

Celina Luci Lazzari

**CARACTERÍSTICAS DA CLASSE DE COMPORTAMENTOS
'PROGRAMAR COMPUTADORES' COMO PARTE DA
CAPACITAÇÃO DE PROFISSIONAIS DA COMPUTAÇÃO**

Dissertação submetida ao Programa de Pós-graduação em Psicologia da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Psicologia

Orientação: Prof. Dr. Sílvio Paulo Botomé

FLORIANÓPOLIS
2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lazzari, Celina Luci

Características da classe de comportamentos 'Programar computadores' como parte da capacitação de profissionais da Computação / Celina Luci Lazzari ; orientador, Sílvio Paulo Botomé - Florianópolis, SC, 2013.

132 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro de Filosofia e Ciências Humanas. Programa de Pós-Graduação em Psicologia.

Inclui referências

1. Psicologia. 2. Comportamento. 3. Programar computadores. 4. Ensino. 5. Aprendizagem. I. Botomé, Sílvio Paulo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Psicologia. III. Título.

Celina Luci Lazzari

**CARACTERÍSTICAS DA CLASSE DE COMPORTAMENTOS
'PROGRAMAR COMPUTADORES' COMO PARTE DA
CAPACITAÇÃO DE PROFISSIONAIS DA COMPUTAÇÃO**

Esta Dissertação foi julgada adequada para obtenção do Título de “Mestre” e aprovada em sua forma final pelo Programa de Pós-graduação em Psicologia.

Florianópolis, 17 de julho de 2013.

Prof. chefe, Dr^a. Carmen Leontina Ojeda Ocampo Moré
Coordenadora do Curso

Banca Examinadora:

Prof. Dr. Sílvio Paulo Botomé
Orientador
Universidade Federal de Santa Catarina

Prof^a. Dra. Juliane Viecili
Universidade do Sul de Santa Catarina

Prof^a. Dra. Lúcia Helena Martins Pacheco
Universidade Federal de Santa Catarina

À minha pequenina dona Alice

AGRADECIMENTOS

Ao professor Botomé pela paciência com minhas dificuldades, pelas reuniões de orientações, pelo incentivo, motivação e dedicação a mim dispensada.

À professora Olga por tudo o que me ensinou a respeito de pesquisa e por ter despertado em mim o interesse pela Análise do Comportamentos.

À CAPES pela bolsa de estudos que possibilitou dedicação exclusiva às atividades oferecidas pelo curso de Pós-Graduação em Psicologia.

Ao Davi, pela ajuda em cada etapa da elaboração dessa dissertação, pelas correções e por ser meu companheiro para todas as horas.

À minha amiga Cíntia, pela ajuda na correção da minha dissertação, pelas discussões e desabafos que me ajudaram a perceber o mestrado de um outro ponto de vista, e por ampliar minha visão a respeito do que somos capazes de fazer como Psicólogos.

RESUMO

Programar computadores é um conjunto de classes de comportamentos cuja aprendizagem é exigente até por sua acentuada complexidade. Tais exigências se manifestam em queixas de quem ensina e quem está aprendendo a programar. Explicitar as características da classe geral de comportamentos 'programar computadores' como uma interação entre classes de situações antecedentes, classes de respostas e classes de situações consequentes pode aumentar a clareza a respeito do que a constitui, facilitar o ensino para alunos de cursos da área de Ciência da Computação, além de proporcionar desenvolvimento de profissionais dessa área de forma mais eficiente. Fontes de informação foram utilizadas para identificar e observar o fenômeno 'programar computadores' e como recurso de análise das informações, estas foram organizadas segundo a noção de comportamento. Essa organização possibilitou identificar cadeias de comportamentos referentes a diversas etapas do processo comportamental 'programar computadores'. Como resultado, foram identificadas sete categorias de comportamentos, organizadas de forma a seguir uma possível sequência de ensino. As categorias são: 'Avaliar argumentos de acordo com regras lógicas', 'Caracterizar funcionamento de computadores', 'Resolver problemas', 'Construir algoritmos', 'Formalizar algoritmos', 'Escrever programas de computador' e 'Avaliar programas de computador'. As cadeias de comportamentos elaboradas ajudam a formar um programa de ensino para aumentar a probabilidade de alunos serem capazes de programar computadores ao eliminar dificuldades e incluir categorias geralmente desconsideradas como parte da classe geral programar computadores.

Palavras-chave: Comportamentos. Programar computadores. Ensino e aprendizagem.

ABSTRACT

Programming computers is a set of behaviors class whose learning is demanding due to its high complexity. Such demands manifest in complaints made by those who teach and those who are learning to program. Expliciting the characteristics of general behavior class 'computer programming' as an interaction among antecedent classes, response classes and consequence classes may increase the clarity with respect to what constitutes the process of computer programming, facilitate teaching undergraduate students of Computer Science courses, as well as provide more efficient development of professionals in that field. Information sources were used in order to identify and observe the phenomenon of 'computer programming'. As a resource to information analysis, these sources were organized according to the notion of behavior. Such organization made possible the identification of behavior chains referring to different steps of the 'computer programming' behavioral process. As a result, seven broad behavior categories were identified and organized in a way to possibly follow a teaching sequence. These categories are: 'Evaluating arguments according to logic rules', 'Characterizing the operation of computers', 'Solving problems', 'Elaborating algorithms', 'Formalizing algorithms', 'Writing computer programs' and 'Evaluating computer programs'. The elaborated behavior chains aid in the design of a teaching programs in order to increase the probability of students to be capable of programming computers as it helps eliminate obstacles and including categories commonly disregarded as part of general behavior class 'computer programming'.

Keywords: Behaviors. Program computers. Teaching and learning.

ÍNDICE DE TABELAS

Tabela 1 Exemplo de um problema descrito em algoritmo e em uma linguagem de programação, ambas considerando o mesmo nível de especificidade.	42
Tabela 2. Exemplo de procedimento para a etapa 'Identificação das relações de cadeia'	56
Tabela 3. Graus de abrangência da classe geral de comportamento 'Programar computadores'	60
Tabela 4. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Avaliar argumentos de acordo com regras lógicas'	62
Tabela 5. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar funcionamento de computadores'	64
Tabela 6. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar problemas'	67
Tabela 7. Classe de comportamento 'Delimitar escopo do problema' e seus componentes	68
Tabela 8. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Representar problemas'	70
Tabela 9 Conjunto das três classes de comportamentos e seus componentes da cadeia 'Decompor problemas'	71
Tabela 10. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Escrever solução de problemas'	73
Tabela 11. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Diferenciar algoritmo de programa de computador'	75
Tabela 12. Classe de comportamento 'Caracterizar estruturas de controle de fluxo' e seus componentes	76
Tabela 13 Classe de comportamento 'Ler algoritmos' e seus componentes.....	77
Tabela 14. Conjunto das cinco classes de comportamentos e seus componentes da cadeia 'Escrever algoritmos'	78

Tabela 15. Conjunto das seis classes de comportamentos e seus componentes da cadeia 'Identificar equivalência das instruções escritas em linguagem natural e de programação'	82
Tabela 16. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Caracterizar descrições formais de problemas'	85
Tabela 17. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Formalizar algoritmos'	86
Tabela 18. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Nomear variáveis de um programa'	89
Tabela 19. Conjunto das cinco classes de comportamentos e seus componentes da cadeia 'Declarar variáveis'	91
Tabela 20. Conjunto das duas classes de comportamentos e seus componentes da cadeia 'Atribuir valores às variáveis'	93
Tabela 21. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar etapas do processo de programar computadores'	94
Tabela 22. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Projetar programas de computador'	96
Tabela 23. Conjunto das três comportamentos e seus respectivos componentes da cadeia 'Escrever programas de computador'	97
Tabela 24. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Detectar erros'	100
Tabela 25. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Depurar erros'	102
Tabela 26. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Avaliar qualidade de programas de computador'	104

ÍNDICE DE FIGURAS

Figura 1 Representação das relações básicas (as setas) entre os três componentes constituintes da definição de comportamento como 'relação entre o que um organismo faz e o ambiente (anterior e posterior) a esse fazer'. Adaptado de Botomé (2001), pg. 16.....	24
Figura 2 Diferentes tipos de relações básicas entre os três componentes de um comportamento. Adaptado de Botomé (2001), pg. 18.....	25
Figura 3 Exemplo de classificação de comportamentos pela similaridade de sua consequência. Nesse exemplo, uma possibilidade de nome para o comportamento seria 'Chamar atenção dos pais'.....	28
Figura 4 Exemplo de uma sequência comportamental.....	29
Figura 5 Exemplo de uma cadeia comportamental em que o estímulo consequente da primeira classe de comportamento torna-se estímulo antecedente da segunda classe de comportamento.....	30
Figura 6 Exemplo de uma relação de composição entre classes de comportamentos.....	31
Figura 7 Exemplo de protocolo de registro das informações selecionadas.....	52
Figura 8 Exemplo de um destaque de aspectos específicos da informação.....	53
Figura 9 Exemplo de procedimento para a etapa 'Decomposição das classes de comportamentos identificadas'.....	54
Figura 10 Exemplo de protocolo de registro para analisar as classes de comportamentos identificadas em conformidade com seus componentes: classe de estímulos antecedentes, classe de resposta e classe de estímulos consequentes.....	55
Figura 11 Graus de abrangência da classe geral 'Programar computadores' e exemplo de uma sequência de classes de comportamentos denominada 'Construir algoritmos', na quarta coluna ('Grau 4').....	59
Figura 12 Sequência de classes de comportamentos de 'Avaliar argumentos de acordo com regras lógicas'.....	61
Figura 13 Sequência de classes de comportamentos de 'Caracterizar funcionamento de computadores'.....	63

