Integração de Ferramentas para Acompanhamento da Aprendizagem de Programação

Thais Helena Chaves de Castro[1,2], Hugo Fuks[1], Alberto Nogueira de Castro Júnior[2], Marcos André Spósito[2]

[1]Departamento de Informática (PUC-Rio)
Av. Mq. de São Vicente, 225 RDC – Gávea - 22453-900 Rio de Janeiro – RJ – Brasil

[2]Departamento de Ciência da Computação (UFAM)
Av. Gal. Rodrigo Otávio Jordão Ramos, 3000 – 69077-300 – Manaus – AM - Brasil

{thais,hugo}@les.inf.puc-rio.br,{albertoc,mafsposito}@dcc.ufam.edu.br

Abstract. *This paper describes a research in collaborative learning of programming, aiming at intervening in first undergraduate course's methodology in order to address the specific problems concerning programming learning activities, providing the correspondent computational support. It is presented a bibliographical survey about the state of the art in collaborative methods and techniques used in programming learning. From this review it is described the planning and implementation of a case study, which has used the AAEP (a tool to support programming) along with the AulaNet (a tool to support collaboration).*

Resumo. *Este artigo descreve uma investigação sobre aprendizagem colaborativa de programação, de forma a interferir na metodologia adotada em cursos introdutórios de computação, de modo a atender às demandas específicas do domínio da aprendizagem de programação, fornecendo suporte computacional adequado. É apresentada uma revisão bibliográfica sobre o estado da arte em métodos e técnicas colaborativos utilizados em cursos introdutórios de computação. A partir desta revisão, é relatado o planejamento e a execução de um estudo de caso, que utilizou o AAEP (ferramenta de suporte à programação) em conjunto com o AulaNet (ferramenta colaborativa).*

1. Introdução

O trabalho em grupo tem se mostrado uma necessidade tanto no mercado de trabalho quanto no ensino. Do ponto de vista empresarial, a crescente demanda por produtos e serviços frequentemente conduz à competição exacerbada, o que tem levado as empresas a buscarem convergência em suas atividades, de forma que não haja duplicação de esforços para uma mesma tarefa, priorizando assim a colaboração entre as diversas equipes. Já sob a ótica do ensino, ao colaborarem os alunos podem enxergar o ponto de vista de seus pares e construir uma solução em conjunto para o problema fornecido. O efeito da colaboração é facilmente percebido quando se acompanha um curso em um ambiente virtual e pode-se analisar uma diferença substancial nos refinamentos das soluções (bem maiores nas atividades desenvolvidas em grupo), conforme descrito em [Almeida et al, 2006].

Na área de aprendizagem de programação em grupo, é necessária a adoção de um esquema para guiar a criação de estratégias e o acompanhamento das atividades dos grupos. Este modelo precisa ser adequado tanto para a atividade meio (colaboração) quanto para a atividade fim (solução de problemas). Embora existam modelos para solução de problemas [Polya, 1976] e [Tommarello e Deek, 2002], e outros para colaboração [McGrath e Hollingshead, 1993], [Goodhue e Thompson, 1995], [Jennings, 1993] e [Weiseth et al, 2006], na literatura técnica da área não foram encontrados relatos sobre modelos específicos para aprendizagem de programação em grupo.

Este artigo tem o objetivo de apresentar um suporte computacional aliado a uma metodologia para aprendizagem de programação. Na Seção 2 a investigação corrente é situada no esforço multi-institucional de pesquisa no ensino e aprendizagem de programação descrito em [Castro *et al*, 2002],[Castro *et al*, 2005] e [Almeida *et al*, 2006]. Na Seção 3 será apresentada uma proposta de suporte computacional para aprendizagem de programação baseado em resultados de projetos descritos na Seção anterior, que culminou na elaboração de um esquema de imersão em atividades que demandam colaboração, o que é evidenciado na aplicação do estudo de caso descrito na Seção 4.

2. Aprendizagem de Programação em Grupo

A comunidade acadêmica, ao longo da existência da área da computação, tem buscado entender porque programar é uma tarefa difícil e como a metodologia adotada em cursos para iniciantes poderia influenciar a maneira como esses alunos aprendem a programar. Neste sentido, há diversos relatos na literatura técnica da área, como fundamentos básicos descritos em [Djkistra, 1976] e relatos como o encontrado em [Brooks, 1975] sobre a formação de equipes em projetos de software. Nesta seção, discutiremos algumas experiências mais atuais, baseadas nos princípios de programação encontrados na literatura básica da área de computação.

