4
Colaboração na Aprendizagem de Programação

Neste capítulo discutimos conceitos de colaboração e como os métodos de colaboração são utilizados para apoiar a aprendizagem de programação em grupo em ambientes CSCL. Inicia-se com uma discussão sobre a necessidade de colaboração em aprendizagem de programação e a formação dos grupos. Em seguida, apresentamos alguns exemplos de métodos conhecidos, adaptados de (Sharan, 1999) para o contexto de aprendizagem do próprio processo de colaboração. Na Seção 4.1 discutimos sobre como os scripts de colaboração auxiliam na aquisição de habilidades para colaboração e na análise das interações. Considerando que os alunos adquiram as habilidades para colaboração e seguindo a ênfase na necessidade de análise das interações produzidas, na Seção 4.2 discutimos sobre os métodos de análise de interações utilizando padrões de interação e atos de fala. Os ambientes CSCL são projetados para fornecerem ferramentas que proporcionam a aplicação de métodos para apoiar a colaboração e sua análise, mas algumas outras tecnologias podem enriquecer esses ambientes, estruturando-os internamente, não afetando a maneira como o usuário se relaciona com as ferramentas.

Após as duas primeiras seções, que são subsídios para nossa proposta de intervenção na aprendizagem de programação em grupo, apresentamos, na Seção 4.3, um estudo de caso exploratório desenvolvido para se conhecer as dificuldades e características apresentadas por alunos iniciantes em programação ao realizarem um exercício em grupo, sem restrições para a organização dos grupos e utilização de métodos. Baseados nos resultados dessa exploração, na Seção 4.4, apresentamos um esquema que, inspirado nos scripts de colaboração, propõe uma transição das práticas de programação individuais para a prática em grupo.

Uma das práticas aceitas (Parrat & Tryphon, 1998) em contextos de aprendizagem é a colaboração como forma de se desenvolver os processos cognitivos, desde que seja respeitada a fase do desenvolvimento em que os alunos se encontram. Conforme afirmado por Piaget em (Parrat & Tryphon, 1998),

geralmente a partir dos 10 ou 11 anos, o respeito às regras nas atividades em grupos passa a ser uma realidade intrínseca e não algo externo, imposto.

No contexto de aprendizagem de programação abordado nesta tese, os alunos mais novos já identificados têm 16 anos. Nessa idade, apesar da provável falta de prática nas escolas, a colaboração já está incorporada à realidade do pensamento deles e às suas tarefas cotidianas.

O trabalho em grupo é benéfico a esses alunos por ser ativo e investigativo, proporcionando as trocas de idéias e as negociações para se chegar a uma concepção do grupo. Para isso, afirma Piaget (Parrat & Tryphon, 1998), que o trabalho em grupo não pode ser totalmente prescrito de fora, devendo envolver uma negociação sobre a formação dos grupos, alinhadas aos interesses dos alunos e professores. O grupo também funciona como um mecanismo regulador do pensamento individual, servindo ao mesmo tempo de estimulador e órgão de controle.

A colaboração é incentivada também nos ambientes de trabalho. No mercado de trabalho da Engenharia de Software, por exemplo, é comum a organização das equipes de desenvolvimento de software por projetos, onde somente um líder é indicado, cabendo a ele formar sua equipe e distribuir as tarefas. Para isso, há técnicas de gestão do conhecimento provenientes de métodos organizacionais que são muito utilizadas. Os egressos dos cursos de computação precisam conhecer e estar fluentes em colaboração, envolvendo diferentes métodos e técnicas. Para isso, é importante que os alunos sejam expostos ao desenvolvimento de programas em grupo desde o primeiro curso de programação.

Relembrando o que foi abordado no Capítulo 2, a colaboração em programação em grupo não ocorre naturalmente, devido à falta desta prática na educação básica, que privilegia a memorização de muito conteúdo, em vez do desenvolvimento do raciocínio. Dado que programação envolve raciocínio muito abstrato, os alunos precisam primeiramente entender os processos para se chegar à solução de um problema, codificado em um programa, para posteriormente perceberem a necessidade das trocas de idéias para gerarem soluções melhores.

