# 6
# Conclusão

Esta tese descreve uma abordagem sistematizada para aprendizagem de programação. Essa abordagem 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.

Através de três estudos de caso, uma inspeção no ColabWeb, ambiente CSCL utilizado para a pesquisa desta tese e a representação formal de padrões de interação provamos a hipótese: **Oportunidades de intervenção na aprendizagem de programação em grupo são ampliadas com o uso de uma abordagem sistemática de acompanhamento**.

A teoria de desenvolvimento cognitivo evidenciada por Piaget é descrita a partir de experimentações com o método clínico utilizado para avaliar os estágios de desenvolvimento cognitivo em que os sujeitos se encontram (Piaget & Inhelder, 1968), (Piaget, 1972), (Piaget, 1978), (Piaget, 1995). 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.

Assim como a criança passa por estágios para desenvolver seu pensamento, entendemos que os adolescentes precisem também passar por estágios no desenvolvimento de pensamentos mais abstratos como os exigidos em programação. O problema é que esses processos não estão claros para o professor nem para os próprios alunos, uma vez que somente se tem acesso aos programas prontos e as pessoas esquecem facilmente o raciocínio que as levou a chegar a determinada solução. Sendo assim, desenvolvemos uma ferramenta chamada AcKnow que procura modificações nas versões dos programas dos alunos previamente capturadas e armazenadas em um banco de dados. Essas modificações na evolução dos códigos são classificadas em sintáticas, semânticas

e de refactoring. O AcKnow investigou os códigos dos alunos de 2007, após a disciplina ser encerrada e foi identificado um padrão de muitas modificações sintáticas seguidas de algumas semânticas intercaladas com sintáticas. Em alguns casos, houve até modificações de refactoring, o que significa uma boa capacidade de enxergar a solução e aprimorá-la contemplando as noções da disciplina de eficiência e legibilidade do código.

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

Nesse ponto da pesquisa já sabíamos que exercícios em grupo eram agradáveis aos alunos, que eles poderiam contribuir mais para a aprendizagem e como os alunos organizavam seu pensamento na codificação. O próximo passo foi refletir sobre uma forma de sistematizar a aprendizagem de programação em grupo, considerando que os alunos não conseguiam se organizar facilmente para programar. Então foi proposto o esquema progressivo de aprendizagem de programação em grupo, que atua como um macro-script para colaboração, introduzindo gradativamente a colaboração nas práticas dos alunos. Antes de testá-lo planejamos a disciplina de Introdução à Computação de 2008 como um curso no ambiente CSCL ColabWeb.

Em 2008 o esquema progressivo de aprendizagem de programação em grupo foi posto em prática com uma turma real de alunos iniciantes em Ciência da Computação e Engenharia da Computação. A turma de Engenharia da Computação apresentou dificuldades no acompanhamento do esquema devido a fatores externos como a configuração do ambiente do curso no ColabWeb e infra-estrutura do laboratório. Consideramos para análise, por esse motivo, somente a turma de Ciência da Computação. O esquema progressivo se mostrou 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.

Avaliamos esse ambiente sob a perspectiva do Método de Inspeção Semiótica proposto por (De Souza, 1995) como um método de inspeção dos signos metalingüísticos característicos do poder de comunicabilidade do projetista da interface com o usuário. Mostramos que muitas sugestões de navegabilidade e visualização da interface herdadas do Moodle não são adequadas para o contexto de aprendizagem de programação em grupo. Fizemos recomendações de utilização de padrões de interface que em sua maioria foram acatadas pelo professor e aplicadas juntamente com o estudo de caso explanatório de 2009.

Para a análise do estudo de caso de 2008 foi realizado uma revisão métodos para colaboração e em formas de análise de interação e atos de fala. 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.

O último estudo de caso, um estudo explanatório, aplicado à turma de calouros de Engenharia da Computação de 2009 foi desenvolvido seguindo o mesmo plano do de 2008 com o intuito de comprovar a aplicabilidade dos padrões de interação e refinar os estereótipos. A análise foi realizada e os padrões de interação foram facilmente identificados. Alguns estereótipos foram refinados e outro foi incluído.

## 6.1. Contribuições

O caminho percorrido para alcançar o objetivo central da investigação – uma sistematização para aprendizagem de programação em grupo – envolveu a produção de elementos que acreditamos contribuem para o conhecimento na área. São eles:

1. Uma série de estudos de caso que possibilitam a exploração de vários aspectos relacionados à aprendizagem de programação, compreendendo desde a caracterização da construção individual de soluções até a elicitação de padrões comportamentais de grupos atuando em vários estágios do processo.
2. A definição de categorias da evolução de códigos dos alunos e o desenvolvimento de um artefato para gestão do conhecimento em desenvolvimento de programas.
3. A aplicação da Engenharia Semiótica na análise e reformulação de interface em software com vários níveis de comunicabilidade entre designer e usuários em plataforma LMS de uso intensivo.
4. Um esquema progressivo para a aprendizagem de programação em grupo que induz e apóia os alunos numa transição de trabalho e aprendizagem essencialmente individual para um estágio colaborativo de atuação.
5. A definição de um conjunto de padrões de interação e a caracterização de estereótipos da ocorrência desses padrões. Esses elementos foram formalizados e podem ser integrados a plataformas multiagente de apoio às atividades dos atores envolvidos na aprendizagem.



## 6.2. Reflexões Adicionais no Tema

O esquema progressivo de aprendizagem de programação em grupo pode ser utilizado em qualquer disciplina ou contexto de trabalho em equipe, desde que a abordagem metodológica utilizada tenha um processo com etapas bem definidas, como é o caso do processo de solução de problemas de Polya, utilizado nos estudos de caso aqui relatados.