No trabalho descrito em [Clancy et al, 2003] é relatado um esforço de desenvolver um novo curso de programação baseado em sessões de laboratório. Neste curso foram planejadas atividades diversificadas como discussões online, exercícios de programação em dupla, leitura de textos disponibilizados na web, anotações de reflexão, entradas em um diário e colaborações utilizando o processo de revisão por pares para criticar a resposta dos colegas a um mesmo tópico. Trata, portanto, da transformação de uma metodologia com aulas teóricas e práticas para outra que utiliza somente aulas práticas, com atividades distintas e bem distribuídas nas sessões.

Seguindo a linha de avaliação da aprendizagem, o trabalho descrito em [Chamillard and Braun, 2000] descreve uma combinação entre algumas técnicas de avaliação em um curso introdutório de computação e demonstra através de análises estatísticas, as diferenças e relacionamentos entre essas técnicas. O curso foi planejado atendendo à concepção que antes de aprenderem a programar os alunos devem ser hábeis em resolver problemas. Portanto, primeiramente, os alunos resolvem problemas sem o uso de uma linguagem de programação e posteriormente aprendem a usar uma linguagem de programação para representar soluções, codificando em ADA. A partir daí, o curso prossegue com seis sessões de laboratório, onde os alunos devem resolver problemas individualmente, podendo consultar seus pares sempre que necessário. A

etapa seguinte requer a divisão da turma em pequenos grupos (dois a quatro integrantes) para desenvolver um programa mais sofisticado, em geral um jogo.

Ainda utilizando modelos de avaliação para ensinar programação, o trabalho descrito em [Lister and Leaney 2005] utiliza uma pré-avaliação dos alunos como base para categorizá-los em estágios de aprendizagem definidos na taxionomia de Bloom. A partir daí, o curso é formatado de modo que haja atividades diferenciadas para os alunos que se encontram nos diferentes estágios. Essa abordagem, ainda segundo descrito no artigo, possibilita a união entre diferentes técnicas de avaliação, como exercícios em laboratório, provas de múltipla escolha, tarefas, projetos e revisão por pares, todas desenvolvidas em torno de uma filosofia de avaliação baseada na taxonomia de Bloom.

Em um esforço para conhecer aspectos que possam facilitar a aprendizagem de programação, o trabalho descrito em [Eckerdal and Berglund 2005] afirma que através de respostas dos alunos a perguntas do tipo "o que é programação?" é possível se definir a ordem de apresentação dos paradigmas de programação. Naquele artigo os autores partem do princípio que os alunos precisam saber o que é aprender a programar para que realmente aprendam. A maioria das respostas aferidas sugere que os alunos devem primeiramente ser expostos a um raciocínio mais estruturado para depois ingressarem em abstrações de objetos.

Baseados em nossa experiência de cerca de 15 anos com ensino de programação, o que realmente é necessário utilizar para facilitar a aprendizagem de programação ainda é uma questão em aberto. Embora os artigos citados acima façam tentativas de encontrar uma solução, ainda não se tem notícia de trabalhos que estabeleçam métodos e técnicas incontestáveis. Por outro lado, há iniciativas cujo foco é despertar e manter o interesse dos alunos no curso, utilizando conceitos de *extreme programming* [Beck, 1999], já amplamente utilizados por times de desenvolvimentos na indústria de software.

Nessa outra vertente, os relatos descritos em [McDowell et al 2002] e [Mendes,Al-Fakhri and Luxton-Reilly 2005] descrevem a utilização de uma técnica de *extreme programming*, chamada de *pair programming*, ou programação aos pares, para desenvolvimento de programas em um contexto de aprendizagem. O pressuposto era de que a técnica de programação aos pares resultaria em uma melhora significativa no desempenho dos alunos. Isto não foi confirmado por nenhum dos relatos citados. Porém, em [McDowell et al 2002] foi verificado que o uso da técnica melhora a qualidade dos programas e contribui para a motivação dos alunos. No relato descrito em [Mendes, Al-Fakhri and Luxton-Reilly, 2005] é apresentada uma pesquisa, na qual os pares trocavam de papéis a cada 20 minutos (um é o controlador do mouse e do teclado e outro é um avaliador), conforme definição do método. Os pares trabalhavam juntos por quatro sessões e depois também eram mudados.