Assim como aprendizagem no contexto da educação básica, a colaboração entre os integrantes de um grupo aprendendo programação também precisa ser mediada. Através desta mediação, o professor pode intervir enquanto os alunos ainda estão elaborando sua solução, através da identificação de possíveis

estagnações nas discussões e soluções parciais dos grupos. Ao utilizar um ambiente CSCL para registrar as conversas dos grupos referentes a cada exercício, o professor terá oportunidades de intervir durante o processo de aprendizagem. Para que isso ocorra é necessário que o ambiente CSCL utilizado ofereça mecanismos para estruturação das conversas espontâneas, de forma que essas estagnações sejam identificadas e o professor seja avisado em tempo de intervir durante o processo, podendo para isso utilizar uma linguagem de representação das interações.

4.1.Scripts para Apoiar o Processo de Colaboração

Os alunos utilizando ambientes CSCL para aprender programação têm acesso e precisam dos mesmos recursos que alunos aprendendo qualquer outro assunto, como ferramentas de chat, fórum, relatórios em estilo wiki e esquema de armazenamento de arquivos. Há relatos (Kobbe, Weinberger, Dillenbourg, Harrer, Hämäläinen & Häkkinen & Fischer, 2007) que afirmam que a aprendizagem colaborativa depende da interação efetiva entre os alunos. No entanto, quando esses alunos são deixados sozinhos para utilizarem as ferramentas do ambiente para interagirem, sem nenhum direcionamento, eles raramente se envolvem em interações produtivas como questionar uns aos outros, articular o raciocínio ou elaborar reflexões. Os scripts para colaboração foram propostos para incentivar a aprendizagem colaborativa moldando a maneira como os alunos interagem. Para isso, especificam uma sequência de atividades de aprendizagem e papéis que deverão ser assumidos pelos alunos durante a interação. Em sua definição clássica (O'Donnel & Dansereaus, 1992), um script para colaboração é um conjunto de instruções referentes a como os integrantes do grupo devem interagir, como devem colaborar e como devem resolver o problema.

Na tentativa de obter um panorama do que já se havia escrito sobre scripts para colaboração e facilitar sua adoção em ambientes CSCL, em (Kobbe, Weinberger et al, 2007) é proposto um framework, contendo os mecanismos e componentes necessários. Um script deve ter a descrição detalhada de atividades, estabelecer papéis com suas expectativas e funções, recursos virtuais ou físicos, grupos, mecanismos como distribuição de tarefas, formação de grupos e seqüenciamento.

Em (Villasclaras-Fernández, Isotani, Hayashi & Mizoguchi, 2009) é proposta uma abordagem utilizando ontologias, unificando os conceitos de micro-scripts e macro-scripts, classificação atual para diferenciar a ênfase na construção de argumentos ou sequências de argumentos (micro-scripts) da preocupação com organização de sessões de interação, incluindo a descrição de grupos, papéis, atividades ou classificação de sentenças (macro-scripts). Essa abordagem propõe uma ontologia para a criação de atividades envolvendo os dois níveis.

As desvantagens da utilização de scripts (Dillenbourg, 2002) se traduzem em preocupações que devem estar presentes em quem vai propor ou utilizar scripts. Os professores devem observar durante o desenvolvimento das atividades propostas se os scripts estão perturbando as interações naturais ou os processos naturais de soluções de problemas, aumentando a carga cognitiva, transformando as interações em uma sequência didática ou ainda fazendo as interações se tornarem sem objetivos claros.

4.2. Análise de Interações em Ambientes CSCL

Considerando que a aprendizagem de programação em grupo seja apoiada por um ambiente CSCL e que este ambiente de alguma forma incentive a colaboração e a utilização de fóruns e chats através da definição de scripts de colaboração ou outra metodologia, a análise das interações requer um trabalho minucioso que ao mesmo tempo seja realizado durante o processo de discussão para a resolução dos exercícios e, no caso específico de programação, durante o desenvolvimento dos códigos.

Para utilizar as informações provenientes das análises de interações, em (Dillenbourg & Fischer, 2007) é sugerido que se possibilite sua visualização aos membros do grupo. Essas ferramentas são chamadas “group mirrors”, pois refletem alguma forma de interpretação externa das interações aos usuários.

Ambientes CSCL devem ser desenvolvidos para proporcionarem interações que produzam bons produtos. Algumas maneiras de estimular essas interações são: deixar os alunos em uma situação onde eles precisem se envolver em interações trabalhosas para construírem um entendimento compartilhado; selecionar uma representação de tarefa que molde a linguagem utilizada pelos alunos; apontar diretamente tipos específicos de “utterances” utilizando interfaces

semi-estruturadas; estruturar colaboração através de scripts; capturar interações, permitindo sua visualização pelos membros dos grupos ou conduzindo análises mais profundas (Dillenbourg & Fischer, 2007).