Figura 14 Classes de comportamentos de 'Resolver problemas'	66
Figura 15 Sequência de classes de comportamentos de 'Caracterizar problemas'	66
Figura 16 Sequência de classes de comportamentos de 'Representar problemas'	69
Figura 17 Sequência de classes de comportamentos de 'Decompor problema'	71
Figura 18 Sequência de classes de comportamentos da categoria 'Escrever solução de problemas'	72
Figura 19 Classes de comportamentos de 'Construir algoritmos'	74
Figura 20 Sequência de classes de comportamentos de 'Diferenciar algoritmo de programa de computador'	75
Figura 21 Sequência de classes de comportamentos de 'Escrever algoritmos'	77
Figura 22 Classes de comportamentos de 'Formalizar algoritmos'	80
Figura 23 Sequência de classes de comportamentos da categoria 'Identificar equivalência de instruções em linguagem natural e de programação'	81
Figura 24 Sequência de classes de comportamentos da categoria 'Caracterizar descrições formais de problemas'	84
Figura 25 Sequência de classes de comportamentos da categoria 'Especificar aspectos formais de algoritmos'	86
Figura 26 Classes de comportamentos de 'Escrever programas de computador'	88
Figura 27 Sequência de classes de comportamentos da categoria 'Nomear variáveis de um programa'	88
Figura 28 Sequência de classes de comportamentos da categoria 'Declarar variáveis'	90
Figura 29 Classe de comportamento da categoria 'Atribuir valores às variáveis'	92
Figura 30 Sequência de classes de comportamentos da categoria 'Caracterizar etapas do processo de programar computadores'	94
Figura 31 Sequência de classes de comportamentos da categoria 'Projetar programas de computador'	95

Figura 32 Sequência de classes de comportamentos da categoria 'Escrever códigos em uma linguagem de programação'	97
Figura 33 Classes de comportamentos de 'Avaliar programas de computador'	99
Figura 34 Sequência de classes de comportamentos da categoria 'Detectar erros'	99
Figura 35 Sequência de classes de comportamentos da categoria 'Depurar erros'	101
Figura 36 Sequência de classes de comportamentos da categoria 'Avaliar qualidade de programas de computador'	103

SUMÁRIO

CARACTERÍSTICAS DA CLASSE DE COMPORTAMENTOS 'PROGRAMAR COMPUTADORES' COMO PARTE DA CAPACITAÇÃO DE PROFISSIONAIS DA COMPUTAÇÃO 17

PROBLEMAS NOS PROCESSOS DE ENSINO E DE APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES	17
BENEFÍCIOS PARA A SOCIEDADE DOS CURSOS DA ÁREA DE COMPUTAÇÃO INDICADOS NAS DIRETRIZES CURRICULARES.....	22
O CONCEITO DE COMPORTAMENTO, CLASSE DE COMPORTAMENTO E SUA RELAÇÃO COM PROGRAMAR COMPUTADORES	23
A IMPORTÂNCIA DOS PROCESSOS DE ENSINO E DE APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES E DESENVOLVIMENTO PROFISSIONAL DO CIENTISTA DA COMPUTAÇÃO	32
A IMPORTÂNCIA DO CONCEITO DE ALGORITMO PARA CARACTERIZAR O PROCESSO DE PROGRAMAR COMPUTADORES	37
CARACTERÍSTICAS DAS LINGUAGENS DE PROGRAMAÇÃO E SUA RELAÇÃO COM O COMPORTAMENTO DE PROGRAMAR COMPUTADORES	44

MÉTODO - PROCESSO DE OBTENÇÃO DE DADOS PARA IDENTIFICAR OS COMPORTAMENTOS DA CLASSE DE COMPORTAMENTOS 'PROGRAMAR COMPUTADORES' 49

1. FONTES DE INFORMAÇÕES	49
2. VARIÁVEIS OBSERVADAS	50
3. SITUAÇÃO E MATERIAIS	51
4. PROCEDIMENTO PARA REGISTRAR, ANALISAR E TRATAR DADOS	51

RESULTADOS..... 59

1. CATEGORIA DE COMPORTAMENTOS 'AVALIAR ARGUMENTOS DE ACORDO COM REGRAS LÓGICAS'	61
2. CATEGORIA DE COMPORTAMENTOS 'CARACTERIZAR FUNCIONAMENTO DE COMPUTADORES'	63

3.	CATEGORIA DE COMPORTAMENTOS	
	'RESOLVER PROBLEMAS'	65
4.	CATEGORIA DE COMPORTAMENTOS	
	'CONSTRUIR ALGORITMOS'	74
5.	CATEGORIA DE COMPORTAMENTOS	
	'FORMALIZAR ALGORITMOS'	79
6.	CATEGORIA DE COMPORTAMENTOS	
	'ESCREVER PROGRAMAS DE COMPUTADOR'	87
7.	CATEGORIA DE COMPORTAMENTOS 'AVALIAR	
	PROGRAMAS DE COMPUTADOR'	98
DISCUSSÃO.....		105
REFERÊNCIAS.....		120

I

CARACTERÍSTICAS DA CLASSE DE COMPORTAMENTOS 'PROGRAMAR COMPUTADORES' COMO PARTE DA CAPACITAÇÃO DE PROFISSIONAIS DA COMPUTAÇÃO

Programar computadores é um conjunto de classes de comportamentos cuja aprendizagem é exigente até por sua acentuada complexidade e é também considerado um problema em relação ao seu ensino e aprendizagem. Tais problemas se manifestam em queixas de quem ensina e de quem está aprendendo a programar. Há uma grande quantidade de pesquisas que propõem soluções para diversos problemas relacionados às dificuldades nos processos de ensino e de aprendizagem em programar computadores (Pinheiro, 2003; Santos e Costa, 2005; Vargas, 2005; Esteves e outros, 2007; Jabłonowski, 2007; Souza, 2009; Hinterholz Jr., 2009; Aguiar e Oeiras, 2010; Hernandez e outros, 2010; Lima e Leal, 2010). No entanto, há na literatura indicações de que ainda é necessário produzir conhecimento a respeito desses processos (Weinberg, 1998; Delgado e outros, 2005; Prietch e Pazeto, 2010). Não há clareza suficiente do que constitui 'programar computadores' nem conhecimento das implicações do ensino dessas classes de comportamentos para a aprendizagem e para o desenvolvimento profissional. Há pontos de vista diferentes por meio dos quais é possível estudar o processo de 'programar computadores'. Um conhecimento que possibilita identificar as características do que uma pessoa faz (ações) para programar um computador, as condições que propiciam a ocorrência dessas ações e o tipo de resultado que decorre dessas ações é o conhecimento da Análise do Comportamento. Observar e caracterizar o processo 'programar computadores' por meio do conhecimento psicológico da Análise do Comportamento pode contribuir para propiciar ensino mais efetivo, eficiente e eficaz, aumentar a clareza em relação às dificuldades dos alunos e profissionais em programar computadores, de professores ao ensinar e do que compõe os 'problemas' existentes no ensino e na aprendizagem de 'programar computadores'. O objetivo dessa pesquisa é caracterizar a classe de comportamentos 'programar computadores' como parte da capacitação profissional do cientista da computação.

PROBLEMAS NOS PROCESSOS DE ENSINO E DE APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES

Pereira Júnior e Rapkiewicz (2004) afirmam que a aprendizagem de programação de computadores por alunos de graduação tem sido difícil e que é preciso realizar pesquisas para sua melhoria. Na revisão que fazem a respeito do ensino de programação de computadores no Brasil, constata pesquisas que abordam separadamente ferramentas e estratégias de ensino. Esses autores ratificam a ênfase dada no trabalho de Rodrigues (2002), que propõe alterações didáticas e metodológicas nos processos de ensino e de aprendizagem de 'fundamentos de programação de computadores' como forma de melhorar a qualidade desses processos. Uma explicitação mais completa das classes de comportamentos que compõem a classe geral denominada 'programar computadores' pode tornar possível elaborar melhores condições para diminuir as dificuldades de seu ensino e aprendizagem.

Buzin (2001, citado em Santos e Costa, 2006) afirma, em relação à educação em Computação, que é importante o professor 'ter aulas diferenciadas' porque um aluno se interessa, 'prende sua atenção' e há influência positiva nas avaliações. 'Prender atenção' é um dos aspectos necessários a considerar para resolver o problema das dificuldades em programar computadores, referente a aspectos anteriores àquilo que deve ser ensinado em aula. 'Prender a atenção' é uma das condições que o professor pode arranjar para aumentar a probabilidade do aluno aprender alguma classe de comportamento, mas a classe de comportamento a ser aprendida ainda precisa ser conhecida, explicitada, planejada, programada e avaliada, de forma que o aluno não apenas se interesse pela aula, mas aprenda comportamentos relevantes para ser capaz de programar computadores. É necessário que as aulas promovam aprendizagens de comportamentos relevantes para a vida do aluno, mais do que 'prendam sua atenção' em aspectos de pouco valor para programar computadores.