Em *extreme programming*, conforme definido em [Beck, 1999] os pares devem possuir mais ou menos o mesmo *background* para que as trocas de papéis fiquem equilibradas. Em nenhum dos três trabalhos citados acima são utilizadas quaisquer técnicas para a formação dos pares, o que pode ter influenciado a aferição do desempenho dos alunos. Além disso, não foram mantidos registros das interações, sendo impossível verificar se os alunos realmente trabalhavam juntos. Com a formação de times nas empresas de desenvolvimento de software, a pesquisa em aprendizagem de programação vem ganhando maiores possibilidades de uso de técnicas. Uma delas é o

desenvolvimento de programas e diagramas de modelagem, conforme utilizado por empresas, por equipes de desenvolvimento. Para isso, faz-se necessário o suporte computacional adequado.

3. Integração de Controle de Versões e *Groupware*

Conforme descrito na Seção anterior, há muitas tentativas de facilitar a aprendizagem de programação. Algumas iniciativas utilizam mais aulas práticas associadas a um maior acompanhamento, enquanto outras procuram manter o foco nos tipos de exercício propostos. Todos os relatos analisados ainda evidenciam significativas lacunas quanto à aprendizagem de programação.

As disciplinas introdutórias de programação são imprescindíveis para a formação dos alunos de computação, pois é nelas que se determina do que se constitui a programação e o que é necessário estudar para se aprender a programar. Neste sentido, na UFAM, a IES onde este projeto é aplicado, acredita-se que programação é uma atividade de resolução de problemas, que apresenta uma maneira específica de descrever soluções. No entanto, a resolução de problemas não é fácil, pois os alunos não são acostumados ao planejamento de soluções e ao uso de etapas bem definidas para atingir a solução de um problema.

Na tentativa de se descobrir em quais tópicos do curso introdutório os alunos apresentavam mais dificuldades, foi realizado um estudo de caso, descrito em [Castro *et al*, 2005], com duas turmas de alunos iniciantes onde, na aula prática semanal, os alunos recebiam um ou dois problemas que deveriam ser solucionados ao longo da aula, com direito a consultas bibliográficas e trocas com outros alunos e monitores. Dos cerca de 80 alunos acompanhados, 10 participaram do grupo de observação. O acompanhamento desse estudo piloto consistiu na descrição de reflexões anteriores e posteriores à investigação e representação da solução dos alunos. Antes que ocorresse a sessão seguinte de laboratório, os pesquisadores liam as observações dos alunos, dos monitores e dos assistentes de pesquisa para confrontarem as informações e, quando adequado, convidarem os alunos, individualmente, para uma entrevista realizada conforme o método clínico proposto por Jean Piaget, explicado em [Del Val, 2002].

A análise desses resultados culminou na concepção e desenvolvimento de uma ferramenta de acompanhamento das soluções dos alunos, chamada AAEP, com possibilidade de, sempre que o professor julgar necessário, resgatar uma comparação entre 2 de quaisquer versões de um mesmo estudante. Conforme estudo de caso descrito em [Almeida *et al*, 2006], o AAEP foi projetado para atender essencialmente às demandas do professor. Portanto, as funcionalidades de gerência, como cadastramento de usuários, cadastramento de problemas e a análise das soluções são de acesso restrito ao mesmo. As outras funcionalidades, como elaboração, edição e teste de código-fonte de programas, associadas a comentários caracterizando cada versão, podem ser realizadas tanto por estudantes quanto por professores.

O AAEP incorpora três categorias de serviços: a gerência de problemas e de usuários, a operação de um interpretador da linguagem de programação utilizada, e um controle de versões dos programas gerados. Como já existem ferramentas confiáveis e de código-aberto para realizar o controle de versões e o interpretador utilizado deveria

ser aquele utilizado na disciplina de programação considerada, o desenvolvimento da ferramenta teve como elemento central a construção de uma camada para gerência de usuários e problemas, e de estruturas para a agregação de serviços externos. Os resultados deste estudo serviram de subsídio para o desenvolvimento de uma ferramenta para o acompanhamento do desenvolvimento de programas.