Ainda em (Dillenbourg & Fischer, 2007), é descrito o termo ‘orquestração’ para identificar os desafios de coordenar produtivamente intervenções em níveis diferentes, enfatizando o papel do professor ao conduzir atividades de CSCL em escolas ou universidades. A orquestração se refere às dimensões cognitiva, pedagógica e prática de um ambiente CSCL distribuído. No nível cognitivo, o professor precisa equilibrar as tarefas em mecanismos de aprendizagem individualizada, interações em pequenos grupos e atividades com toda a turma. No nível pedagógico, o professor precisa adaptar em tempo real as atividades planejadas ao que está ocorrendo naquele momento em sala de aula.

Neste trabalho consideramos a pesquisa em scripts de colaboração como forma de estruturar o curso de programação introdutória e facilitar a análise das interações. No entanto, as desvantagens do uso de scripts apontadas na seção anterior nos levou à reflexão sobre o caráter abstrato da programação já discutido no Capítulo 2 e em como poderia ser prejudicial a adoção de mecanismos externos, micro-scripts, para o processo de elaboração do raciocínio em programação.

Independente da utilização de micro-scripts é necessário se adotar mecanismos para acompanhar as interações e outros ainda para analisá-las. Este trabalho tem como objetivo a sistematização da aprendizagem de programação em grupo, o que envolve acompanhamento e análise das interações. Para elaborar esses mecanismos é necessário que se explore o contexto de aprendizagem de programação que se deseja trabalhar, observando como os grupos se organizam para resolver um exercício de programação em grupo. Por isso, foi desenvolvido um estudo de caso exploratório, conforme definição em (Yin, 2010).

4.3.Um Estudo de Caso Exploratório sobre Aprendizagem de Programação em Grupo

No primeiro semestre de 2007 foi desenvolvido um estudo de caso em introdução à programação com duas turmas de alunos matriculados no primeiro

semestre dos cursos de graduação em Ciência da Computação e Engenharia da Computação na Universidade Federal do Amazonas (UFAM).

O objetivo desse estudo de caso foi proporcionar aos alunos a experiência no desenvolvimento de soluções para problemas complexos através da distribuição de tarefas, negociação, composição de soluções parciais e refinamentos sucessivos. Isto foi alcançado trabalhando em grupos de até 5 alunos os quais eram também responsáveis pelo registro das atividades desenvolvidas ao longo das etapas do trabalho, utilizando um ambiente baseado no controle de versões chamado AAEP (Almeida, Castro & Castro, 2006). Ao final do curso, como trabalho final, foi proposta uma tarefa em grupo. Conforme exigência da tarefa, a solução deveria ser acompanhada de um registro das interações entre os membros das equipes. Após a entrega e correção dessas tarefas, houve uma fase de análise e interpretação dos dados gerados baseados nos processos de classificação, codificação e tabulação.

Além do desenvolvimento de códigos e manutenção do registro das interações, os alunos também responderam a questionário, o qual foi submetido a uma análise quantitativa. Finalmente, eles puderam expressar livremente sobre suas dificuldades e sobre o desenvolvimento dos trabalhos, sendo esses comentários submetidos a uma análise qualitativa.

4.3.1. Análise Quantitativa

O objetivo do questionário era revelar o nível de aderência à colaboração na aprendizagem de programação como uma forma de mudar de práticas individuais para aprendizagem de programação em grupo através da interação entre os membros dos grupos. Os dados foram analisados de acordo com a distribuição absoluta sugerida em (Levin, 1987) descrita na Tabela 4.1.

Tabela 4.1 – Distribuição de Critérios para Estudo Experimental

Critérios	*Respostas*		
	Totalmente	Parcialmente	Em branco / Não observado
Seqüências sugeridas	7	2	0
Metas alcançadas	7	2	0
Problema resolvido	7	1	1
Busca por códigos similares	4	2	3
Anotações satisfatórias	6	3	0
Integração entre os membros do time	4	3	2
Reutilização dos códigos do próprio time	8	1	0

Conforme destacado na análise mostrada na Tabela 4.1, houve 9 grupos de respondentes e em todos os critérios houve uma predominância de respostas “Totalmente”, o que indica satisfação e sucesso na execução da tarefa em grupo. Na maioria dos critérios, a maioria das respostas corresponde à primeira coluna (Totalmente) indicando que houve uma tentativa de seguir a especificação do problema. No entanto, no critério “Busca por códigos similares” e “integração entre os membros do grupo” as respostas foram quase igualmente distribuídas entre as três colunas, indicando que os respondentes não estavam muito confortáveis em lidar com esses critérios.