Não é imprescindível a utilização de um ambiente CSCL para o acompanhamento das etapas do esquema progressivo. Se o contexto for uma disciplina de natureza mais discursiva, o ambiente pode ajudar, mas não é essencial. Nesse caso, o acompanhamento pode ser realizado na sala de aula e pela análise dos textos produzidos em sala de aula. No contexto de programação, dado o seu caráter extremamente abstrato, a utilização de um ambiente CSCL, aliado às técnicas de identificação dos padrões de interação propostas neste trabalho, a utilização de um ambiente CSCL configurado segundo alguma técnica de

avaliação de comunicabilidade, como o MIS é imprescindível. Os padrões de interação propostos nesta tese se aplicam somente ao contexto de programação em grupo, embora possam ser reavaliados e estendidos para comportarem outros contextos.

Os padrões de interação encontrados na análise do estudo de caso descritivo provaram-se aplicáveis ao contexto de aprendizagem de programação descrito, um curso introdutório de programação utilizando o paradigma funcional. Analisando as características das conversas referentes aos exercícios de programação observamos que eles se aplicam a esses exercícios porque os alunos possuem uma atividade de resolução de problemas para desenvolver. Portanto, se a matéria estudada for engenharia de software ou arquitetura de computadores, se a metodologia utilizada for resolução de problemas, os padrões de interação ainda serão aplicáveis.

Uma vez utilizados os padrões de interação, os estereótipos definidos precisam ser dinâmicos, pois dependem do objetivo do professor e características da turma. Ao aplicar os padrões de interação como mecanismo de análise a um novo contexto, o professor define a priori alguns estereótipos para ter um ponto de partida para sua análise. Ao longo da primeira aplicação da análise ele pode refinar os estereótipos previstos e identificar outros.

Após a primeira aplicação da análise utilizando os padrões de interação e estereótipos a um novo contexto, se os padrões de interação já estiverem incorporados a um ambiente CSCL, o professor precisa ter acesso à criação de novos estereótipos e adaptação dos existentes para outros contextos ao seu.



## 6.3. Trabalhos Futuros

A seguir listamos alguns desdobramentos ou aprofundamentos da investigação relatada nesta tese.

- Inserir toda sistematização no Open Knowledge, inclusive definindo assistentes inteligentes para atuar na intervenção.
- Aplicar diferentes combinações dos elementos a times de desenvolvimento de software – com experimentos exploratórios em disciplinas avançadas de desenvolvimento de software; em empresas *start-up* ou incubadas; em empresas com projetos na UFAM.

- Investigar a incorporação de outras ferramentas nos diversos estágios da sistematização – por exemplo, de *clustering*, de análise qualitativa, de mineração de dados, etc.
- Aplicar a sistematização a outros domínios de formação onde a natureza da atividade envolvendo abstração provoque bloqueios e dificuldades similares nos alunos.
- Aplicar a sistematização no mesmo domínio (aprendizagem de programação), porém utilizando outro paradigma não imperativo – programação de jogos, programação visual, sistemas dinâmicos, etc.

## 6.4. Publicações de Resultados Parciais da Tese

A seguir listamos os relatos de resultados parciais desta tese, apontando também a contribuição a que se refere conforme os itens na Seção 6.1.

- CASTRO, T., FUKS, H., CASTRO, A. & SPÓSITO, M. Integração de Ferramentas para Acompanhamento da Aprendizagem de Programação. Anais do XVIII Simpósio Brasileiro de Informática na Educação – SBIE 2007 / Workshop - Ambientes de apoio à aprendizagem de algoritmos e programação, ISBN 978-85-7669-159-4, São Paulo, SP, 2007.
  [1]
- 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, 2008, Santander, Spain.
  [1]
- CASTRO, T., FUKS, H. & CASTRO, A. Detecting Code Evolution in Programming Learning. In Proceedings of the 19th Brazilian Symposium on Artificial Intelligence, Salvador, Brazil, October 26-30, 2008, Salvador, Brazil. Series: Lecture Notes in Computer Science , Vol. 5249. Sublibrary: Lecture Notes in Artificial Intelligence. ISBN: 978-3-540-88189-6, pp.145-156.
  [2]

- CASTRO, T., FUKS, H. & CASTRO, A. Programming in Groups: a Progression Learning Scheme from the Individual to the Group. FIE - Proc. of the 38th Annual Frontiers in Education Conference, October 22-25, 2008, Saratoga Springs, New York, USA. IEEE Catalog Number: CFP08FIE-CDR. ISBN: 978-1-4244-1970-8. Library of Congress: 79-640810. ISSN: 0190-5848. Pp F1F15-F1F20.
  [4]
- CASTRO, T., FUKS, H. & CASTRO, A. Aprendendo a Programar em Grupo. Anais do V Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2008. 27 a 29 Outubro 2008, Vila Velha, ES. ISBN: 978-0-7695-3500-5/08, Ed. IEEE-CS, pp. 45-54.
  [4]
- CASTRO, T., FUKS, H., SANTOS, L. & CASTRO, A. Fleshing out Clues on Group Programming Learning. ICEIS 2009, 11th International Conference on Enterprise Information Systems, Milan, May 2009. ISBN: 978-989-8111-85-2.
  [5]
- 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
  [3]
- CASTRO, T., FUKS, H., SPÓSITO, M. & CASTRO, A. Análise de um Estudo de Caso para Aprendizagem de Programação em Grupo. IEEE-RITA: Revista Iberoamericana de Tecnologia del Aprendizaje. ISSN: 1932-8540. V.4, N.2, pp. 155-160. 2009.
  [1]