Os cursos da área de Ciências da Computação apresentam um dos índices mais elevados de desistência e abandono em todo o Brasil, de acordo com um levantamento encomendado pelo MEC em 2009 ao Instituto Lobo para Desenvolvimento da Educação, Ciência e Tecnologia (Silva Filho e outros, 2007; Folha Online, 2009¹). Esse levantamento considera a quantidade de alunos que deixou de se matricular de um ano para o outro e utiliza uma média dos dados de cinco anos do Censo da Educação Superior. Constata-se que 28% da evasão ocorre na área de Ciências da Computação e Matemática. A Diretoria de Educação da Sociedade Brasileira de Computação (SBC, 2010), no Plano de Ação e Gestão para a SBC, confirma a discrepância na relação entre formados e ingressantes

¹ Não foi possível acessar a fonte original do levantamento do MEC

em diferentes cursos da área da Computação, (Ciência da Computação, Sistemas de Informação e Engenharia de Computação): 15% de formados em relação à quantidade de ingressos. A partir desses dados, a diretoria conclui que o alto nível de evasão dos cursos de Ciências da Computação é um dos maiores 'problemas' na Educação em Computação no Brasil. Dados isolados a respeito de cursos de Ciências da Computação ilustram o mesmo padrão de evasão: no curso de Licenciatura em Ciência da Computação da Universidade Federal da Paraíba, Duarte e outros (2010) indicam que a evasão na disciplina de Introdução a Programação atingiu 70% dos alunos no primeiro semestre de 2009. Na Universidade Federal de Santa Catarina, dados disponibilizados a respeito do curso de Bacharelado em Ciências da Computação de 2007 indicam 18,5% de saídas do curso e 7,8% de concluintes naquele ano (UFSC, 2007). Os índices de evasão em cursos da área da Computação, embora não devam ser considerados o problema em si, como afirma a Sociedade Brasileira de Computação, são indicadores de problemas que necessitam ser explicitados e que possivelmente estão relacionados aos processos de ensino e de aprendizagem.

Rodrigues (2002), Chaves de Castro e outros (2003), Schultz (2003), Delgado e outros (2004) (citados em Pereira Júnior & Rapkiewicz, 2004) consideram a baixa motivação, apatia e baixa autoestima de alunos como produtos da dificuldade nos processos de ensino e de aprendizagem de 'programação de computadores' e que tem como decorrência evasão e reprovação. O foco do problema na evasão e desistências pode prejudicar a observação do comportamento do aluno ao aprender a programar e das condições de ensino oferecidas aos estudantes de Ciência da Computação, dois aspectos nucleares do problema, do qual alguns indicadores estão na evasão e reprovação desses alunos. Um aspecto que necessita ser identificado é o que considerar como determinante dos problemas de ensino e de aprendizagem quando são apresentados índices de evasão em cursos de computação. Há aspectos que são partes constituintes de um problema e aspectos que se referem aos determinantes de um problema. Diferenciá-los possibilita compreender o que constitui um problema e o que o determina.

Muitos determinantes para a evasão nos cursos de Ciências da Computação são apresentados. Hipólito (Folha Online, 2009) afirma que há ensino deficitário de matérias exatas no ensino básico e falta de informação das características do curso de Ciências da Computação. Tais cursos, afirma o autor, apresentam 'conteúdos pesados' de cálculos e contas, indo contra as expectativas do aluno, voltadas para fazer jogos e obter oportunidades de emprego na área. Santos & Costa (2006) consideram o alto índice de reprovação em disciplinas de Algoritmos e Programação e

a desistência dos cursos de Ciência da Computação resultado da falta de compreensão do raciocínio lógico. Para Hinterholz Jr (2009, citado em Prietch & Pazeto, 2010), a desmotivação, reprovação e evasão têm problemas em comum e ocorrem devido à dificuldade no desenvolvimento do raciocínio lógico, falta de motivação, ensino 'instrucionista' (que utiliza informações prontas para serem passadas aos alunos, geralmente por meio de um computador) e dificuldade na aprendizagem de comportamentos nomeados pelos autores como 'assimilação de abstrações e desenvolvimento de raciocínio lógico'. Prietch & Pazeto (2010) apresentam uma pesquisa a respeito dos motivos de evasão nos cursos de licenciatura em informática e obtêm como resultado respostas diversificadas, com ênfase em falta de afinidade com o curso, turno de funcionamento e dificuldades financeiras. Esses fatores são uma amostra da diversidade de variáveis consideradas como parte do problema do ensino de Ciências da Computação. Além das dificuldades dos alunos (em desenvolver raciocínio lógico, 'assimilar abstrações') e sua falta de motivação, há as características do ensino das disciplinas básicas de Ciências da Computação como variáveis que contribuem para as dificuldades em programar computadores. Há diferentes variáveis relacionadas às dificuldades em programar computadores, porém, falta diferenciar se elas constituem o problema ou são decorrências de um problema ainda pouco claro que tem como indício a quantidade de pessoas que desistem dos cursos.

Muitos artigos foram escritos com a finalidade de desenvolver software educativos - ambientes de ensino e desenvolvimento de programação de computadores - para solucionar o 'problema de aprendizagem' em programar computadores (Pinheiro, 2003; Vargas, 2005; Santos & Costa, 2005; Esteves e outros, 2007; Souza, 2009; Aguiar & Oeiras, 2010; Hernandez e outros, 2010). Tais artigos carecem de resultados precisos em relação à melhoria da aprendizagem de classes de comportamentos que caracterizam a classe geral 'programar computadores'. Os resultados informam que 'houve melhoria', 'maior aprendizagem', que os 'alunos gostaram', 'sentiram-se mais motivados', mas não há dados que evidenciem a melhoria na aprendizagem desses alunos devido às intervenções propostas. O conhecimento a respeito das classes de comportamentos da classe 'programar computadores' torna-se necessário para aumentar a clareza desse processo e o que desse processo pode ser um determinante de dificuldades para alunos ou professores, possibilitando elaborar critérios relevantes para observar e avaliar o resultado do ensino e da aprendizagem dos alunos.

Pesquisas de outra natureza de procedimentos visando contribuir para a melhoria do ensino e da aprendizagem de programação de computadores recomendam o uso de uma linguagem de programação específica ou a criação de uma linguagem de programação direcionada para o ensino de programação de computadores. Por exemplo, Pinheiro (2003) recomenda o uso de linguagens de programação que sejam semelhantes à linguagem de programação *Pascal*; Hinterholz Jr. (2009) apresenta uma linguagem de programação voltada para o ensino de algoritmos em língua portuguesa, denominada '*Tepequém*'; a linguagem de programação *Logo* é recomendada por Lima & Leal (2010). Também há a recomendação de não usar uma determinada linguagem de programação, como é o caso do trabalho de Jabłonowski (2007), que, ao realizar um estudo de caso sobre a escolha da primeira linguagem de programação para o estudo de Ciência da Computação, conclui que a linguagem *Java* é inadequada para os cursos introdutórios, pois nela se escondem pacotes especialmente concebidos, elementos artificiais no código (como a contagem de *arrays* a partir do número 0 e não do número 1), entre outras características. Jabłonowski (2007) considera que as linguagens mais adequadas para o contexto e objetivo do processo de ensino seriam *Pascal* ou *Smalltalk*. Sem clareza a respeito das classes de comportamentos da classe 'programar computadores' diminui a consistência de pesquisas que recomendam o uso de uma linguagem de programação específica para o ensino ou que visam elaborar ambientes de ensino, já que o que está sendo resolvido por meio dessas intervenções pode não ser o que mais importa para a aprendizagem e o sucesso do ensino de classes de comportamentos da classe 'programar computadores'.

Em síntese, os problemas nos processos de ensino e de aprendizagem de programação de computadores são tratados com diferentes estratégias, com a elaboração de ambientes de desenvolvimento de programas como forma de ensino e com a escolha de uma determinada linguagem de programação. Há um entendimento equivocado a respeito do papel da motivação e da atenção nesse processo, o que dificulta a intervenção nos comportamentos a serem ensinados. Evasão e repetência são consideradas como o problema, quando de fato há mais indícios de problemas nos processos de ensino e de aprendizagem que apresentam, como decorrência, a evasão e a repetição. Outro aspecto importante a considerar é a diferença entre determinantes e constituintes do problema, como forma de aumentar a probabilidade de identificar o que é objeto de ensino do professor e o que determina as dificuldades tanto de ensino quanto de aprendizagem. Tais aspectos podem não ser os melhores meios para resolver o 'problema' de programar computadores, caso não se tenha clareza das classes de

comportamentos que constituem esse fenômeno e quais classes de comportamentos necessitam ser ensinadas. Os benefícios de tal elucidação se estendem não só ao professor e aos alunos, mas sim ao profissional em sua atuação na sociedade.

BENEFÍCIOS PARA A SOCIEDADE DOS CURSOS DA ÁREA DE COMPUTAÇÃO INDICADOS NAS DIRETRIZES CURRICULARES

O Ministério da Educação (BRASIL, 2003) organizou a área geral da Ciência da Computação em 11 cursos: Administração de Redes; Banco de Dados; Ciência da Computação; Computação Gráfica; Engenharia de Computação (hardware); Engenharia de Software; Informática (Ciência da Computação); Sistemas Operacionais; Tecnologia da Informação; Tecnologia em Desenvolvimento de Software e; Tecnologia em Informática. Para os cursos de Bacharelado em Ciência da Computação, Engenharia de Computação, Engenharia de Software e Sistemas de Informação e os cursos de Licenciatura em Computação há diretrizes curriculares que apresentam, dentre outras informações, os benefícios que tais cursos fornecem à sociedade:

'Os computadores têm um papel fundamental na sociedade. Estão presentes, nas comunicações, na saúde, na gestão, nas artes, no ensino e na pesquisa. Não é um exagero dizer que a vida das pessoas depende de sistemas de computação e de profissionais que os mantêm, seja para dar segurança na estrada e no ar ou ajudar médicos a diagnosticar e tratar problemas de saúde, seja com um papel fundamental no desenvolvimento de novas drogas' (BRASIL, 2003, pg. 4)

Essas contribuições são provenientes do conhecimento dos cursos e de profissionais formados na área da Computação e se subdivide em dois tipos. As diretamente dependentes de meios computacionais, nas quais sem a computação o processo não existiria (por exemplo: na construção de sistemas de computação, na manipulação de conjuntos de dados de múltiplas grandezas, etc.); e as contribuições indiretas, que são aceleradas pela computação (por exemplo, no auxílio em áreas como a Matemática, Física, Estatística, Biologia, Meteorologia, etc.) Além disso, há

as diversas possibilidades de uso de redes sociais que são decorrências do avanço no conhecimento da Ciência da Computação (BRASIL, 2003).