4.3.2. Análise Qualitativa

Na análise qualitativa desta pesquisa os objetos da pesquisa não foram reduzidos aos critérios do questionário; em vez disso, eles foram estudados como uma unidade em seu contexto de aplicação diário, o que já foi mencionado na descrição do estudo de caso. Assim como em toda pesquisa qualitativa, vale ressaltar que, de acordo com (Flick, 2004), as reflexões dos pesquisadores com respeito às suas ações e observações em seu campo de trabalho (sala de aula, neste caso), suas impressões, sentimentos e outras digressões se tornam dados por elas mesmas e constituem uma parte da interpretação.

As reflexões dos grupos e dos pesquisadores foram coletadas e transformadas em textos baseadas nos relatórios de desenvolvimento dos alunos, reunidas pelos grupos e anotações do observador/pesquisador. A documentação de dados não é simplesmente um registro neutro da realidade, mas um estágio essencial de sua construção no processo da pesquisa qualitativa. A interpretação dos dados é adaptada ou para a codificação e categorização ou para a análise de estruturas seqüenciais no texto.

A fim de se analisar os textos dos relatórios de desenvolvimento, dois aspectos foram considerados: as dificuldades encontradas pelos grupos e os sucessos relatados na conclusão. O procedimento metodológico utilizado foi o descrito em (Mayring, 1983), propondo três técnicas: a) abreviação da análise de conteúdo; b) análise explanatória do conteúdo, e c) análise estrutural do conteúdo. Para sintetizar essas técnicas, elevando-as a um nível mais alto de abstração, a redução do material (os textos dos relatórios) foi realizada através da omissão de declarações redundantes, conforme a primeira técnica propõe. As outras técnicas foram aplicadas em seguida e resultaram nos dados da Tabela 4.2, que descreve as dificuldades sentidas pelos grupos e na Tabela 4.3, que descreve as conclusões fornecidas pelos grupos, conforme descritas nos relatórios de desenvolvimento dos grupos.

Tabela 4.2 – Dificuldades sentidas pelos grupos

Informantes	Descrição das dificuldades sentidas pelos grupos
A	Interpretação da declaração de algumas questões
B	Transformação dos esboços de solução em código Haskell; reunião de todo o time; alcance de um consenso; agregação de todas as soluções.
C	Entendimento da sintaxe do Haskell; agregação de todas as soluções.
D	Entendimento de como se constrói um programa; construção adequada do programa; verificação de erros; utilização do interpretador Hugs.
E	Refinamento das soluções.
F	Implementação do código.
G	Utilização do interpretador Hugs; entendimento da sintaxe do Haskell.
H	Refinamento das soluções; utilização da recursão.
I	Interpretação da declaração de algumas questões; planejamento da solução; entendimento da sintaxe do Haskell.

Para clarificar textos difusos, ambíguos ou contraditórios envolvendo conteúdo relacionado ao contexto de aplicação, como, por exemplo, informação sobre autor, situações gerativas, etc, os pesquisadores utilizaram a técnica de análise explanatória de conteúdo. Baseados nessa análise, as percepções dos pesquisadores sobre as dificuldades sentidas pelos grupos são apresentadas na Tabela 4.4.

Tabela 4.3 – Conclusões fornecidas pelos grupos

Informantes	Descrição das conclusões fornecidas pelos grupos
A	Foi muito produtivo; nós colocamos nosso conhecimento em prática; nós aprendemos como trabalhar em grupo.
B	Ensinou-nos a trabalhar como um time, ajudando cada participante a amadurecer e aprender como lidar com as dificuldades e diferenças dos outros; houve um comprometimento do grupo em todas as atividades realizadas.
C	Não relatado
D	Não relatado
E	Demandou um trabalho conjunto do grupo; demandou uma conexão e, sobretudo consenso entre todos os membros do grupo; comunicação entre os membros do time foi mantida principalmente via Internet (Chat e e-mail).
F	Não relatado
G	Não relatado
H	Não relatado
I	Uma discussão direcionada do grupo foi necessária para se encontrar uma solução viável.