Embora esse ambiente tenha atendido à demanda inicial dos cursos introdutórios, percebeu-se que não era suficiente para todos os casos. Por exemplo, mesmo não se tratando de trabalhos em grupo, os alunos costumam trocar informações sobre os exercícios, o que não é apoiado pelo ambiente. Acredita-se que programação, assim como outras atividades cognitivas, é mais facilmente compreendida quando realizada em equipe, o que motivou a proposta de integrar ao ambiente de apoio ao acompanhamento e desenvolvimento de programas, uma ferramenta para trabalho em grupo, o AulaNet [Fuks et al, 2004], de modo a possibilitar o acompanhamento e análise do comportamento dos alunos perante o grupo do qual participam, especialmente a transposição de práticas e estratégias do individual para o coletivo, procurando perceber possíveis mudanças de comportamento perante os problemas sugeridos.

A implantação desse novo ambiente e sua utilização em situação real são elementos fundamentais para a aprendizagem de programação em grupo, que envolve, dentre outras ações atuação nas seguintes frentes de trabalho:

- Verificar se os alunos procuram códigos para problemas semelhantes antes de tentarem resolver um problema específico, o que evidenciaria a importância de soluções-exemplo e aprendizagem de programação por análise de programas prontos;
- Evidenciar a importância do registro, tornando o processo de planejamento das soluções mais explícito;
- Observar se a interação leva a resoluções mais rápidas, pois um dos pressupostos considerados é que se aprende a programar mais facilmente em equipe;
- Observar incorporações (à pratica) do individual para o coletivo, procurando perceber possíveis mudanças de comportamento perante os problemas sugeridos;
- Verificar se há reaproveitamento de suas próprias soluções anteriores. Isto reforça a necessidade do uso de ambientes de apoio à programação e ao trabalho em grupo utilizados;
- Analisar o comportamento dos alunos perante o grupo: critérios de escolha de soluções condizentes com as avaliações individuais.

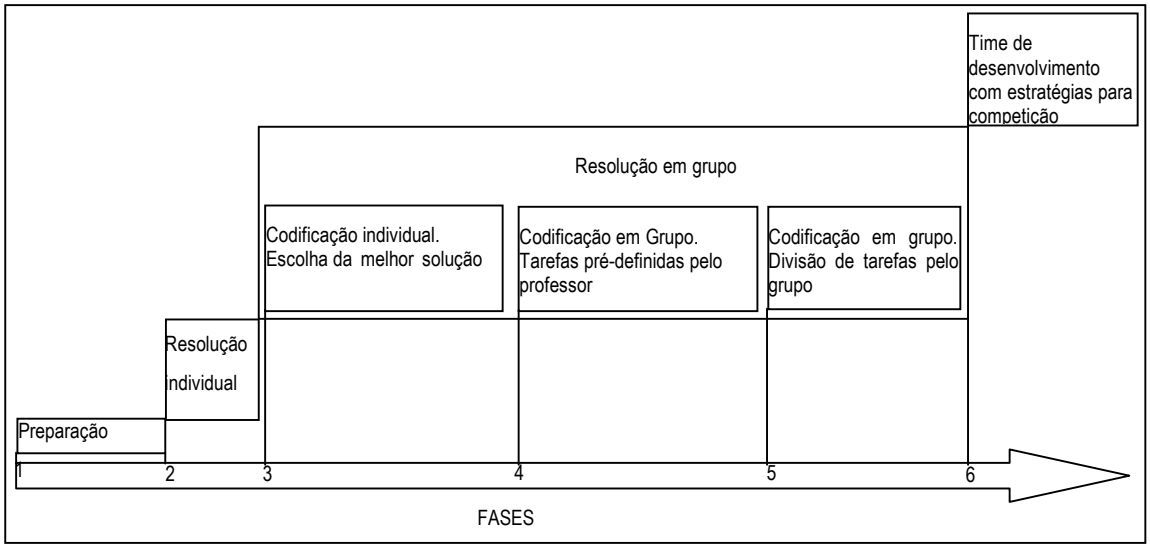

Figura 1 – Esquema de progressão de construções (individuais para coletivas)

A Figura 1 ilustra um possível esquema de organização de atividades que define uma progressão do trabalho individual para o coletivo, num cenário que inicia na etapa 1, com uma preparação que envolve trabalho em laboratório com problemas simples e esclarecimentos sobre a "metodologia" de progressão. A fase 2 consiste do registro e resolução exclusivamente individuais. Na fase 3 o trabalho em grupo é iniciado a partir da escolha da melhor solução construída individualmente. Na fase 4, a resolução de problemas também passa a ser coletiva, sendo que as tarefas são definidas pelo professor, cabendo ao grupo a definição dos "atores" para cada atividade. Na fase 5, o grupo fica responsável pela definição de tarefas e correspondente definição de atores. Por fim, na fase 6 há um trabalho de desenvolvimento onde os grupos competem entre si, num formato mais próximo de situações reais de trabalho.

