Difficulties of Programming Learning from the Point of View of Students and Instructors

Y. Bosse, M. A. Gerosa

***Abstract*— Computer programming courses are mandatory for many majors. However, the high rate of failures shows that students have difficulties in assimilating the topics. The objective of this research is to understand these difficulties. Analyzing diaries filled out by students and interviews with instructors, we identified difficulties related to language and understanding and some strategies used to mitigate them. The analysis and understanding of the difficulties may support the creation of teaching strategies and tools to facilitate the teaching and learning of computer programming.**

***Keywords*— Difficulties, barriers, programming learning, novices, introduction to programming, computational thinking.**

I. INTRODUÇÃO

ENSINAR programação de computadores para as novas gerações tem sido um desafio que está sendo apoiado por governos e grandes empresas do mundo, promovendo iniciativas como as da Australia, EUA, Brasil, Reino Unido e Code.Org, com apoio de várias empresas, entre elas o Facebook e a Microsoft. Entretanto, a área ainda possui muitos problemas em aberto. Iniciantes em programação são tipicamente limitados ao conhecimento superficial [13]. Aprendem a sintaxe e semântica dos comandos individuais, mas não sabem como combiná-los em um programa [24]. Nas universidades, pesquisas mostram altos índices de reprovações nas disciplinas de introdução à programação [2, 4]. Por outro lado, muitos estudos vêm sendo realizados para melhorar e facilitar o aprendizado dos conceitos abordados na disciplina [6, 10, 18, 21]. Entretanto, ainda falta um entendimento sistematizado sobre as dificuldades específicas que os aprendizes enfrentam. Consequentemente, muitos trabalhos são desenvolvidos com base em evidências anedóticas e falta fundamentação para os professores planejarem as estratégias de ensino.

O objetivo principal desta pesquisa é identificar cientificamente dificuldades enfrentadas pelos aprendizes de programação sob o ponto de vista do aprendiz e do professor. Por se tratar de um fenômeno complexo e subjetivo, adotamos uma abordagem qualitativa baseada em diários no caso dos aprendizes, de modo que eles relatassem ao longo do curso as dificuldades e as tentativas de soluções. Com os professores foram feitas entrevistas semiestruturadas de modo a explorar as nuanças do processo. As duas questões de pesquisa que nortearam nossos estudos foram:

- QP1 – Quais as principais dificuldades encontradas pelos alunos no aprendizado de programação?
- QP2 – Quais os recursos utilizados por alunos e professores para amenizar as dificuldades detectadas por eles?

O artigo está organizado em seis seções. A seção seguinte apresenta os trabalhos relacionados e o diferencial deste trabalho. Em seguida, na Seção III, é descrita a metodologia utilizada na nossa investigação e na Seção IV, os resultados e discussões. Na Seção V é comentado sobre as limitações encontradas e possíveis ameaças à validade, seguida da conclusão, na Seção VI.

II. TRABALHOS RELACIONADOS

Há alguns poucos estudos para entender as dificuldades enfrentadas pelos aprendizes de programação de computadores. Cechinel et al. (2008), por exemplo, relataram que as dificuldades mais comuns são a falta de capacidade de encontrar erros, conseguir desenvolver um programa para resolver uma tarefa e modularização do código utilizando funções e procedimentos [5]. Os assuntos considerados mais difíceis foram funções e procedimentos, manipulação de erros e arrays (vetores) [5]. Beaubouef e Mason relatam que os motivos das altas taxas de desistência dos cursos de computação são devidas ao conhecimento equivocado do que realmente é estudado no curso, à necessidade de habilidades matemáticas e de resoluções de problemas; além de pouco uso de laboratórios para desenvolvimento de habilidades [1]. A falta de *feedback* dado aos alunos também é considerada pelos autores como um dos problemas pelas taxas de desistências. Esse *feedback* demanda muito tempo ou o uso de um sistema de correção automático.

Denny et al. [7] analisaram submissões de exercícios curtos feitos em Java por alunos da Universidade de Auckland, na Nova Zelândia, com o objetivo de verificar o quão frequente eram os erros de sintaxe, se tinham relação com o desempenho no curso e o quanto os erros de sintaxe impediam os alunos de resolver os exercícios e de receberem o *feedback* de suas lógicas nas resoluções dos problemas. Como resultado obtiveram que a sintaxe é uma barreira ao estudante [7]. Mesmo em problemas simples, os estudantes tinham dificuldade na primeira submissão. Eram várias submissões incorretas antes de conseguir compilar o código e assim ter um retorno se a lógica estava correta. Ribeiro et al. [16] investigaram em sua pesquisa as diferenças entre o uso de ambientes textuais e programação visual na introdução de programação de computadores. Após a coleta de dados, resolução das atividades e um questionário, chegaram à conclusão que a programação visual é um bom modelo para ensino de algoritmos e programação.

Y. Bosse, Federal University of Mato Grosso do Sul, Ponta Porã, Brazil, yorah.bosse@ufms.br

M. A. Gerosa, University od São Paulo, São Paulo, Brazil, gerosa@ime.usp.br

(Corresponding author: *Y. Bosse*)

Entender o processo de aprender a primeira linguagem de programação pode ajudar na tarefa de desenvolver ambiente de aprendizado mais efetivos [8], reduzindo assim as dificuldades encontradas pelos aprendizes de programação.

Lahtinen et al. [13] conduziram um survey em seis universidades distribuídas em cinco países. Eles obtiveram a resposta de 559 estudantes e 34 professores. As respostas foram dadas em uma escala de 1, fácil de aprender, até 5, muito difícil. Quanto ao conteúdo educacional abordado no curso, a percepção média do estudante sobre o quão difícil é o curso (média de 2,8) é menor do que a dos professores (média de 3,5). Estudantes e instrutores tiveram a mesma percepção dos três conteúdos considerados os mais difíceis. São eles, nessa ordem: ponteiros, tratamento de erros e recursão. Além desses, foi considerado difícil usar a biblioteca da linguagem de programação e tipos abstratos de dados. Ambos, estudantes e professores, tiveram também a mesma percepção dos três conteúdos mais fáceis: seleção, repetição e variáveis. Contudo, aprender os conceitos não foi considerado por eles o maior problema enfrentado pelos aprendizes, mas sim aplicá-los na prática.