Considerando as conclusões relatadas pelos grupos, é possível formular a seguinte paráfrase explanatória: “é necessário se trabalhar em grupo e há uma demanda por compromisso, esforço e acordo dos participantes. As atividades propostas foram benéficas e representou uma boa oportunidade de pôr em prática o conhecimento em programação até então adquirido por todos”.

Os grupos experimentaram muitas dificuldades principalmente relacionadas à codificação em linguagem Haskell da solução proposta. Mesmo que o planejamento da solução tenha sido muito difícil para os alunos, já que esta habilidade (solução colaborativa de problemas) depende de coordenação e interação, as principais dificuldades relatadas foram as relacionadas às habilidades necessárias ao processo de codificação, utilização das técnicas de programação apropriadas e conhecimento da linguagem de programação específica. Isto resulta que aprendizagem de programação pode ser mais bem aproveitada quando

conduzida em grupo, desde que siga um modelo ou esquema que facilite este processo.

Tabela 4.4 – Percepções sobre as dificuldades sentidas pelos grupos

Informantes	Percepções dos pesquisadores
A	Interpretação da declaração de algumas questões.
B	Implementação do código; reunião de todo o time; alcance de um consenso; agregação de todas as soluções.
C	Implementação do código; agregação de todas as soluções.
D	Entendimento de como se constrói um programa; implementação do código; verificação de erros; utilização do interpretador Hugs.
E	Refinamento de soluções
F	Implementação do código
G	Utilização do interpretador Hugs; implementação do código;
H	Refinamento das soluções; utilização da recursão.
I	Documentação da atividade desenvolvida; planejamento da solução; implementação do código.

Finalmente, a maioria dos grupos relatou que a experiência de desenvolver uma atividade de programação em grupo foi positiva. Foi também observado que tão importante quanto a formação dos grupos é o desenvolvimento da própria tarefa, com a participação efetiva de cada membro do grupo.

4.4. 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); o uso de métodos colaborativos para representar o conhecimento sobre resolução de problemas (Mendonça *et al*, 2002) (Pereira *et al*, 2002) (Silva *et al*, 2002); a especificação de padrões de colaboração (Kobbe *et al*, 2007); e um estudo-piloto na 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 análise do estudo de caso descrito em (Almeida, Castro & Castro, 2006) resultou na concepção e desenvolvimento do “AAEP”, uma ferramenta desenvolvida no contexto de uma dissertação de mestrado para a organização e acompanhamento das soluções dos estudantes que permite, caso o professor considere necessário, a comparação entre 2 versões quaisquer da solução de um estudante. Essa ferramenta foi desenvolvida essencialmente para atender as demandas do professor, possibilitando a ele o gerenciamento dos registros de problemas pelos usuários e análises das soluções. Outras funcionalidades tais como a elaboração de código-fonte de programas, edição e teste associado a descrições para cada versão são acessíveis a professores e alunos.

O AAEP foi usado em sala de aulas para o apoio à programação em grupo, o que foi crucial para adaptar um modelo de colaboração para aprendizagem de programação em grupo envolvendo as seguintes ações:

- Verificar se os estudantes procuravam por código já escrito para problemas similares antes de tentar resolver um problema específico, o que evidenciaria um estreito relacionamento entre solução de problemas e programação baseada em exemplos;
- Fazer o planejamento do processo de resolução mais explícito através da organização e registro das versões de código produzidas;
- Verificar se os estudantes reutilizavam as versões anteriores de seus próprios códigos;
- Observar se a interação no grupo conduzia a soluções mais rápidas;
- Observar a incorporação da prática do indivíduo no grupo, com foco nas possíveis mudanças comportamentais;
- Analisar o comportamento do estudante dentro do grupo.

A Figura 4.1 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.