4. Estudo de Caso para Aprendizagem de Programação em Grupo

Baseados em nossa experiência com ensino e, mais especificamente, ensino de programação, elaboramos um estudo de caso para aferir comportamentos na aprendizagem de programação em grupo que segue o esquema de progressão de construções definido no final da seção anterior e com isso, traçar os passos seguintes desta pesquisa. Este estudo de caso, a exemplo dos anteriores, foi realizado com alunos iniciantes nos cursos de Engenharia da Computação e Ciência da Computação da UFAM, no período acadêmico de 2007.1.

Segundo evidenciado nos estudos de caso anteriores, os alunos costumam ingressar nos cursos de computação sem saber colaborar. Normalmente eles fazem colagens de material ao invés de elaboração de sínteses. Na disciplina de Introdução à Computação, as aulas foram divididas entre teóricas e práticas. As aulas teóricas foram ministradas uma vez por semana e as outras duas foram divididas entre práticas em sessões de laboratório e práticas externas. Estas práticas externas foram utilizadas para os trabalhos em grupo, sendo obrigatório o registro digital das interações. Para tanto foi

fornecido um ambiente de apoio ao trabalho em grupo, contendo todo o conteúdo trabalhado ao longo do curso e espaços disponíveis para registro de interações. A Tabela 1 descreve o planejamento das atividades práticas, a Tabela 2 associa um problema exemplo a cada fase do estudo de caso.

Tabela 1 - Cronograma de Atividades Práticas

Fases	Semana no Curso	Descrição do Conteúdo
1. Preparação	1 e 2	Levantamento inicial: os alunos responderam a um questionário, disponível no ambiente de apoio ao trabalho em grupo.
	3	Definição dos níveis de habilidade: Atividade prática em laboratório, utilizando o AAEP, com sessão pré-definida, utilizando problemas geométricos básicos (90 minutos) e análise de resultados baseada no tempo de resolução.
2. Resolução I	5	Resolução de 1 exercício em Haskell, com registros individuais feitos como anotações.
3. Resolução II elaboração de soluções coletivas	6	Análise e seleção das soluções individuais; refinamentos; registro coletivo do processo, como chats e conversas presenciais gravadas digitalmente.
4. Resolução III sistematização dos processos de construção coletiva	8	Problema 1 – os alunos resolviam conforme as fases anteriores.
	9 e 10	Problema 2 – uma parte para cada aluno, mas o professor faz a divisão do trabalho.
	11, 12 e 13	Problema 3 – os próprios alunos devem utilizar técnicas de modularização para dividir o trabalho e escolher quem desenvolve qual parte, documentando essas tomadas de decisão.
5. Resolução IV competição	14	Estilo de maratona de programação, com um prêmio em jogo. Consistiu em: observação externa por alunos mais avançados ou de mestrado; utilização de uma ferramenta para monitoramento das atividades do grupo; e etapas para resolução do problema da maratona: registro de coisas a observar; registro para cada elemento da solução; responder questionário.
6. Avaliação do Processo	16	Aplicação de questionário e entrevistas.

A partir da fase de resolução II, os alunos foram orientados para trabalhar em grupos. Estes grupos foram definidos na terceira semana de práticas, após a aplicação da sessão de laboratório supervisionada. O requisito adotado era de que cada grupo fosse o mais heterogêneo possível, contendo ao menos 5 integrantes, dos quais no máximo 2 com experiência formal em programação, 1 com alguma experiência (ex. Programação em scripts para web) e 2 ou 3 com pouca ou nenhuma experiência em programação.