O uso de diários, método utilizado na nossa pesquisa, já foi adotado por vários pesquisadores para estudar assuntos de como avaliar software [19]; verificar a influência do uso do Facebook sobre o narcisismo [22]; entender e comparar o uso de tablet e smartphone [14]; e verificar o contexto do uso de aparelhos de computação pessoais [12].

Diferentemente dos estudos da literatura, esta pesquisa realizou um estudo qualitativo baseado em entrevistas e diários, considerando e comparando o ponto de vista dos docentes e estudantes.

III. METODOLOGIA

Neste estudo, realizamos uma pesquisa qualitativa com base em informações relatadas pelos alunos em diários preenchidos durante seus estudos e informações fornecidas pelos professores durante entrevista semiestruturada (Fig. 1). O estudo foi realizado no campus principal da Universidade de São Paulo - USP, durante o segundo semestre de 2015 e teve como público alvo estudantes da disciplina de Introdução à Programação de cursos que não são da área afim de computação. Foram convidados a participar os alunos das seis turmas cujas aulas estavam sendo ministradas por professores do Instituto de Matemática e Estatística (IME) da USP. Os alunos que aceitaram participar da pesquisa assinaram um Termo de Consentimento, autorizando que os dados fornecidos por eles fossem usados na pesquisa. Os professores dessas turmas foram entrevistados.

Para coleta das dificuldades enfrentadas pelos alunos foi escolhido o método Diário. O motivo da escolha pelos Diários é porque o método traz como benefício a possibilidade de examinar eventos e experiências na sua forma natural e espontânea [15]. Bolger et al. [3] acrescentam que os diários encurtam o tempo da experiência e do momento em que ela é relatada. Assim, espera-se que a perda de informações por esquecimento seja diminuída e a riqueza de detalhes aumentada [23].

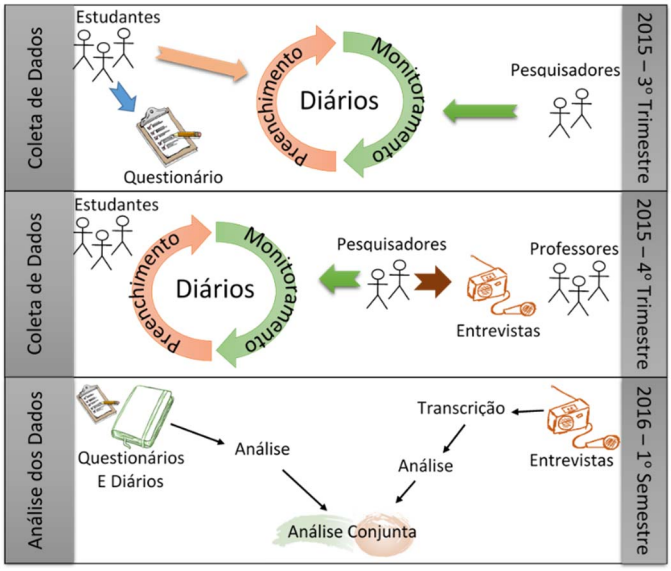

Figura 1. Esquema dos passos da metodologia adotada.

O editor de texto online Google Docs foi usado como ambiente para a escrita dos diários. Cada aluno tinha acesso a um documento compartilhado com os autores deste artigo. Nesse documento foi preparada uma abertura com orientações sobre como deveria ser o preenchimento do diário e um breve questionário para traçar o perfil dos participantes. Os históricos escolares dos alunos participantes também foram coletados.

A área dos Diários era de preenchimento livre, sem perguntas específicas, porém com orientações que os participantes preenchessem com dados referentes aos seus sentimentos sobre o aprendizado, dificuldades encontradas e formas que utilizavam para resolver os problemas encontrados. Foi pedido também que colocassem códigos desenvolvidos por eles durante os estudos e que o diário fosse preenchido no momento em que estivessem estudando programação.

O passo seguinte da pesquisa foi acompanhar o preenchimento dos diários. Esse acompanhamento se dava na forma de orientações, estímulos e perguntas feitas por meio de mensagens deixadas nos próprios diários. Em seguida foram analisados os dados coletados e feita a formatação dos resultados. A análise qualitativa dos dados dos diários foi realizada utilizando procedimentos de *Grounded Theory*, de acordo com conceitos e técnicas descritas por Strauss e Corbin [20]. Esse método de análise baseia-se em codificar os dados, identificando conceitos e categorias. Um conceito é o nome dado a um fenômeno, identificado nos dados, com um significado de interesse do pesquisador. As categorias são junções desses conceitos. Nesta pesquisa, foram criadas quatro categorias, conforme apresentado na seção a seguir. Para identificar o aluno que escreveu comentários reproduzidos neste artigo, utilizamos um "a" subscrito seguido de uma numeração.

A partir do último mês de aula, os professores que lecionaram a disciplina para os alunos foram entrevistados, utilizando o formato de entrevista semiestruturada. A entrevista foi gravada, transcrita e analisada pelo método de codificação

aberta. O entrevistado foi guiado a comentar sobre as dificuldades enfrentadas pelos alunos nos conteúdos abordados na disciplina. Também foi questionado sobre estratégias utilizadas para ajudar a superar tais dificuldades. Como nomenclatura para identificar cada professor foi utilizado um "p" subscrito seguido de uma numeração de 1 a 6.