A elaboração de programas de computador (*software*) beneficia a sociedade de diversas maneiras. Seus produtos estão presentes na vida de toda a sociedade, em governos, bancos, educação, transportes, medicina, indústrias e muitos outros. Além disso, programas de computador mantêm serviços eletrônicos, programas sociais de governos, fornecimento de energia elétrica, redes de telecomunicações, serviços de transporte aéreo, caixas eletrônicos, cartões de crédito, bolsas de valores e mercadorias, e muito mais (BRASIL, 2003). Produzir conhecimento acerca das classes de comportamentos de quem programa um computador e elabora *software* torna-se essencial devido à quantidade, diversidade e importância desses benefícios.

O CONCEITO DE COMPORTAMENTO, CLASSE DE COMPORTAMENTO E SUA RELAÇÃO COM PROGRAMAR COMPUTADORES

O entendimento do processo 'programar computadores' como um fenômeno psicológico possibilita a identificação das características desse fenômeno e dos aspectos que apresentam mais dificuldade para o ensino, aprendizagem e desenvolvimento profissional. No entanto, o estudo desse fenômeno como comportamento humano (como um fenômeno psicológico) não é comum e bem aceito (Weinberg, 1998). Weizenbaum (1976, pg. 80), baseado na tese de Turing, afirma que todas as falhas em computadores podem ser atribuídas ao comportamento humano:

'Turing demonstrou que todos os computadores (exceto alguns tipos específicos que não vêm ao caso) são equivalentes entre si, isto é, são todos universais. De onde, qualquer falha num computador que funciona bem em termos técnicos, precisamente de acordo com o programa que lhe introduzimos, não pode ser atribuída a qualquer particularidade de um computador específico que tenhamos usado. Na verdade, a falha deve encontrar-se numa desatenção da nossa parte, ao transcrevermos as regras comportamentais que julgamos entender para a linguagem formal que o nosso computador re-

quer, ou então na explicação inicial, independentemente da forma que tínhamos em mente ao estarmos convencidos das nossas ideias, ou então dever-se-á a uma má compreensão. Com frequência, é este o caso'.

A afirmação de Weizenbaum (1976) possibilita entender que o comportamento humano é um conhecimento básico para programar computadores. Quem faz um programa está se comportando e a análise desses comportamentos é nuclear para entender e produzir os resultados desejados no computador. A classe de comportamentos 'programar computadores' pode ser, em termos gerais, entendida como um conjunto de relações entre o que o programador faz, as condições em que o faz, e o que acontece em um computador como resultado desse 'fazer'. As contribuições da Análise Experimental do Comportamento podem ajudar a avançar na caracterização dessas relações, além de possibilitar a elaboração de procedimentos mais eficazes de intervenção no ensino de estudantes e no desenvolvimento de profissionais da Ciência da Computação.

Figura 1 Representação das relações básicas (as setas) entre os três componentes constituintes da definição de comportamento como 'relação entre o que um organismo faz e o ambiente (anterior e posterior) a esse fazer'. Adaptado de Bottomé (2001), pg. 16.



O conceito de comportamento humano utilizado por estudiosos e profissionais da área de conhecimento da Análise Experimental do Comportamento é entendido diferentemente do conceito de comportamento geralmente utilizado em senso comum e em algumas teorias psicológicas: comportamento como tudo aquilo que o indivíduo faz (ações) ou como uma parte restrita da interação do indivíduo com seu meio. Para a Análise Experimental do Comportamento, o comportamento humano é um sistema de inter-relações entre propriedades do que o organismo faz e dos eventos ambientais relacionados a esse fazer. Esse sistema de relações é

composto, além das ações do organismo, das situações que precedem essa ação e do que acontece no ambiente após a ocorrência dessa ação (o que sucede ou decorre da ação). Na Figura 1 há uma representação do conceito de comportamento como um conjunto de relações entre esses três componentes (situação, ação e consequência). Na primeira coluna, a situação antecedente, na segunda coluna, a ação ou resposta (aquilo que um organismo faz) e na terceira coluna, a situação consequente (ou consequência). Cada seta representa um tipo de relação possível entre os componentes.

Na Figura 2 cada relação básica entre os componentes do comportamento ilustrada é explicitada em seu papel básico no sistema de relações.

Figura 2 Diferentes tipos de relações básicas entre os três componentes de um comportamento. Adaptado de Botomé (2001), pg. 18.

Componentes Tipos de relação	Situação (O que acontece antes ou junto à ação de um organismo)	Ação (Aquilo que um organismo faz)	Consequência (O que acontece depois da ação de um organismo)
1	→		
2		→	
3	←		
4		←	
5	→		
6	←		←

Cada uma das seis relações básicas apresentadas por Botomé (2001) enfatizam uma relação específica entre cada componente do comportamento. Na primeira relação, a ênfase está em uma parte do comportamento que se refere à influência dos aspectos de uma situação na ação do organismo (influências de diferentes tipos, facilitadora, sinalizadora, impeditiva ou dificultadora, por exemplo). Na segunda relação, a ênfase está no que se segue ou é produzido no ambiente a partir da ação do organismo. O terceiro tipo de relação (terceira seta) refere-se a diferentes graus de controle (influência) a que a ação de um organismo está em relação a determinados aspectos do meio existente quando a ação é apresentada. As propriedades das consequências indicadas na quarta relação alteram a probabilidade de ocorrência de uma ação da mesma classe que produziu essas consequências. A quinta relação indica que os aspectos da situação 'sinalizam' uma consequência a ser obtida caso uma ação seja apresentada. Na sexta relação o resultado de uma ação altera os aspectos

do meio no sentido de torná-los sinalizadores e providos de 'significado' ao organismo (Botomé, 2001). O conceito de comportamento, considerado em todas essas relações, explicita as relações do organismo com o meio, torna as ações contextualizadas e modificáveis por meio da alteração do ambiente em que elas ocorrem.

Tais relações comportamentais possibilitam entender a grande complexidade envolvida na classe geral de comportamento 'programar computadores'. Muitos autores (Weizenbaum, 1976; Shneiderman, 1980, citado por Pressman, 1995; Pressman, 1995, pg. 677; Weinberg, 1998) consideram que elaborar um programa ou programar é uma atividade humana e que há necessidade de conhecer os aspectos humanos para desenvolver sistemas baseados em computador. Weinberg (1998), no livro *The Psychology of Computer Programming*, afirma que há certo mistério quando se tenta investigar a maneira pela qual a programação é feita, sendo que em geral as pessoas consideram que a programação não é feita de uma determinada maneira, apenas 'é feita', e que uma pessoa sabe ou não sabe programar computadores. O entendimento científico de programação de computadores como uma atividade humana envolve conhecer esse 'mistério' e identificar as maneiras pelas quais ocorre esse fazer. Ao analisar as classes de situações antecedentes, classes de ações e classes de situações consequentes e todas as relações possíveis entre elas, explicitadas na Figura 2, aumenta a probabilidade de conhecer tais aspectos humanos referidos pelos autores citados acima, tornando mais claro o que caracteriza a classe geral de comportamentos 'programar computadores'.

Botomé (2001) sistematizou contribuições acerca do comportamento humano em diferentes tipos de conhecimento e momentos da história. Há diversas formas de entender o que é o comportamento humano, e algumas já estão superadas cientificamente, embora ainda sejam utilizadas, tanto em ambientes acadêmicos quanto no conhecimento de senso comum (Botomé, 2001). Tais formas de compreensão do comportamento humano se evidenciam em como se conceitua o que é comportamento. Algumas vezes, o nome mais apropriado de um comportamento enfatiza a relação entre o que o organismo faz e as condições em que ele faz; em outras há ênfase na própria ação; ou então ênfase na relação entre a ação e o que decorre dela ou outras combinações. Nomear o que acontece apenas considerando o que a pessoa faz, sem identificar as consequências da ação e em quais situações ela faz o que faz, limita o que é possível compreender acerca do seu fazer. O que uma pessoa faz não é alheio ao que acontece antes dela fazer, e o que ela faz tem decorrências no ambiente que retroagem em seu fazer, alterando a probabilidade de ocorrência do mesmo tipo de comportamento no futuro.

Programar computadores necessita ser estudado como uma classe de comportamentos para que seja possível a identificação não só das ações do organismo, mas sim do seu comportamento como um todo, considerado as situações que antecedem e sucedem tais ações. A palavra 'classe' significa que o fenômeno é um conjunto de diversas unidades de comportamentos que apresentam como características em comum o tipo de consequência do comportamento - aquilo que decorre do que o organismo faz em uma dada situação (Botomé, 2001). A partir desse conceito é possível classificar uma unidade de comportamento em uma determinada classe por sua similaridade funcional. O mesmo pode ser dito para cada um dos componentes de um comportamento: cada aspecto da situação antecedente, da ação ou da situação consequente pode variar em graus mantendo a mesma função na relação, por isso são denominados por 'classe' (ou 'conjunto').