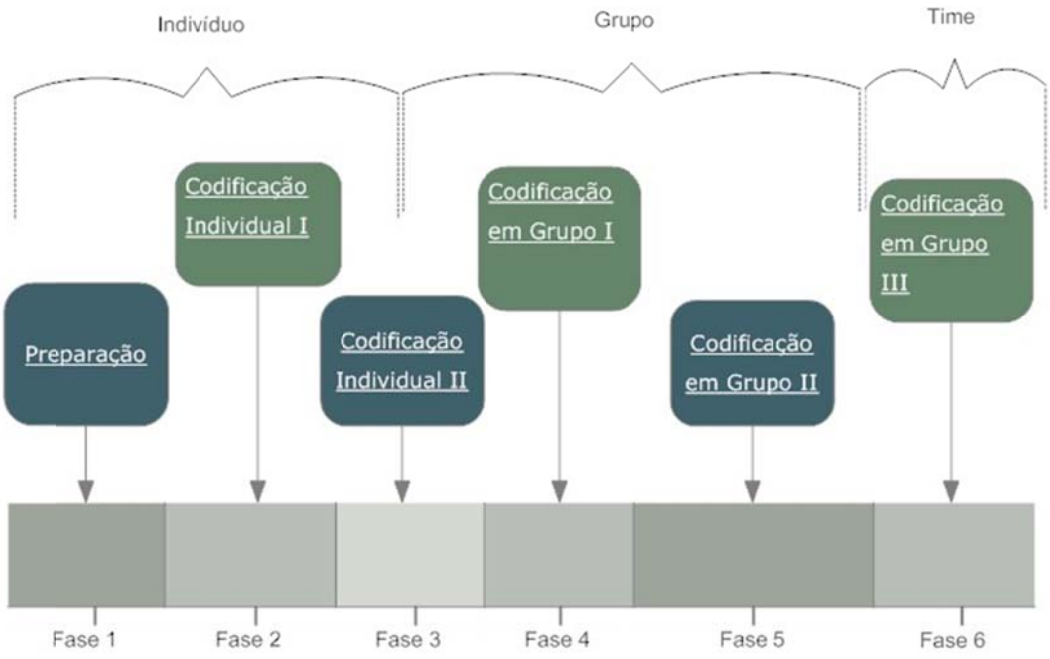

Figura 4.1 – Esquema Progressivo de Aprendizagem de Programação em Grupo

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.

Uma ferramenta de apoio à colaboração foi utilizada e incorporou todo o conteúdo criado antes e durante o curso, seguindo as diretivas encontradas em

(Fuks, Pimentel & Lucena, 2006), com atenção aos problemas descritos em (Pimentel, Fuks & Lucena, 2003). A Figura 4.2 descreve o planejamento das atividades práticas de acordo com as fases e semanas do curso. A seguir são apresentados amostras de exercícios por fase. Esses exercícios são oriundos do repositório de exercícios de programação da UFAM, com acesso restrito via colabweb.ufam.edu.br.

1. Preparação – *Survey* inicial onde estudantes respondem um questionário disponível no ambiente de apoio e resolvem problemas introdutórios, por exemplo: “Um grupo de quatro amigos deseja comprar um produto que custa mais dinheiro do que eles têm. De modo a realizar a compra, o grupo decide que o valor final deve ser dividido proporcionalmente ao valor que cada um já contribuiu, considerando que aquele que contribuiu com mais dinheiro poderia, em princípio, conseguir mais algum na mesma proporção. Descreva como você resolveria esse problema usando o Haskell, indicando quanto cada um deveria pagar.”
2. Codificação Individual I – Sessões de laboratório para resolver problemas geométricos básicos com análise das soluções baseada nos tempos de resolução. Exemplo: “Dados 2 pontos, p1 e p2, localize no espaço cartesiano uma linha. Determine a equação dessa linha.”
3. Codificação Individual II – Resolução de um exercício em Haskell com registro das anotações. Exemplo: “Dados 3 pontos, p1, p2 e p3, localizados no espaço cartesiano, determine se eles constituem um triângulo e, se for caso, determinar sua área.”
4. Codificação em Grupo I – Análise e seleção de soluções individuais, com anotações colaborativas sobre o processo decisório e os refinamentos identificados. Exemplo: “Considere 2 pontos, p1 e p2, localizados no espaço cartesiano. Estamos interessados em identificar entre os seguintes relacionamentos entre eles são aplicáveis: (a) se é possível traçar uma linha passando por p1 e p2 e em paralelo à linha das abscissas; (b) se é possível traçar uma linha passando por p1 e p2 e em paralelo ao eixo das ordenadas; (c) se p1 e p2 estão no mesmo quadrante; (d) se p1 e p2 estão em diferentes

quadrantes; (e) se p1 e p2 estão em quadrantes opostos; (f) se p1 e p2 estão em quadrantes adjacentes.”