IV. RESULTADOS E DISCUSSÃO

Os resultados são oriundos de quatro fontes distintas de coleta de dados: questionário, histórico escolar, diários e entrevistas. Foram 6 professores entrevistados e 24% dos 144 alunos que assinaram o termo de consentimento participaram respondendo o questionário e preenchendo o diário. As turmas tinham uma média de 53 alunos. As 2 turmas que usavam a linguagem de programação Python, tiveram uma média de reprovação de 27,5% e média das notas finais de 5,8. Nas 4 turmas que usaram C como linguagem de programação, a reprovação foi maior (54,9%) e a média final foi menor (3,9).

A. Perfil dos alunos e das turmas

No início dos diários, foi pedido para que cada aluno respondesse um questionário de 7 perguntas. Responderam esse questionário 47 alunos, porém apenas as respostas dos que preencheram o diário foram consideradas, totalizando 34 respondentes. A faixa etária dos respondentes varia de 17 a 31 anos, sendo 59% com até 20 anos. Dos respondentes, 50% estudaram em escolas privadas, 32% em escolas públicas e 18% em ambas (Fig. 2A). Em relação à experiência anterior, 76% ainda não sabiam programar (Fig. 2B). Além disso, 41% afirmaram trabalhar, e desses, 50% trabalham de 6 a 8 horas diárias (Fig. 3A). Quanto a já possuir outro estudo superior, 18% já tinham iniciado um outro curso de graduação, porém não tinham se formado, ou seja, nenhum aluno era graduado (Fig. 3B).

Por meio do histórico dos alunos respondentes, verificamos que 30% já tinham cursado a disciplina de IP anteriormente. Considerando todas as disciplinas cursadas por esses alunos no segundo semestre de 2015, foram 161 matrículas, média de quase 5 disciplinas cursadas por aluno, com um resultado de 20% de reprovações ou trancamento. Já considerando apenas a disciplina de IP, esse percentual de reprovações e trancamento sobe para 29,41%. Foram 10 reprovações para 34 matrículas em IP, cursadas por alunos de 6 cursos distintos (TABELA I). Das matérias cursadas, a média das notas mais baixas tiradas pelos alunos é 2,4, sendo que 12 alunos tiraram a nota mais baixa na disciplina de Cálculo Integral e Diferencial II e 8 em Introdução à Programação.

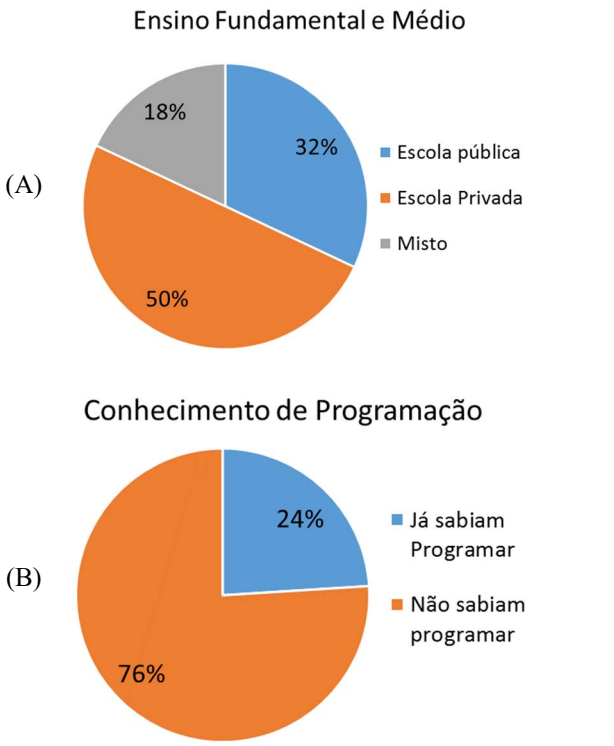

Figura 2. (A) Ensino fundamental e médio dos alunos participantes do questionário. (B) Conhecimento em desenvolvimento de programas.

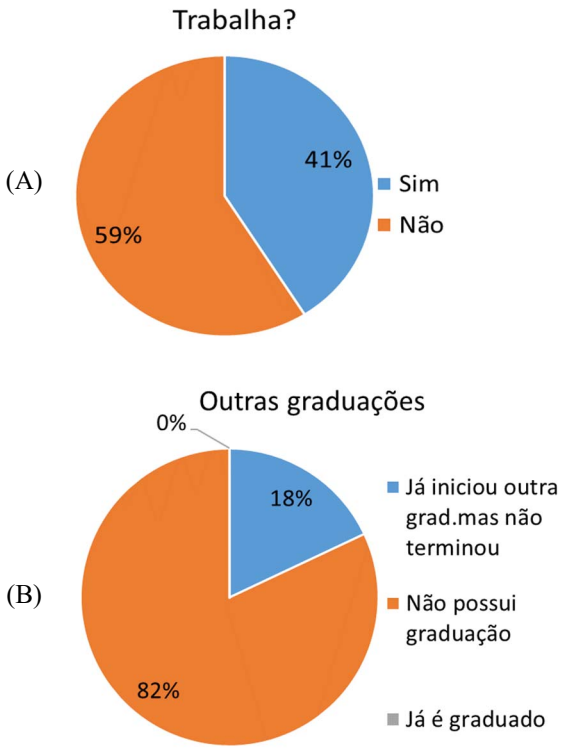

Figura 3. (A) Relação entre os alunos que trabalham e os que não trabalham. (B) Relação entre graduações anteriores concluídas ou iniciadas.

TABELA I
CURSOS FREQUENTADOS PELOS ALUNOS QUE PREENCHERAM O DIÁRIO E O QUESTIONÁRIO

Nome do Curso	Quantidade de Alunos
Bacharelado – Física	16
Bacharelado – Oceanografia	6
Licenciatura – Matemática	5
Bacharelado – Geofísica	3
Licenciatura – Física	2
Bacharelado – Meteorologia	2
	34

Quase 30% dos alunos já tinham cursado a disciplina de IP anteriormente (TABELA II).