Para Follette, Naugle & Linnerooth (1999) o que caracteriza a função do comportamento é 'o exame das variáveis relevantes que controlam um comportamento-objetivo definido, incluindo seus antecedentes, consequências, e as condições sob as quais o comportamento ocorre mais frequentemente'. No exemplo fornecido por esses autores, supondo um cliente que chora durante a terapia, para caracterizar a relação funcional desse choro é necessário identificar sob controle de que aspectos do ambiente essa resposta de chorar ocorre e o tipo de consequência que o cliente obtém ao apresentar tal resposta de chorar.

'... Até que se compreenda qual a função do chorar, isto é, apenas com a observação de que o cliente está chorando não se pode fazer nada. Quando se compreende exatamente o motivo que desencadeou o choro, e que mudanças no ambiente são associadas com as oscilações do choro, pode-se afirmar que foi realizada uma avaliação funcional do choro e o que poderia afetar a sua ocorrência.'

O conceito de classe é fundamental para realizar avaliação funcional do comportamento. Para que um comportamento seja considerado como pertencente a uma mesma classe, não basta que a mesma resposta ocorra, é preciso que a função daquela resposta (o choro, como no exemplo acima) seja identificada. Quando unidades de comportamento apresentam a mesma consequência, mesmo que outros componentes sejam diferentes, essas unidades podem formar uma mesma classe (Figura 3).

Por exemplo, 'chamar a atenção dos pais' pode ser uma 'classe de comportamento' no sentido que muitas respostas ou ações do organismo podem produzir o mesmo resultado. Mudanças em qualquer um dos componentes do comportamento alteram o que se diz que a pessoa está fazendo e como esse comportamento é nomeado.

Figura 3 Exemplo de classificação de comportamentos pela similaridade de sua consequência. Nesse exemplo, uma possibilidade de nome para o comportamento seria 'Chamar atenção dos pais'.

SITUAÇÃO ANTECEDENTE	AÇÃO	SITUAÇÃO CONSEQUENTE
Presença dos pais	Gritar	Atenção dos pais
Presença dos pais	Chorar	Atenção dos pais
Presença dos pais	Falar	Atenção dos pais

Para que dois ou mais comportamentos formem uma classe estes necessitam ter uma função similar no ambiente (Follette, Naugle e Linnerooth, 1999). No exemplo acima, o comportamento foi nomeado como 'Chamar atenção dos pais'; embora sejam diferentes classes de respostas (gritar, chorar, falar, e poderiam ser outras, como se jogar no chão, etc.), a função de tais respostas (ou ações) é a mesma, expressa na coluna 'situação consequente': chamar a atenção dos pais. Essa noção de classe implica em considerar que há maneiras múltiplas pelas quais uma interação ocorre (Follette, Naugle & Linnerooth, 1999)

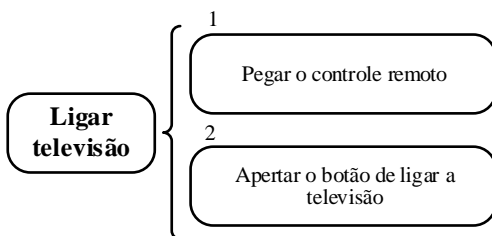
Da mesma forma que é possível se referir a uma 'classe' de comportamentos, os componentes do comportamento também podem ser compreendidos como classes. Diferentes estímulos (antecedentes ou consequentes) e diferentes ações podem ser membros de uma classe comum. Follette, Naugle & Linnerooth (1999, pg. 4) ilustram essa afirmação:

'Para um cliente que tenha um histórico de relacionamentos fracassados, pode-se imaginar que para ele todos os relacionamentos sociais sejam vistos como potencialmente dolorosos. Neste caso, o cliente pode incorretamente discriminar que todos os relacionamentos terminam em sofrimento, ao invés de permitir que ele perceba que algumas pessoas podem ser perigosas, outras não, uma diferença que seria importante ele começar a perceber.'

Nesse exemplo, o estímulo 'relacionamento social' tem uma mesma função (um mesmo 'significado'), devido ao seu histórico de relacionamentos fracassados, fazendo com que o indivíduo classifique todos os estímulos 'relacionamentos' em uma mesma classe, como estímulos aversivos (dolorosos).

As classes de comportamentos podem se relacionar de diversas formas. Uma relação possível entre classes de comportamento é a sequência de classes de comportamentos. Esse tipo de relação é caracterizado pela precedência entre comportamentos ('B' *segue-se* a 'A'), como evidenciado na Figura 4.

Figura 4 Exemplo de uma sequência comportamental



Para 'ligar televisão', uma sequência de classes de comportamentos possível é: 'pegar o controle remoto' e 'apertar o botão de ligar a televisão'. Tais comportamentos são etapas para a ocorrência do comportamento 'ligar televisão'. 'Pegar o controle remoto' *precede* o comportamento 'Apertar o botão de ligar a televisão'.

A relação entre classes de comportamentos em que o estímulo consequente da classe de comportamento anterior é estímulo antecedente da classe de comportamento posterior é denominada de cadeia comportamental. Uma cadeia comportamental é um conjunto de relações entre classes de comportamentos (Millenson, 1975), em que os estímulos consequentes da classe de comportamento 'A' tornam-se estímulos antecedentes para a classe de comportamento 'B'. Em uma cadeia de comportamentos sempre é possível identificar também uma sequência de classes de comportamentos. No conceito de cadeia de comportamentos, cada classe de comportamento tem uma consequência que é condição para ocorrer outra classe de comportamento, considerado o 'elo seguinte' na cadeia de comportamentos. Geralmente uma 'cadeia de comportamentos' (várias classes de comportamentos encadeadas) recebe um nome como se fosse uma classe de comportamento. Na Figura 5 há um exemplo de uma relação de cadeia entre duas classes de comportamentos.

Figura 5 Exemplo de uma cadeia comportamental em que o estímulo consequente da primeira classe de comportamento torna-se estímulo antecedente da segunda classe de comportamento

Classe de estímulo antecedente	Comportamento / Classe de resposta	Classe de estímulo consequente
Controle remoto	Pegar o controle remoto	Controle remoto em mãos

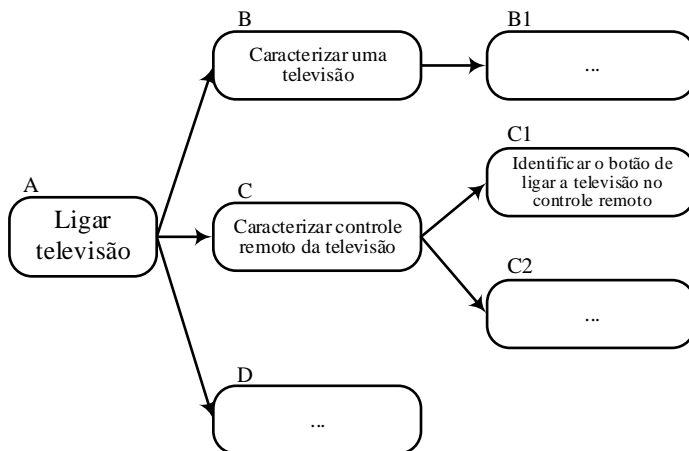
↓

Classe de estímulos antecedentes	Comportamento / Classe de resposta	Classe de estímulos consequentes
Controle remoto em mãos Botões do controle remoto	Apertar o botão de ligar televisão	Botão de ligar pressionado Televisão ligada

A cadeia de comportamentos para ligar televisão é evidenciada pela relação entre as classes de estímulos antecedentes e consequentes. 'Pegar o controle remoto' apresenta como classe de estímulo consequente 'controle remoto em mãos', e este é classe de estímulo antecedente da classe de comportamento 'Apertar o botão de ligar televisão'. As classes de comportamentos também têm entre si uma relação de precedência: 'A' precede 'B', se as classes de estímulos consequentes da classe de comportamento 'A' tornam-se classes de estímulos antecedentes para a classe de comportamento 'B'.

Outro tipo de relação entre classes de comportamentos é a relação de composição (Botomé, 1975; Boratti, 2007). Nesse tipo de relação, a classe de comportamento 'A' é *composta por* outras classes de comportamentos ('B' e 'C') que são necessários para a ocorrência da classe de comportamento 'A', ou, de outra forma, as classes de comportamentos 'B' e 'C' são *partes* da classe de comportamento 'A'. Essa relação possibilita identificar classes de comportamentos pré-requisitos para a ocorrência de outra classe de comportamento. Na Figura 6 há um exemplo de uma relação de composição entre classes de comportamento:

Figura 6 Exemplo de uma relação de composição entre classes de comportamentos



Na Figura 6, 'Ligar televisão' é uma classe de comportamento mais abrangente e composto por outras duas classes de comportamentos: 'Caracterizar uma televisão' e 'Caracterizar controle remoto da televisão' e esta é composta pela classe de comportamento 'Identificar o botão de ligar a televisão no controle remoto' e provavelmente outras classes de comportamentos. Essas classes de comportamentos são pré-requisitos para ligar uma televisão, ou seja, para ser possível ligar uma televisão é necessário que o indivíduo seja capaz de caracterizar o que é uma televisão e identificar formas de ligar uma televisão, entre outras classes de comportamentos.

O processo de 'programar computadores' nem sempre é considerado um comportamento humano, no entanto, esse processo implica em uma pessoa se comportando e produzindo como resultado de seu comportamento um programa de computador. Os diferentes entendimentos acerca do que caracteriza o comportamento humano pode ser parte da dificuldade em considerá-lo como parte importante para o processo de programar computadores. O comportamento humano é um sistema de inter-relações composto de ações do organismo, situações que precedem e sucedem essa ação. Diferentes relações entre comportamentos podem ser explicitadas de forma a evidenciar a complexidade de processos comportamentais como o de programar computadores.