Tabela 2 - Exemplos de Exercícios por Fase

Fases do Curso	Exercício Proposto
Preparação	"Um grupo de quatro amigos deseja fazer uma compra cujo valor total ultrapassa a soma dos valores que cada um possui. Para efetuar a compra o grupo decidiu que a quantia que faltasse seria dividida proporcionalmente ao que cada um já possui, usando o raciocínio de que quem contribuiu com mais dinheiro pode arranjar mais. Descreva como você pode resolver este problema através de um script em Haskell, indicando quanto cada um deverá pagar"
Resolução I	Problema geométrico
Resolução II	Problema da senha em um pronto socorro
Resolução III	Problema da formação da tabela de campeonato
Resolução IV	Problema do banco de sangue

Após a formação dos grupos, foram introduzidos exercícios com maior grau de complexidade (Tabela 2), conforme a mudança de fases e o avanço no conteúdo da disciplina. Isto tornou possível a mudança gradativa de um ritmo de trabalho baseado em práticas individuais à incorporação de práticas de desenvolvimento coletivos de programas.

5. Conclusão

Este trabalho discutiu como a aprendizagem de programação pode fazer uso da colaboração e ser auxiliada por um conjunto de ferramentas integradas em ambientes de registro dos artefatos e das interações desenvolvidas. A partir dessa estrutura de trabalho, um estudo de caso para a validação de pressupostos e prospecção de novos elementos foi concebido, implementado e está na fase de análise dos resultados.

A dinâmica das sessões práticas tem sido incondicionalmente bem aceita, o que é consistente com um aumento na participação e no interesse pelas atividades. A partir de uma análise aprofundada do conteúdo digital produzido será possível analisar as estratégias utilizadas pelos alunos nas construções coletivas de programas e, com isso prosseguir na investigação de um modelo de colaboração para aprendizagem de programação.

Referências

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

Beck, K. (1999) "Extreme Programming Explained: Embrace Change". Addison Wessley. Reading, MA, USA. ISBN 0-201-61641-6.

Brooks, F. P. (1975) "The mythical man-month: essays on software engineering". Reading, MA : Addison-Wesley, 195p. ISBN 0201006502.

Castro, T. H. C. ; Castro Jr, A. N. ; Menezes, C. S. ; Boeres, M. C. S. ; Rauber, M. C. P. V. (2002). "Utilizando Programação Funcional em Disciplinas Introdutórias da Computação". 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. v. 4. p. 157-168.

Castro, T. ; Menezes, C. S. ; Castro, A. N. ; Oliveira, R. S. C. ; Boeres, M. C. S. (2005) "Enhancing Programming Understanding through Conceptual Schemas in Introductory Courses". In: CLEI Eletronic Journal, vol. 9, number 2, http://www.clei.cl/cleiej/volume.php.

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

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

Del Val, J. (2002) "Introdução à Prática do Método Clínico". Artmed. ISBN 8536300132.

Dijkstra, E. W. (1976) "A discipline of programming". Englewood Cliffs, N. J. Prentice-Hall. 217p. - ISBN 013215871X.

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

Fuks, H., Gerosa, M.A. e Lucena, C.J.P. "Using the AulaNet Learning Environment to Implement Collaborative Learning via Internet". (2003) in: Aung et al. (ed), Innovations 2003 - World Innovations in Engineering Education and Research, iNEER, USA, Chap. 23, pp. 225-235 ISBN 0-9741252-0-2.

Goodhue, Dale L.; Thompson, Ronald L. (1995) "Task-technology fit and individual performance", MIS Quarterly, 19, 2, 213-236.

N. Jennings. (1993) "Coordination: Commitment and conventions, the foundation of coordination in multiagent systems". Knowl. Eng. Rev. 8(3), 223-250.

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

McDowel, C.; Werner, L.; Bullock, H. and Fernald, J. (2002) "The Effects of Pair-Programming on Performance in an Introductory Programming Course". In: SIGCSE, ACM 1-58113-473-8/02/0002. Corvington, Kentucky, USA.

McGrath, J. E., & Hollingshead, A. B. (1993). "Putting the "group" back in group support systems: Some theoretical issues about dynamic processes in groups with technological enhancements". In L. M. Jessup & J. S. Valacich (Eds.), Group support systems: New perspectives (pp. 78-96). New York: Macmillan.

Mendes, E.; Al-Fakhri, L. and Luxton-Reilly, A. (2005) "Investigating Pair-Programming in a 2nd-year Software Development and Design Computer Science Course". In: ITiCSE, ACM 1-59593-024-8/05/0006. Monte de Caparica, Portugal.