TABELA II
QUANTIDADE DE VEZES QUE OS ALUNOS CURSARAM A DISCIPLINA DE INTRODUÇÃO À PROGRAMAÇÃO ANTERIORMENTE

Quantidade de Vezes	Quantidade de Alunos	Percentual
4	1	2.9%
3	3	8.8%
2	2	5.9%
1	4	11.8%
0	24	70.6%
	34	**100.0%**

No semestre, havia turmas que adotavam C e Python e as ferramentas Code::Blocks, Dev-C++ e IDLE. Um dos professores utilizou nas suas aulas o iVProg e o VPL. O iVProg é um sistema desenvolvido na USP que utiliza o conceito de programação visual e pseudocódigo [11]. Ele era usado nas aulas como apoio ao desenvolvimento da lógica para resolver os exercícios propostos. Com essa lógica desenvolvida, era feita a tradução para C. Para isso era usado o Virtual Programming Lab – VPL [17], que é um *plugin* para o Moodle que oferece o compilador para o aluno digitar e testar seu código. Além disso, o VPL e o iVProg utilizam casos de testes inseridos pelo professor para dar *feedback* imediato ao aluno quanto à corretude da lógica.

Como conclusão, observamos que na sua maioria os participantes são jovens e sem experiência com programação. Muitos trabalham e o fato de quase 30% já ter cursado a disciplina anteriormente não obtendo sucesso dá uma ideia da dificuldade em aprender o conteúdo abordado na disciplina.

B. Diários preenchidos pelos alunos

Utilizando o método Grounded Theory [20] para a análise qualitativa dos dados fornecidos pelos alunos nos diários, os conceitos foram organizados em categorias e o resultado dessa categorização pode ser vista na Fig. 4. Nessa figura são mostradas as quatro categorias criadas: Dificuldades, Técnicas para Estudo, Preferências e Autoavaliações. Cada uma delas é detalhada nas subseções a seguir.

Categoria: Dificuldades

A primeira categoria foi a de Dificuldades no processo de aprender programação (TABELA III). 'Erro de sintaxe', com 13 ocorrências, foi o problema mais citado pelos alunos, com comentários como: "*ainda erro muito nas coisas básicas, como colocação de chaves, parênteses, ponto e vírgula*"$_{a1}$, "*o programa ainda não rodava por conta de alguns erros de sintaxe que eu não soube resolver na hora*"$_{a20}$ e "*está dando*

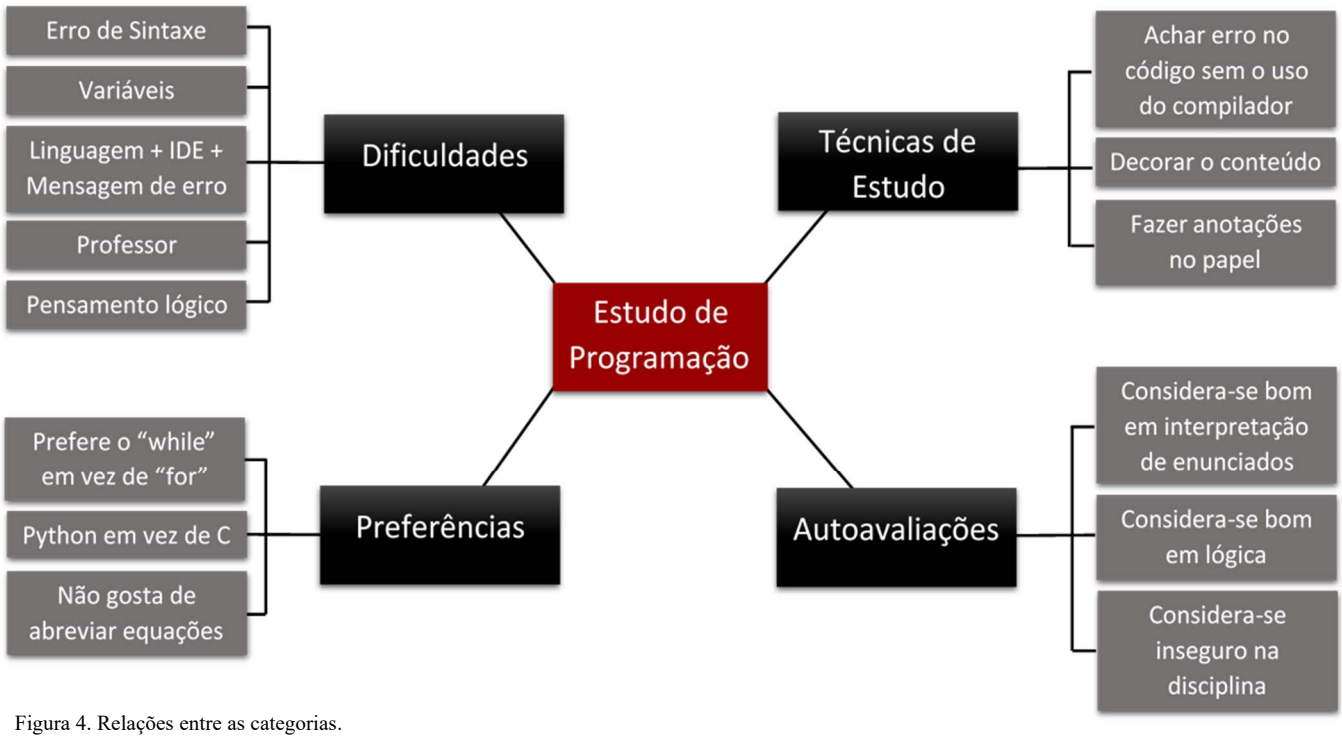

Figura 4. Relações entre as categorias.

erro de sintaxe toda hora"$_{a22}$. Esse tipo de erro faz com que o aluno tenha que voltar muitas vezes ao código antes de poder verificar se o programa está com a lógica correta e realmente resolve o problema proposto.