5. Codificação em Grupo II – Resolução de 3 problemas, sendo que o primeiro é resolvido como na fase anterior, com membros do grupo revisando os códigos uns dos outros, fazendo anotações. No segundo problema ocorre a divisão de tarefas, realizada pelo professor. No terceiro problema a distribuição de tarefas é feita pelo grupo, que registra todo o processo decisório. Exemplo para uso como segundo ou terceiro problema: “Em uma clínica, tão logo um paciente chega, ele recebe um número de atendimento. Em cada turno há sempre três médicos, que atendem os pacientes dependendo do número de pacientes que cada um já tem em sua lista de atendimento. O médico que tiver menos pacientes na lista recebe o próximo paciente. Usando tuplas, pode-se definir a seguinte entrada: médicos (("dr. A", 4, 23), ("dr. B", 1, 13), ("dr. C", 3, 27)), onde o segundo termo de cada tupla refere-se ao número de pacientes na lista do médico e o terceiro termo se refere ao último paciente atendido por aquele médico. Baseado nessa entrada, escreva um script em Haskell que, para um dado paciente, escolha qual lista de atendimento ele será alocado.”
6. Codificação em Grupo III – Atividade no estilo de uma maratona de programação, consistindo de: observação externa por professor ou programador experiente; uso de ferramenta de monitoramento de atividades do grupo; e estágios com complexidade crescente. Exemplo de cenário: “Num banco de sangue há o registro de doação que inclui o CPF do doador (CPF), sexo (S), idade (I), tipo de sangue (TS), fator RH (RH), data (DD) e a quantidade de sangue doada (QS), que pode ser 250 ou 500ml. O sangue é mantido em recipientes com capacidade fixa de 250ml. Hospitais (H) solicitam sangue diariamente. Cada solicitação indica as características do sangue (tipo e fator RH) e a quantidade requisitada (QR). Sabe-se que homens e mulheres devem guardar intervalos mínimos entre doações, sendo 2 meses para mulheres e 3 meses para homens. As

idades máximas e mínimas para doadores são 60 e 15 anos respectivamente...”

