboração de um esquema progressivo de aprendizagem de programação em grupo.

O esquema progressivo de aprendizagem de programação em grupo foi proposto e posto em prática, mostrando-se eficaz para a introdução de práticas de colaboração à programação porque estimula os participantes a refletir sobre o processo de solução de problemas como um grupo, percebendo a necessidade de interdependência entre os envolvidos.

Para a análise do estudo de caso de 2008 utilizamos a forma de categorizar discurso dos atos de fala [Searle, 1969] e criamos padrões de interação, onde a unidade de análise é uma fala, que pode estar em duas ou mais entradas desde que faça parte do mesmo pensamento, e suas respostas ou continuações. Discutimos com o professor da turma de 2008 sobre os momentos em que ele interveio e, vendo a análise, os momentos que gostaria de intervir se estivesse ciente dos padrões de interação durante a resolução dos exercícios. A partir daí elaboramos estereótipos positivos e negativos, os bons sendo aqueles que promoviam colaboração e os maus os que a atrapalhavam ou não a incentivavam. Representamos formalmente os padrões de interação e executamos com as informações do 2º. exercício da fase 5. Os padrões provaram-se consistentes. No estudo de caso seguinte, alguns estereótipos foram refinados e outro foi incluído.

Referências

Almeida Neto, F. A. ; Castro, T. ; Castro, A. N. Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação. In: XVII Simpósio Brasileiro de Informática na Educação, 2006. Brasília: Gráfica e Editora Positiva Ltda, v. 17. p. 184-193. 2006.

Castro, T.; Fuks, H.; Castro, A. N. Detecting Code Evolution in Programming Learning. In: 19th Brazilian Symposium on Artificial Intelligence, 2008, Salvador, Bahia. Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence, v. 5249. p. 145-156. 2008b.

Castro, T.; Fuks, H. Inspeção Semiótica do ColabWeb: Proposta de Adaptações para o Contexto de Aprendizagem de Programação . Revista Brasileira de Informática na Educação. Vol.17, N. 1. Pp 71-81. ISSN 1414-5685. 2009.

Castro, T.; Fuks, H.; Spósito, M.; Castro, A. The Analysis of a Case Study for Group Programming Learning. ICALT - Proc. of the 8th IEEE International Conference on Advanced Learning Technologies, July 1-5, Santander, Spain. 2008.

Delval, J. Introdução à Prática do Método Clínico: descobrindo o pensamento das crianças. Editora ARTMED. Porto Alegre, Brasil. 2002.

De Souza, C. S. Compulsory Institucionalization: investigating the paradox of computer supported informal social process. Interacting with Computers, v. 16, n. 4, p. 635-656. 2004.

Kobbe, L., Weinberger, A., Dillenbourgh, P., Harrer, A., Hämäläinen, R., Häkkinen, P. And Fischer, F. Specifying computer-supporter collaboration scripts. Computer-Supported Collaborative Learning, 2:211-214. 2007.

Robertson, D. A Lightweight Method for Coordination of Agent Oriented Web Services. In Proceedings of AAAI Spring Symposium on Semantic Web Services. Stanford. 2004.

Searle, J. Speech Acts. Cambridge University Press. ISBN 0-521-09626-X. 1969.

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

Yin, Robert K. Estudo de Caso: Planejamento e Métodos 4ª. Edição. Bookman Editora. ISBN 8577806553. 2010.