Problemas com 'variáveis', como declaração, utilização, inicialização e tipagem, foi o segundo mais recorrente, como observado no comentário "*tive dificuldade em perceber o que deveria ser float e o que deveria ser int, então tive que ir testando até achar*"$_{a1}$. O conceito 'Linguagem + IDE + Mensagem de erro' também foi bastante citado, tendo reclamações como: "*inicialmente tive dificuldade com a linguagem, mesmo tendo apoio do caderno e apostila, tive dificuldade em colocar em prática*"$_{a5}$, "*porque as mensagens do programa não ajudaram em nada*"$_{a20}$ e "*não consegui interpretar as mensagens que o programa mostrava então tive que executar partes do programa separadamente em outra janela até conseguir identificar o erro*"$_{a20}$. Além dessas reclamações sobre a linguagem e as mensagens de erro, recebemos comentários também sobre o ambiente de trabalho utilizado, como "*o site do professor da matéria não possui o link para baixar a versão atualizada do programa, e a versão de lá não funciona no Windows 8*"$_{a17}$. A TABELA III mostra os conceitos que compõe a categoria Dificuldade e a quantidade de vezes que cada um deles foi citado.

As três categorias seguintes foram criadas a partir de comentários espontâneos e não solicitados aos alunos, porém considerados importantes durante a realização da análise.

Categoria: Técnicas de Estudo

No momento que os alunos foram revelando as dificuldades encontradas, de modo espontâneo iam informando as técnicas que utilizavam para suprir essas dificuldades, formando assim a segunda categoria – Técnicas para Estudo (TABELA IV). A técnica mais citada foi a de 'tentar achar erros sem o compilador', "*tentei montar os códigos no papel primeiro, ir fazendo simulações e encontrando os erros sozinha*"$_{a1,}$ tentando evitar assim erros de sintaxe, o conceito mais citado na categoria Dificuldades. Conceitos como 'decorar' e 'repetir mecanicamente' também foram citados como Técnicas de Estudos, "*minha abordagem tem sido decorar e repetir mecanicamente o que é apresentado, pois ainda não entendi os porquês da programação*"$_{a1}$ e "*só vou alterando o código por tentativa e erro*"$_{a3}$ dão a entender que o aluno ainda se encontra em um alto nível de dificuldade, não tendo condições de criar a solução a partir dos conhecimentos adquiridos. 'Trocar conhecimento com colegas' também foi considerado um conceito dentro da categoria Técnicas de Estudo, sendo relatado que "*quando tinha dúvidas, perguntava a um colega que também estava fazendo o EP (e vice-versa), e achei essa comunicação fundamental para a execução da tarefa*"$_{a20}$, mostrando a importância da comunicação e trocas de experiências entre eles.

Relatos mostram também que há alunos cientes de que alguns conceitos dessa categoria podem causar problemas quando usados no processo de aprendizado de programação. Eles mencionaram que "*tentativa e erro para um programa sendo feito no computador acredito não ser muito problema, mas quando se está escrevendo no papel isso pode me trazer problemas*"$_{a3}$, "*não adianta decorar, porque cada exercício é um*"$_{a15}$ e "*basta um pouco de ansiedade para apagar todas as decorebas da memória*"$_{a1}$ mostrando que os conceitos 'tentativa e erro' e 'decorar' podem trazer problemas futuros nos estudos. Além desses, também foi mostrada preocupação quanto ao não uso de papel para esboçar a lógica do programa, como ilustrado no seguinte comentário "*não consigo escrever (no papel) a lógica do programa, pois me confunde, mas em alguns momentos percebo que deveria treinar essa parte, para me ajudar na execução*"$_{a6}$.

TABELA III
CATEGORIA DIFICULDADES, QUE REPORTA AS DIFICULDADES APRESENTADAS PELOS ALUNOS NOS DIÁRIOS PREENCHIDOS DURANTE OS ESTUDOS

CATEGORIA: DIFICULDADES	
Qtd	**Conceitos**
13	Erro de sintaxe
10	Variáveis
8	Linguagem + IDE + Mensagem de erro
7	Professor
6	Falta pensamento lógico
6	Estruturas Homogêneas (matrizes e vetores)
5	Indentação
5	Operadores
5	Interpretação
4	Repetição
4	Matemática
4	Funções
3	Seleção
2	Atribuição
2	Localização dos elementos dentro do código
1	Apreensivos com os novos termos
1	Diferenciar entrada de saída de dados
1	Fazer teste de mesa

TABELA IV
CATEGORIA TÉCNICAS PARA ESTUDOS ADOTADAS PELOS ALUNOS

CATEGORIA: TÉCNICAS PARA ESTUDO	
Qtd	**Conceitos**
3	Tentar achar erros sem o compilador
2	Decorar
2	Usar papel para anotações
1	Repetir mecanicamente
1	Tentativa e erro
1	Não adianta decorar (Tenta entender)
1	Trocar conhecimento com colegas
1	Praticar

Categoria: Preferências

A terceira categoria – Preferências - foi criada com os

comentários sobre as preferências dos alunos quanto às linguagens, forma de programar e estilo de avaliação (TABELA V). Frases sobre comandos "*acho muito mais fácil e didático o uso do laço while com um contador interno ao invés do for*"$_{a21}$ e "*prefiro usar o 'while' do que o 'for' e 'n = n + 1' do que 'n ++ '*"$_{a1}$; forma de programar "e*u costumo programar no celular os exercícios que o professor faz*"$_{a21}$ e linguagem "*entendia mais os programas em Python do que em C. Esta linguagem parece muito mais 'intuitiva'*"$_{a29}$, chamaram nossa atenção ao ler os diários.

TABELA V
CATEGORIA PREFERÊNCIA

CATEGORIA: PREFERÊNCIAS	
Qtd	**Conceitos**
2	While em vez de For
1	Não abreviar equações
1	Python ao invés de C
1	Provas no computador ao invés de fazer no papel
1	Programar no celular

O comentário "*Acredito que as provas de computação deveriam ser feitas em computadores, para que possamos nos atentar mais à lógica do que à sintaxe e também receber um feedback instantâneo do que estamos fazendo...*"$_{a20}$ foi feito por um aluno repetente e que estava cursando a disciplina com as aulas feitas todas no laboratório, tendo o VPL como apoio de feedback imediato.