A IMPORTÂNCIA DOS PROCESSOS DE ENSINO E DE APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES E DESENVOLVIMENTO PROFISSIONAL DO CIENTISTA DA COMPUTAÇÃO

Aprendizagem é o nome de um processo que indica uma mudança de comportamento (Catania, 1999). Mudar o comportamento implica em duas condições a se observar, um comportamento antes e outro depois da mudança. Essa mudança compõe uma parte do que é necessário para definir a aprendizagem. O processo de aprendizagem não deve ser entendido como um comportamento (não é correto falar em 'comportamento de aprender') (Kubo & Botomé, 2001). As pessoas aprendem (fazem coisas em certas circunstâncias e obtém certos resultados), mas é necessário explicitar o comportamento que se aprende. No conceito de aprendizagem está implícito que o organismo não apresentava um comportamento que agora apresenta. É a mudança de comportamento que indica a ocorrência da aprendizagem.

Um dos critérios para considerar que um organismo aprendeu é quando este apresenta algum comportamento que não apresentava. Em termos do ensino formal, a mudança de comportamento que é considerada como aprender algo é aquela na qual uma determinada situação antecedente é resolvida, alterada de forma resolutiva (produz como situação consequente a resolução de um problema). Nessa situação de ensino é necessário especificar uma classe de comportamento a ser apresentada que o organismo ainda não apresenta. Ao estabelecer uma classe de comportamento específica a ser aprendida é possível observar se o organismo apresentou aquela classe de comportamento que não era capaz de apresentar e é possível concluir se houve ou não aprendizagem (daquela classe de comportamento) (Kubo & Botomé, 2001). Aprendizagem é definida pela alteração de uma condição do organismo que começa em um estado em que não apresenta determinada classe de comportamento para outro em que apresenta a classe de comportamento. Quando não há alteração das condições por meio da ação do organismo, (mesmo que haja mudança em outras classes de comportamentos), em relação àquela classe de comportamento, não é possível afirmar que ocorreu aprendizagem (Kubo & Botomé, 2001).

A mudança de um comportamento por meio da intervenção planejada de outra pessoa define o comportamento de ensinar. Ensinar é todo comportamento de um professor que produz aprendizagem de algum comportamento de seus alunos. O aspecto mais crítico do comportamento

de ensinar é o que decorre de sua ação: a aprendizagem do aluno. É nesse sentido que se pode falar em processo de ensino-aprendizagem como dois processos interdependentes. Essa concepção possibilita refutar definições que enfatizam outros aspectos secundários do ensinar, como ações isoladas do professor (dar aulas, passar tarefas, cobrar ditados, escrever no quadro). As características do que o professor faz não definem por si só o processo de ensinar. No caso do ensino, o núcleo da definição é a aprendizagem (a mudança de um determinado comportamento do aluno) obtida pelas condições planejadas pelo professor (Kubo & Botomé, 2001).

Para a Análise do Comportamento, ensino e aprendizagem são dois processos comportamentais diferentes que envolvem professor e aluno. Aprender pode ou não envolver a intervenção (por meio de planejamento de condições de ensino) de um professor, o que é condição necessária no processo de ensinar. Tais conceitos têm implicações no estudo da classe de comportamento 'programar computadores'. Ao se referir a programar computadores como uma classe de comportamentos, considera-se que essa classe pode ser aprendida com ou sem auxílio direto de professores e que, se for por meio do ensino, será necessário que o professor planeje condições de ensino adequadas para produzir a aprendizagem dessa classe comportamental nos alunos.

No conhecimento da Análise Experimental do Comportamento é possível identificar alguns princípios que servem para nortear a aprendizagem de um comportamento e que facilitam o planejamento e a programação de ensino. São cinco princípios do ensino programado, que Botomé (1970) adaptou a partir de um texto da Teaching Machines Incorporated, de 1961. O princípio dos pequenos passos se refere a programar o ensino para que o aluno progrida gradualmente, por meio de comportamentos com grau de dificuldade crescente e com início próximo daquilo que o aluno já consegue fazer. Tais passos são organizados para diminuir a probabilidade de erro do aluno, pois as classes de comportamentos são decompostas de forma a tornar suave e fácil o avanço em direção a classes de comportamentos mais complexas. O princípio da resposta ativa se refere à descoberta de que um aluno, ao responder de forma ativa, ou seja, ao se comportar de diferentes maneiras em relação a uma situação-problema, aumenta a probabilidade de aprender. Com o princípio da resposta ativa, há uma ênfase na apresentação de classes de comportamentos por parte do aluno, em oposição às características do ensino tradicional em que o aluno não atua efetivamente para produzir soluções, tendo um papel mais passivo, recebendo informações prontas a serem repetidas, copiando informações, etc. O princípio da verificação imediata originou-se da ob-

servação de que há aumento da aprendizagem quando o aluno tem a oportunidade de verificar seu desempenho imediatamente. O princípio do ritmo individual considera o respeito ao ritmo de aprendizagem de cada aluno como condição importante para o ensino. Além desses princípios que se relacionam diretamente ao processo de aprender do aluno, há um princípio chamado de princípio do teste de avaliação, cuja ênfase está no comportamento do professor e se refere à revisão do que está sendo ensinado a partir dos desempenhos dos alunos. Nesse princípio, é necessário registrar o desempenho dos alunos, identificar quais comportamentos estão muito complexos, para revisar e melhorar as condições de ensino, que podem até ser inadequadas para a aprendizagem do comportamento-objetivo.

É necessário elaborar programas de ensino compatíveis com as necessidades do aluno e coerentes com as necessidades sociais da área de atuação (Jesus & Brito, 2009). As condições de ensino devem estar orientadas por aprendizagens que o aluno precisa realizar e não por recursos e procedimentos didáticos padronizados ou considerados como adequados como tais (Botomé, 1996). Definir o que o aluno precisa aprender (qual comportamento-objetivo ou objetivo de ensino) é o primeiro requisito para programar o ensino. Após a elaboração do objetivo de ensino, o comportamento-objetivo definido deve ser decomposto em comportamentos mais simples, orientando-se pela aproximação ao repertório de entrada do aluno como critério de término da decomposição. Com os comportamentos mais gerais decompostos em seus componentes mais específicos, é necessário realizar a análise de cada um dos comportamentos de forma a explicitar seus componentes e planejar condições de ensino para que cada um deles sejam apresentados pelos alunos. Além desses comportamentos, o professor precisa elaborar situações para que o aluno se comporte, situações o mais semelhante possível daquela em que ele terá que lidar em sua vida profissional. Com a oportunidade de se comportar, também surge a necessidade do aluno avaliar seu desempenho para corrigir e aperfeiçoar seu comportamento. Essa avaliação pode ocorrer com o *feedback* do professor, e as dificuldades que o aluno apresentar devem ser estímulos para que o professor reformule e aperfeiçoe seu programa de ensino, as condições que elaborou, as consequências que forneceu etc. Ao decompor o comportamento-objetivo e analisar os comportamentos decompostos, a probabilidade de erros durante a aprendizagem do aluno diminui e o planejamento do ensino feito pelo professor torna-se mais coerente com o que é necessário ensinar. Professores que ensinam alunos a programar computadores podem se beneficiar ao utilizar tais princípios para elaborar programas de ensino.

A compreensão de comportamentos-objetivo tende a alterar significativamente o comportamento do professor ao ensinar. O tipo de avaliação deixa de ser focado em medir o desempenho do aluno e passa para avaliar sua aprendizagem; aumenta a importância de cada comportamento aprendido, e não apenas o que o aluno repete após um longo período de leituras e exposições do professor (Botomé, 1996). A noção de comportamento-objetivo exige alteração da maneira pela qual os professores planejam e programam suas aulas e as condições de ensino, as atividades propostas e em como verificam os resultados obtidos (Botomé, 1985; Botomé, 1996; Botomé & Kubo, 2003).

No entanto, há vários critérios que parecem orientar a elaboração de um objetivo de ensino com graus de clareza variados do que cada um significa (Botomé, 1985). Há objetivos em que a ênfase está no que o professor deve 'passar' ao aluno ('conteúdos' ou informações), sem que precise verificar o que o aluno faz com esse 'conteúdo'. A palavra 'conteúdo' é uma metáfora que encobre o que acontece ou deveria acontecer com o aluno em um programa de ensino. Zabala (1998) apresenta uma crítica ao ensino de 'conteúdos' como sendo aquilo que deve ser ensinado. O termo 'conteúdo' se refere a informações de matérias ou disciplinas clássicas. Zabala (1998) afirma que é necessário entender 'conteúdo' de forma mais ampla que ensino de informações e fatos, incluindo nesse termo tudo aquilo que é necessário aprender para alcançar determinados objetivos. Objetivos de ensino cuja ênfase está em 'conteúdos' foram avaliados por Delgado e outros (2005) em relação a disciplinas de algoritmos e programação de computadores. Esses autores avaliam que a aprendizagem baseada em 'conteúdos' não supre todas as necessidades dessas disciplinas, fragmenta a evolução e a aprendizagem é diminuída em função do não cumprimento de 'aprendizagens pré-requisitos'.

Outros objetivos cujo núcleo está no professor se referem aos que apresentam intenções do professor sem explicitar o que será garantido por sua intervenção e objetivos com ênfase restrita ao que o professor vai fazer sem clareza de como se relacionam com a aprendizagem do aluno na sua vida profissional. Além desses objetivos, Botomé (1985) identifica objetivos de ensino que contêm atividades do aluno sem clareza da função de cada atividade; objetivos de ensino considerados equivalentes a atividades de ensino, confundindo o que é feito em sala de aula com o que será exigido fora dela; objetivos de ensino que descrevem ações dos alunos derivadas de 'conteúdos' já existentes sem relação com as características da vida do aluno e de seu futuro profissional. Atividades sem função, equivalência entre atividade e objetivo, 'conteúdos' transformados em atividades não caracterizam o que é importante em um objetivo de ensino.