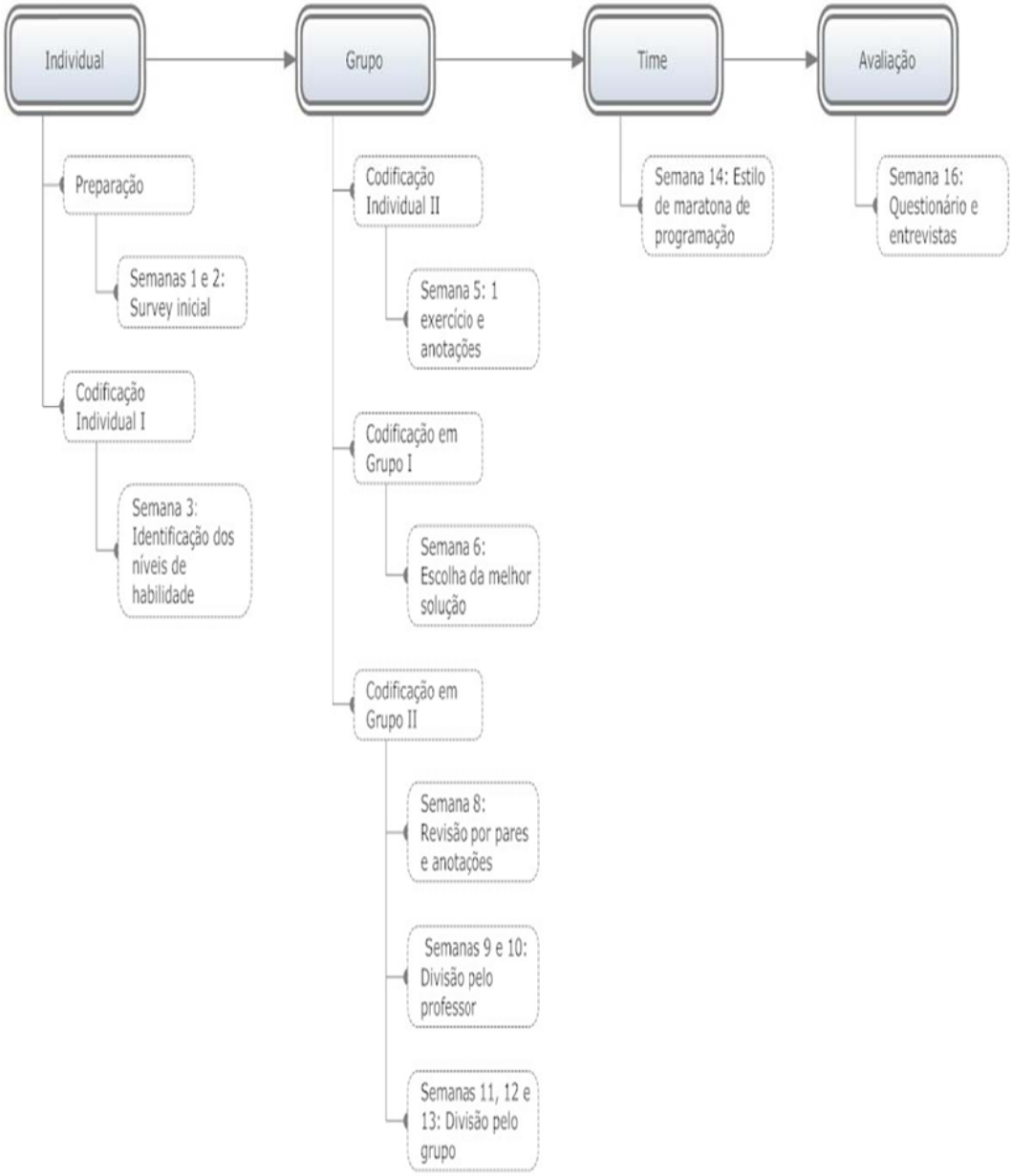

Figura 4.2 – *Workflow* do Esquema Progressivo de Aprendizagem de Programação em Grupo

Ainda com respeito ao esquema proposto, a modularização de problemas e a definição em Haskell são os conceitos envolvidos na fase de preparação. A fase de resolução individual requer o mesmo nível de *expertise*, em adição a um raciocínio mais matemático devido à natureza dos problemas geométricos. A fase seguinte, Codificação em Grupo I, requer o uso de expressões condicionais em

adição ao conhecimento sobre problemas geométricos. Na fase de Codificação em Grupo II, os estudantes usam tuplas e listas para resolver os problemas propostos. Por fim, na fase de Codificação em Grupo III, os estudantes precisam resolver problemas que envolvem tuplas, listas e recursão.

A partir da primeira fase de codificação em grupo, os estudantes trabalham nos grupos que são formados na terceira semana de práticas, após a aplicação de uma sessão supervisionada de laboratório. O requisito adotado é que cada grupo seja o mais heterogêneo possível, com um mínimo de 5 membros. Não mais que 2 devem ter experiência formal em programação, 1 pode ter alguma experiência (por exemplo programação de scripts Web) e 2 a 3 não possuem experiência em programação.

Conforme mostrado nos exemplos de exercícios nessa seção, após a formação dos grupos, os exercícios aumentam em complexidade de acordo com o esquema progressivo e o conteúdo do curso. Isso possibilita o avanço gradual de um ritmo de trabalho baseado em práticas individuais para a incorporação de práticas colaborativas no desenvolvimento de programas.

4.5. Conclusão do Capítulo

Neste capítulo apresentamos o trabalho em grupo como uma maneira de se desenvolver habilidades cognitivas. Como em muitos ambientes de trabalho na área de computação, as habilidades relacionadas à colaboração são requisitos necessários aos desenvolvedores e gerentes, essa é uma prática que os alunos precisam ser expostos desde a graduação.

Para incorporar colaboração nas práticas da disciplina introdutória destacamos que é necessária, além de planejamento, a adoção de mecanismos de estruturação das interações. Um mecanismo muito utilizado em aprendizagem colaborativa é script de colaboração. Discutimos como eles podem ser utilizados para estruturas as conversas durante a realização de atividades em grupo, apresentando também as desvantagens em utilizá-los.

Independentemente da utilização de scripts é necessário se conhecer como os alunos, intuitivamente, se organizam em seus grupos para resolverem exercícios de programação. Por isso, na Seção 4.3 apresentamos um relato

baseado em (Castro et al, 2008) que descreve a análise de um estudo de caso exploratório.

No próximo capítulo discutimos a aplicação e análise de dois estudos de caso. O primeiro para aplicar o esquema progressivo de aprendizagem de programação em grupo descrito na Seção 4.4 deste capítulo. O outro para confirmar as categorias para análise das conversas identificadas no primeiro estudo de caso, chamadas de padrões de interação. Os três estudos de caso, o apresentado neste capítulo e os do próximo capítulo, embasam a sistematização da abordagem sistematizada para aprendizagem de programação em grupo.