Categoria: Autoavaliações

A quarta e última categoria tirada dos diários é a de autoavaliações. Eles se avaliam espontaneamente, revelando situações ou características que possam estar ajudando ou atrapalhando seu desenvolvimento nos estudos. Alguns comentários expõem atitudes como a 'falta de estudo' - "*acredito que toda dificuldade que tive foi devido à falta de preparo e estudo adequados*"$_{a8}$ e características próprias como a de ser 'bom em lógica' - "*a vantagem que tenho é que sou boa em lógica*"$_{a1}$ e "*como já tenho prática com a lógica...*"$_{a21}$. A TABELA VI mostra a relação dos conceitos criados nessa categoria.

Concluindo, é importante que o professor observe e oriente os alunos quanto aos métodos de estudo adotados e materiais de estudos utilizados. Gomes e Mendes [9], verificaram que na opinião dos professores entrevistados, os estudantes não têm habilidades de estudo, organização e hábitos mínimos de trabalho. Conforme observado na Tabela IV, podemos verificar essa falta de habilidade, como é o caso da escolha por decorar. Resolver tarefas com programação se assemelha muito com resolver problemas práticos na matemática, os conceitos são aplicados e usados dependendo do que se pede, não existindo um roteiro pré-definido. Quanto às dificuldades encontradas, muitas delas podem ser amenizadas com a utilização de didáticas como trabalhos em duplas na sala de aula, exercícios já resolvidos estudados com os alunos em sala de aula, exercícios em ordem crescente de dificuldades sendo resolvidos em sala pelos alunos com posterior discussão e correção entre aprendizes e professor, entre outras. Essas técnicas com muita prática, discussão e troca de informações ajudariam também em aspectos como insegurança, citado na Tabela VI, pois o aluno poderia entender melhor seus erros.

TABELA VI
CATEGORIA AUTOAVALIAÇÕES

CATEGORIA: AUTOAVALIAÇÕES	
Qtd	**Conceitos**
2	Bom em lógica
1	Facilidade em interpretar enunciado
1	Insegurança com a disciplina
1	Começa a resolver antes de entender
1	Falta de estudo
1	Dificuldade em mexer com computador

C. Entrevistas feitas pelos professores

O ponto fundamental das dificuldades citadas pelos professores é quanto ao 'raciocínio lógico'. Ele é essencial para o desenvolvimento dos algoritmos. Estratégia como a utilização de pseudocódigos foi considerada falha pela maioria dos professores. Segundo eles, a transição do pseudocódigo para a LP é bastante difícil. Alguns usam o pseudocódigo apenas para explicar rapidamente o conceito e já vão direto para LP. Alguns professores usam o pseudocódigo paralelamente, ou seja, fazem no pseudocódigo e em seguida traduzem para a LP, *"...se você não sabe por onde começar, escreve em português um rascunho. Do português vai para o pseudocódigo e só no final você vai para o Python"*$_{p3}$. Outra informação importante é que para professores com 'muita experiência' situações como a do seguinte comentário acabam acontecendo: *"eu vejo um problema, ele já se estrutura na minha mente e eu não sei como é que isso acontece"*$_{p1}$, podendo assim, ter dificuldade em lecionar a disciplina e ajudar o aluno a desenvolver o raciocínio lógico.

Quanto às 'questões sintática da linguagem', foi bastante comentado que o C possui muito mais detalhes de sintaxe a serem observados durante a programação: *"o C tem muitas questões sintáticas complicadas"*$_{p2}$. Já o Python é uma linguagem em que, mesmo sendo orientada a objetos, o professor consegue 'esconder o paradigma', facilitando assim o ensinamento da lógica: *"...programação orientada a objetos, eu não dei"*$_{p3}$. Além do C e Python utilizadas no semestre, outras LP já foram utilizadas pelos professores como Pascal, Java, Fortran, Algol e VBA. Foi comentado também que a 'escolha da linguagem' influencia no desenvolvimento do aluno. Há linguagens mais simples para ensinar, como o Python; as que se adaptam melhor às necessidades do aluno de um determinado curso, como o VBA para as turmas de Administração; ou por já existirem muitos pacotes de códigos prontos, como o C para a Física.

Os operadores – aritméticos, lógicos e relacionais – também são fontes de dificuldades. Os alunos confundem a precedência deles. Um professor explicou que tenta ajudá-los falando *"se*

você tá vendo que vai se confundir, não fica pensando (...) põe um parêntese a mais ali que não vai ter problema"$_{p1}$. Há dificuldade também em diferenciar os operadores lógicos 'e' e 'ou' e de fazer operações aritméticas com tipos de valores iguais, porém com o resultado tendo que ser de tipo diferente. Um exemplo é na divisão de tipos inteiros e o resultado, para ter valor, precisa ser do tipo real, senão dá zero, aí *"é difícil eles perceberem o erro"*$_{p4}$.

Entre as estruturas de seleção/decisão e repetição/laço, a maioria dos professores começa lecionando pelo laço, mais especificamente pelo *while*. Foi muito citado que o *while* dá a impressão de mais controle da estrutura, que o aluno 'prefere o *while* em vez do *for'*, informação que corroborou com as dos diários escritos pelos alunos. Uma estratégia didática para ensinar o *for* é 'escrever a estrutura também no while para comparação'. O aluno também tem dificuldade em laços encaixados e de como definir a condição de parada. Na estrutura de seleção, a dificuldade está em 'enxergar os pares *if..else*' e há trocas de conceitos, confundindo seleção e repetição.

Para vetores e matrizes, existe o 'esquecimento de colocar o índice' referente a posição, que é resolvido com a estratégia de 'prática intensiva': *"tem que fazer por repetição mesmo, que é um negócio um pouco enjoativo no começo, até pegar o jeito"*$_{p1}$. Os professores também comentam que os alunos entendem o conceito, mas não conseguem aplicar na prática, informação que veio reforçar o que já foi publicado [24].