Ensinar 'conteúdos', ações alienadas da realidade, apenas o que o professor gosta ou sabe não são critérios relevantes e mais provavelmente tendem a ensinar comportamentos de pouco valor profissional, como seguir roteiros pré-definidos, repetir informações obtidas na faculdade, ou 'reproduzir' discursos proferidos pelos professores e autores.

Há problemas com as diferentes formulações de objetivos de ensino para a educação superior. Enfatizar 'conteúdos' ou informações como o que deve ser ensinado faz com que diminua a capacidade do profissional de lidar com as situações nas quais se defrontará em sua vida profissional - situações específicas de seu trabalho (Botomé, 1996). Ensinar comportamentos profissionais relevantes para o campo de atuação profissional do aluno é o objetivo de ensino tanto para professores de educação superior quanto para profissionais de nível superior que atuam em organizações. No caso de cursos de Ciência da Computação, é necessário ter clareza de quais são esses comportamentos-objetivo para ser possível derivar, a partir deles, objetivos de ensino relevantes relacionados à programação de computadores. Afinal, são esses comportamentos que, aprendidos, constituirão o 'exercício profissional' nesse campo de atuação.

Além dos aspectos referentes ao processo de ensino e ao processo de aprendizagem, há contribuições da Análise Experimental do Comportamento para o 'desenvolvimento de pessoal' ou 'desenvolvimento profissional' em organizações. O processo de aprendizagem também pode ser entendido como um processo de desenvolvimento de comportamentos profissionais e de constituição de campos de atuação profissional coerentes com a realidade social, dada as características do próprio conceito de comportamento (Kienen & Wolff, 2002). Desenvolver pessoas é uma expressão utilizada na área de conhecimento da Administração e da Psicologia Organizacional. Essa expressão se aproxima do conceito de aprendizagem e de comportamento e pode se referir à aprendizagem de comportamentos relevantes para a atuação do profissional em um campo de atuação.

Para ser capaz de ensinar de forma a produzir aprendizagens da classe geral de comportamentos de 'programar computadores', é necessário conhecer as características dos comportamentos dessa classe e elaborar condições adequadas de ensino para sua ocorrência. Utilizar o conceito de comportamento para elaborar objetivos de ensino e organizar o que vai ser ensinado orientando-se por comportamentos favorece a formação de profissionais capazes de fazer o trabalho que lhes compete com toda a complexidade que se exige de um profissional de nível superior. Além disso, objetivos de ensino orientados por comportamentos garantem

mais conhecimento e controle para o profissional dos efeitos de sua atuação, por explicitar as situações nas quais é necessário atuar, que coisas devem ser feitas e que tipos de resultados são produzidos a partir de sua ação, aumentando sua capacidade de selecionar intervenções mais adequadas a cada situação (Botomé, 1985; Botomé & Kubo, 2003).

As disciplinas de algoritmos e programação têm como objetivo ensinar ao aluno os comportamentos básicos que caracterizam o profissional formado em cursos de Ciências da Computação. Quais são esses comportamentos básicos e quais as relações entre esses comportamentos é uma pergunta a ser respondida. Considerando esses aspectos, caracterizar a classe de comportamentos 'programar computadores' como parte da capacitação profissional do cientista da Computação pode ser objetivo de um trabalho de investigação científica que avançará no conhecimento relativo a esses problemas.

A IMPORTÂNCIA DO CONCEITO DE ALGORITMO PARA CARACTERIZAR O PROCESSO DE PROGRAMAR COMPUTADORES

O conhecimento a respeito do processo de programar computadores já está disponível, embora disperso, com contribuições em diferentes etapas desse processo e explicitado de formas diferentes. Há definições de programar computadores que necessitam ser explicitadas como forma de aumentar a clareza desse fenômeno importante da área de conhecimento da Ciência da Computação.

Ynoguti (2005) considera que programar é 'essencialmente projetar'. Para Guerreiro (1986) 'programar é resolver problemas usando uma linguagem de programação'. Rocha (2006) define programar computadores como representar, segundo modelos diferentes, a solução do problema a ser resolvido na máquina e entende que linguagens de diferentes paradigmas de programação são meios para resolver um problema. Essas definições não incluem de forma explícita os componentes do comportamento de quem programa um computador. Incluir o conceito de comportamento como parte da definição de programar computadores pode facilitar o entendimento do que é essa classe de comportamentos e tornar explícito que programar computadores é um sistema de comportamentos humanos.

Menezes & Coello (2006) ampliam o conceito de programar computadores, ao afirmar que 'programar não é simplesmente codificar uma

solução em termos de uma linguagem aceitável a um computador, mas também identificar e definir o problema e desenvolver uma solução lógica para esse problema'. Aviz Junior (2007) considera que programar computadores implica em descrever uma sequência de ações em uma linguagem que o computador possa executar. Para que seja possível executar, é necessário que o algoritmo seja traduzido para uma linguagem de programação adequada. Tais ações necessitam ser explicitadas para que o processo de programar computadores torne-se mais evidente. Há ênfase, nas definições apresentadas, de que programar computadores envolve um problema a ser resolvido e uma linguagem de programação a ser utilizada como recurso para que o computador resolva o problema.

Além da ênfase em resolver problemas, destacado nas definições acima, há mais aspectos definidores de 'programar computadores' indicados por outros autores. Cristovão (2008), baseando-se em Papert (1994), enfatiza processos de descrever, executar, refletir e depurar, que ocorrem com o programador ao programar. Para Weizenbaum (1976), programar é um teste de compreensão como escrever. A diferença da escrita comum está em que ela é uma linguagem natural, portanto, flexível e ambígua, possibilitando ocorrer falhas de lógica e falta de compreensão. Já o computador seria como um intérprete que não admite ou funciona com tais falhas. Weizenbaum (1976) considera que programar é um processo experimental, como qualquer outra forma de escrita, em que a compreensão perfeita a priori do ato de programar não é necessária. Concepções diferentes a respeito de programar computadores são indicadas pelas definições apresentadas. A Análise Experimental do Comportamento pode contribuir para diferenciar as características das classes de comportamentos envolvidas no uso de linguagem natural e uso de uma linguagem de programação, facilitando o entendimento desses aspectos da classe geral de comportamentos programar computadores.

Outras características que ajudam a definir programar computadores podem ser identificadas com base no que Ascencio & Campos (2007) apresentam a respeito das etapas para desenvolvimento de um programa. Tais etapas possibilitam definir programar computadores como um processo composto de: análise do enunciado de um problema; descrição do problema e de suas soluções por meio de um algoritmo e transformação do algoritmo em códigos de alguma linguagem de programação específica - uma codificação. A definição de programa, para Ascencio (1999, citado em Ascencio & Campos, 2007) é 'a codificação de um algoritmo em uma determinada linguagem de programação'. Pereira Junior (2006) entende programar computadores como um processo composto de instruções estruturadas de forma lógica que o computador executa para obter

soluções para um conjunto de problemas, sendo que tais instruções são definidas por meio de uma linguagem de programação. Há coerência nas definições apresentadas em relação a programar computadores ter como características a exigência de um problema a ser solucionado e a necessidade de uso de algoritmos como recurso para solucionar o problema. Outros aspectos referentes ao que acontece com quem programa podem ser mais bem explicitados e reafirmam a importância de progredir na especificação de quais classes de comportamentos constituem esse processo.

Em algumas definições de programar computadores recorre-se ao conceito de algoritmo. De acordo com Knuth (1997, pg. 1), a noção de algoritmo é básica para toda a programação de computadores. Essa palavra foi utilizada na forma antiga 'algorismo', e se referia ao processo de fazer aritmética com algarismos arábicos. Derivado do nome de um autor árabe (Mohamed ben Musa Al-Khwarizmi ou Abu Abd 'Allah Muhammad ibn Musa Al-Khwarizmi) que introduziu a numeração decimal no ocidente, teve seu uso generalizado para se referir a qualquer processo de cálculo (Knuth, 1997; Abbagnano, 2007).

As definições de algoritmo muitas vezes se aproximam das definições de programar computadores. No dicionário de lógica, Hegenberg (1995) define algoritmo com base no conhecimento matemático, e afirma que se refere 'a processos (ou métodos, ou procedimentos) de cálculo com símbolos (não obrigatoriamente numéricos), adotando regras bem determinadas - e que, a par disso, conduz à solução de qualquer problema de certa classe fixa de problemas'. Ascencio (1999 citado em Ascencio & Campos, 2007) apresenta uma definição de algoritmo como 'uma descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa'; e Manzano (1997, citado em Ascencio & Campos, 2007) considera um algoritmo composto de 'regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas'.

Outras definições a respeito de algoritmos, cuja ênfase está em instruções ou conjunto de passos para realizar uma tarefa ou atingir um objetivo, são apresentadas por Forbellone (1999, citado em Ascencio & Campos, 2007), Forbellone & Eberspächer (2005) e Sipser (2005). A noção de sequência finita de passos ou instruções é base das definições de Gersting (1995), Salvetti (1999, citado em Ascencio & Campos, 2007) e Brookshear (2003). Definições que incluem o que antecede a ação de elaborar um algoritmo e o que deve decorrer dessa ação são apresentadas por Farrer (1999, citados em Ascencio & Campos, 2007), Forbellone & Eberspächer (2005) e Said (2007). Para Forbellone & Eberspächer (2005),

'quando elaboramos um algoritmo, devemos especificar ações claras e precisas, que a partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem definido. Isso significa que o algoritmo fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com vistas a alcançar, como resultado final, a solução de um problema, garantindo que sempre que executado, sob as mesmas condições, produza os mesmos resultados' (pg. 3).