Sobre função, a dificuldade está em entender o escopo das variáveis e a importância do retorno de valor. Os professores acreditam não ter problemas grandes com passagem de parâmetro por valor, porém, quando é por referência e a linguagem utilizada é o C, os conceitos de ponteiros envolvidos fazem o aluno ter muita dificuldade. Alguns professores utilizam aqui a estratégia de 'ocultar conceitos' para tentar fazer o aluno aprender o conteúdo principal que é a passagem de parâmetro.

Os professores comentam que existem alguns fatores que dificultam lecionar a matéria, como: heterogeneidade das turmas e entre turmas, baixa participação nas aulas, baixa frequência, turmas muito grandes, desinteresse em aprender, a sala adotada, traumas de alunos repetentes na matéria, entre outras. Eles também se mostraram preocupados em tentar motivar o aluno, para isso eles utilizam estratégias como trabalhar com jogos, desafiar e gerar competições em turmas mais fortes, fazer simulados antes das provas e adaptar as aulas para cursos específicos, levando exercícios parecidos com o que os alunos encontrarão em sua vida profissional. Aluno desmotivado, sem tempo ou com alguma dificuldade acaba usando estratégias nada adequadas como o plágio – *"..mas eu peguei também alunos que copiaram da internet, aí tem que ter esse cuidado, não sei se deixaram pra última hora..."*$_{p5}$ – e decorar em vez de aprender – *"alguns alunos deixaram escapar a verdade, ... , que é 'ah, eu vi uma solução semelhante a esta mas não consegui me lembrar', ou seja, o cara tenta decorar."*$_{p4}$. Decorar também foi citado pelos alunos nos diários.

Concluindo, algumas informações dadas pelos professores corroboraram com aquelas retiradas dos diários, como por exemplo, a preferência pela estrutura de repetição "while" em vez de "for". Fica o questionamento se o aluno prefere o "while" porque dá mais controle da estrutura ou se deve ao fato dele ter aprendido antes do "for". Outra situação percebida por professores e informada pelos alunos é quanto ao decorar o conteúdo em vez de entender. Esse é um ponto que mostra a dificuldade do aluno em aprender e aplicar os conceitos. O professor por sua vez não consegue detectar e ajudar esses alunos por vários motivos, como é o caso das salas muito cheias. Questões sintáticas da linguagem foi o principal item citado pelos alunos, revelando a importância da escolha da linguagem para se poder focar mais no conteúdo. Um dos pontos de maior dificuldade citados pelos professores foi ensinar com sucesso "ponteiros", o que seria importante para entender e usar funções, por exemplo. Alguns erros básicos como esquecer índice ao se trabalhar com listas/vetores ou matrizes também é uma dificuldade citada por ambos. O fato de muitas dificuldades serem percebidas tanto por alunos como pelos professores reforça a percepção da falta de estratégias que possam auxiliar no ensino-aprendizagem desses conteúdos.

V. LIMITAÇÕES E AMEAÇAS À VALIDADE

A pouco aderência da participação dos alunos nesta pesquisa pode ter ocasionado perda de bons relatos de dificuldades encontradas por eles neste processo de aprendizagem. Essa baixa participação pode ter sido ocasionada por motivos como a falta de tempo por fazerem outras disciplinas ou por trabalharem fora, o e-mail convite ter sido direcionado à caixa de *SPAM* fazendo assim com que o aluno não o visualizasse, entre outros.

Nem todos alunos relataram suas experiências com a riqueza de detalhes desejada nos diários. Pode ter havido falha no treinamento ou no acompanhamento para motivar os alunos. Como os diários eram identificados, pois eram feitos e armazenados no Google Drive, pode ser que alunos ficaram receosos em se expor e ter sua identidade revelada ao professor, apesar de termos deixado claro que isso não aconteceria.

VI. CONCLUSÃO

As dificuldades detectadas mostram que o erro de sintaxe é o mais comum. Esse resultado confirma a pesquisa de Denny et al. (2011), em que afirmam que muitas submissões são realizadas antes que o código esteja livre de erros de sintaxe [7]. Além disso, os alunos mostraram maior dificuldade com a manipulação de vetores e matrizes do que com estruturas de repetição e seleção. A falta de pensamento lógico foi muito citada pelos professores e por alunos, sendo considerada o ponto fundamental das dificuldades. A falta de capacidade de interpretar os exercícios e a escolha da LP também é fonte de dificuldade. Python, considerada por alunos e professores mais simples e fácil de ser usada, traz melhores resultados. Turmas que utilizaram Python tiveram uma diferença média de 27,4% a menos de reprovados do que as que usaram C.

A utilização do método de análise qualitativa *Grounded Theory* trouxe benefícios à pesquisa pois conseguimos identificar nos diários e entrevistas detalhes não inicialmente

planejados, como técnicas utilizadas para amenizar as dificuldades. Uma delas, nada efetiva segundo alguns alunos e professores, é a de decorar o conteúdo em vez de aprendê-lo, por exemplo. Outra técnica para superar as dificuldades no aprendizado é a de compartilhar o conhecimento entre os colegas. Muitas estratégias didáticas são usadas pelos professores para facilitar a passagem do conhecimento, como a de omitir detalhes da LP, explicar usando problemas e exemplos, resolver passo-a-passo, desenvolver o exercício junto com o aluno, fazer simulados e os próprios alunos corrigem uns dos outros, prática intensiva, entre muitas outras.

Tendo em vista os problemas e dificuldades encontrados, nossa sugestão é detalhar os exercícios, colocando exemplos de entrada e saída para o aluno entender melhor, diminuindo assim problemas de interpretação; explicar a lógica e em seguida a sintaxe, mostrando aos alunos erros comuns cometidos nos códigos e porque estão errados. Nosso estudo também mostrou que a motivação dos alunos é importante. Manter linhas de comunicação entre aluno e professor, não somente em sala, facilita que o aluno com maior dificuldade se expresse, ajudando o professor a detectar esses alunos, podendo assim usar estratégias diferenciadas para melhorar o aprendizado e motivação para que os alunos não desistam das aulas.

Como trabalhos futuros, pretendemos expandir o escopo da análise, entrevistando mais professores e coletando dados de diários de mais alunos, bem como realizar um questionário confirmatório sobre as dificuldades identificadas. Esperamos que os resultados desta pesquisa ajudem professores a pensarem em estratégias para dificuldades específicas e pesquisadores a elaborem ferramentas direcionadas.

AGRADECIMENTOS

Agradecimento muito especial aos alunos e professores que disponibilizaram parte do seu tempo preenchendo diário ou sendo entrevistados.

REFERÊNCIAS

[1] Beaubouef, T. and Mason, J. 2005. Why the high attrition rate for computer science students. ACM SIGCSE Bulletin. 37, 2 (2005), 103.

[2] Bennedsen, J. and Caspersen, Michael, E. 2007. Failure rates in introductory programming. ACM SIGCSE Bulletin. 39, 2 (2007), 32–36.

[3] Bolger, N. et al. 2003. Diary methods: Capturing life as it is lived. Annual Review of Psychology. 54, (2003), 579–616.

[4] Bosse, Y. and Gerosa, M.A. 2015. Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo : Um Estudo Preliminar. WEI - Workshop sobre Educação em Computação. (2015), 1–10.

[5] Cechinel, C. et al. 2008. Desenvolvimento de Objetos de Aprendizagem para o Apoio à Disciplina de Algoritmos e Programação. Simpósio Brasileiro de …. (2008).

[6] Correia, A.L. et al. 2015. Uso de avaliação por pares em disciplinas introdutórias de programação. Workshop sobre Educação em Computação. (2015), 1–10.

[7] Denny, P. et al. 2011. Understanding the syntax barrier for novices. Proceedings of the 16th ACM conference on Innovation and technology in computer science education - ITiCSE ’11. (2011), 208.

[8] Garner, S. et al. 2005. My program is correct but it doesn’t run: A preliminary investigation of novice programmers’ problems. Conferences in Research and Practice in Information Technology Series. 42, (2005), 173–180.

[9] Gomes, A. and Mendes, A. 2015. A teacher’s view about introductory programming teaching and learning: Difficulties, strategies and motivations. Proceedings - Frontiers in Education Conference, FIE. 2015–Febru, February (2015).

[10] Gomes, M. et al. 2015. Um estudo sobre erros em programação - Reconhecendo as dificuldades de programadores iniciantes. Cbie (2015), 1398.

[11] Kamiya, R.R. and Brandão, L.D.O. 2009. iVProg - um sistema para introdução à Programação através de um modelo Visual na Internet. Anais do Simpósio Brasileiro de Informática na Educação (2009), 4.

[12] Kim, J. et al. 2014. A Study on the Use Contexts of Personal Computing Devices Using a Diary Study Method. 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Oct. 2014), 290–296.

[13] Lahtinen, E. et al. 2005. A study of the difficulties of novice programmers. ACM SIGCSE Bulletin. 37, 3 (2005), 14–18.

[14] Müller, H. et al. 2015. Understanding and Comparing Smartphone and Tablet Use : Insights from a Large-Scale Diary Study. (2015), 427–436.

[15] Reis, H.T. 1994. Domains of experience: investigating relationship processes from three perspectives. 87–110.

[16] Ribeiro, R. da S. et al. 2014. Programming web-course analysis: How to introduce computer programming? 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. 2015–Febru, February (2014), 1–8.

[17] Rodríguez-del-Pino, J.C. et al. 2012. A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features. Conference on e-Learning, e-Business, Entreprise Information Systems, & e-Government. (2012).

[18] Santos, A. et al. 2015. A Importância do Fator Motivacional no Processo Ensino- Aprendizagem de Algoritmos e Lógica de Programação para Alunos Repetentes. WEI - Workshop sobre Educação em Computação. (2015), 1–10.

[19] Schütz, I. and Stelzer, A. 2015. Field Evaluation of a New Railway Dispatching Software. c (2015), 63–68.

[20] Strauss, A. and Corbin, J. 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. SAGE Publications.

[21] Vahldick, A. et al. 2015. Testando a Diversão em um Jogo Sério para o Aprendizado Introdutório de Programação. XXIII Workshop sobre Educação em Computação. (2015).

[22] Walters, N.T. and Horton, R. 2015. A diary study of the influence of Facebook use on narcissism among male college students. Computers in Human Behavior. 52, August (2015), 326–330.

[23] Wheeler, L. and Reis, H.T. 1991. Self-recording of everyday life events: origins, types, and uses. Journal Personal. 59, (1991), 339–54.

[24] Winslow, L.E. 1996. Programming Pedagogy - A Psychological Overview. ACM SIGCSE Bulletin. 28, 3 (1996), 17–22.

Yorah Bosse is a PhD candidate at the University of São Paulo in the Computer Science Department. She received her MSc degrees from the Federal University of Santa Catarina (UFSC) in Production Engineering and a BSc degree from Foundation University of Blumenau (FURB) in Computer Science. Her thesis research aims to define patterns of difficulties in the process of learning how to program. She is a lecturer at the Federal University of Mato Grosso do Sul and teach introductory programming courses, algorithms and data structure, and others.

Marco Aurélio Gerosa is an Associate Professor at the University of São Paulo, Brazil, working in the Computer Science Department, where he coordinates the Inter-institutional Center for Research on Collaborative Environments on the Web (NAWEB) and the Software Engineering and Collaborative Systems Research Group. He was a visiting professor at the University of California, Irvine. A more formal bio can be found on the page https://www.ime.usp.br/~gerosa/career.html. He researches the intersections of Software Engineering (SE) and Social Computing (CSCW). Currently, He is working on the analysis and support of open-source software development communities and collaborative environments in general. http://lattes.cnpq.br/4507073071352893.