Embora não explicitem nesse trecho que ações concretamente são essas, sob que condições elas serão executadas, e embora haja o uso de uma metáfora ('norma de execução a ser trilhada'), Forbellone & Eberspächer (2005) avançaram em sua definição de algoritmos, possibilitando entender melhor quais condições são necessárias para elaborar um algoritmo. A classe de comportamento elaborar um algoritmo parte de uma situação diferente daquela em que o algoritmo será executado. Os estados inicial e final da execução de um algoritmo são parte da descrição do problema que o algoritmo resolve, portanto fazem parte da situação antecedente da elaboração do algoritmo. As definições de algoritmos contêm conceitos em comum, embora escritos de formas diferentes: descrição de passos (ações, instruções, sequências), uma tarefa ou problema a resolver, e como resultado, sua solução. Essas definições não explicitam classes de comportamentos completos que a pessoa deve realizar e não explicitam a função dessas classes de comportamentos, o que pode ser desenvolvido ao explicitar as características da classe geral de comportamentos 'programar computadores'.

É possível perceber que há uma relação entre elaborar um algoritmo e programar computadores. Podem ser conceitos diferentes utilizados para se referir a um mesmo processo, sinônimos ou conceitos que se referem a dois processos diferentes, embora relacionados. Diversos autores enfatizam que a relação entre esses conceitos se refere à tradução que ocorre de um algoritmo para uma linguagem de programação (Pereira Junior, 2006; Ascencio & Campos; 2007; Aviz Junior, 2007). Pereira Junior (2006) diferencia elaborar um algoritmo de programar computadores considerando que algoritmo pode ser descrito sem formalismo, enquanto que na linguagem de programação de computadores é necessário um formalismo rígido 'para expressar instruções que um computador deve executar

para solucionar um problema'. Aviz Junior (2007) considera que programar computadores é interpretar problemas do mundo real por meio de algoritmos e traduzi-los para uma linguagem formal de programação para o computador. Em ambos os autores, há uma 'passagem' entre elaborar um algoritmo e programar em uma linguagem de programação que ocorre por meio de um processo de tradução. Para programar seria necessário ser capaz de traduzir algoritmos em linguagens acessíveis ao computador.

Há muitos sentidos possíveis em que os autores utilizam o termo 'traduzir' para referenciar o que ocorre entre elaborar um algoritmo e transformá-lo em uma linguagem de programação de computador. Uma das possibilidades de entendimento do que é a tradução é comparar com a tradução que ocorre em linguagens naturais (considerando linguagens naturais as diferentes línguas como portuguesa, alemã, francesa etc.). Para traduzir uma linguagem natural para outra, é necessário conhecer os conceitos e regras gramaticais de ambas as linguagens. Para traduzir um algoritmo para uma linguagem de programação seria necessário conhecer os conceitos (palavras-chave, nomes de variáveis) e as regras 'gramaticais' da linguagem de programação. Fica implícito, ao considerar que, para programar, basta elaborar um algoritmo e traduzi-lo, que o algoritmo estará escrito em linguagem natural, mas já incluso diversas características da linguagem de programação a ser traduzida. No processo de transformação de um algoritmo para uma linguagem de programação, há mais características a considerar além de traduzir a descrição dos passos de um algoritmo. Explicitá-las, sob a forma de classes de comportamentos a serem ensinadas ou aprendidas, parece ser útil para desenvolver aprendizagens significativas, tanto para alunos de cursos de Ciência da Computação, quanto para profissionais que atuam nessa área.

Para comparar as características e diferenças entre um algoritmo e um código escrito em uma linguagem de programação, o seguinte exemplo pode ser útil: em uma partida de cacheta, dado o problema de identificar quando três cartas formam um 'jogo' (isto é, em que condições três cartas podem ser abaixadas da mão do jogador), uma possível solução em forma de algoritmo e em uma linguagem de programação estão apresentadas na Tabela 1:

Tabela 1 Exemplo de um problema descrito em algoritmo e em uma linguagem de programação, ambas considerando o mesmo nível de especificidade.

	Em algoritmo	Em linguagem de programação imperativa 'C'
1	Dadas três cartas:	boolean formaJogo(cartas[] cartas) {
2		if (cartas[0].numero == cartas[1].nu-
3	- Se os números das 3 cartas forem	numero
4	iguais e os naipes das 3 cartas forem	&& cartas[1].numero == cartas[2].numero
5	diferentes entre si, então é um jogo.	&& cartas[0].naipe != cartas[1].naipe
6	Senão, se os naipes das 3 cartas forem	&& cartas[1].naipe != cartas[2].naipe
7	iguais:	&& cartas[2].naipe != cartas[0].naipe)
8	- Ordene as cartas em ordem	{
9	crescente de número.	return true;
10	- Se o número da carta menor	}
11	for o antecessor do número da carta do	else if (cartas[0].naipe == car-
12	meio, e o número da carta do meio for	tas[1].naipe
13	igual ao antecessor do número da carta	&& cartas[1].naipe == cartas[2].naipe)
14	maior, então é jogo. Senão, não é	{
15	jogo.	ordenarEmOrdemCrescente(cartas);
16	Fim.	if (cartas[0].numero == cartas[1].nu-
17		mero + 1
18		&& cartas[1].numero == cartas[2].numero
19		+ 1) {
20		return true;
21		}
22		else {
23		return false;
24		}
25		}
26		else {
27		return false;
28		}
29		}

Em relação à primeira coluna da Tabela 1, onde a solução do problema está escrita em forma de algoritmo, uma das condições nas quais um conjunto de cartas é um jogo está descrita na linha 3 à linha 5: quando o número das três cartas forem iguais e os naipes das três cartas forem diferentes entre si. A outra condição na qual um conjunto de três cartas forma um jogo é descrita na linha 6 à 16: se os naipes das três cartas são iguais, elas podem formar um jogo caso os números das cartas formem uma sequência. Para verificar isso antes é necessário ordenar as cartas em ordem crescente, caso contrário sequência do tipo 'carta rei', carta às e carta nº 2 poderiam ser consideradas um jogo. Por fim, na linha 14-15 é explicitado que, caso nenhuma das condições anteriores se verifique então o conjunto de três cartas não forma um jogo.

Para a solução do problema em linguagem de programação imperativa 'C', na linha 1 da segunda coluna da Tabela 1 está descrito a criação de um procedimento denominado 'formaJogo', pois sua função é identificar quando um conjunto de três cartas formam um jogo. Procedimentos são a forma usada para delimitar os passos da solução na linguagem de programação 'C'. Esse procedimento método é do tipo booleano (ou seja, que pode resultar em valores do tipo 'verdadeiro' ou 'falso'). Os procedimentos podem receber como dado de entrada certos parâmetros (variáveis

ou outros procedimentos). Na linha 1 é possível observar o dado de entrada 'carta[] cartas'. Este dado de entrada é um variável composta (que pode receber um conjunto de valores, o que está sinalizado pelos colchetes), do tipo carta[], ou seja, os valores que pode receber são do tipo 'cartas'. Para que seja possível se referir a uma variável do tipo 'carta', é necessário que em alguma parte do código esta tenha sido criada. Isso significa que foi criado uma variável denominada 'carta', com características específicas, como número e naipe, para que seja possível se referenciar a ela no procedimento formaJogo. Após o procedimento ser criado, uma chave é aberta, indicando o início da solução do problema que o procedimento formaJogo vai executar. Da linha 2 até a linha 7 uma das condições nas quais um conjunto de três cartas forma um jogo é verificada, que é quando o número das cartas é igual e o naipe é diferente. A partir da linha 2 pode-se ler o código da seguinte maneira: se (*if*) o número da 1ª carta (*cartas[0].numero*) for igual (*==*) ao número da 2ª carta (*cartas[1].numero*) e (*&&*) se o número da 2ª carta for igual ao número da 3ª carta (*cartas[1].numero == cartas[2].numero*) e se o naipe da 1ª carta for diferente (*!=*) do naipe da 2ª carta (*cartas[0].naipe != cartas[1].naipe*) e se o naipe da 2ª carta for diferente do naipe da 3ª (*&& cartas[1].naipe != cartas[2].naipe*) e se o naipe da 3ª carta for diferente do naipe da 1ª carta (*&& cartas[2].naipe != cartas[0].naipe*) então retorne para o procedimento formaJogo o valor verdadeiro (*true*, linha 9), ou seja, o conjunto de três cartas forma um jogo. Na linha 11 inicia a descrição da segunda condição na qual um conjunto de três cartas forma um jogo: se a condição anterior não se verifica então (*else if*) verificar se o naipe da 1ª carta é igual ao da 2ª e se o da 2ª é igual ao da 3ª (*cartas[0].naipe == cartas[1].naipe && cartas[1].naipe == cartas[2].naipe*). Se essa condição for verdadeira, então deve-se ordenar as cartas em ordem crescente, utilizando um procedimento denominado 'ordenarEmOrdemCrescente(cartas)', para poder verificar se as cartas estão em sequência (*if(cartas[0].numero == cartas[1].numero + 1 && cartas[1].numero == cartas[2].numero + 1)*). Por fim, se nenhuma das condições anteriores se verifica (*else*), o procedimento resulta no valor falso (*return false*), indicando que as cartas contidas na variável de entrada 'cartas' não formam um jogo).

Uma característica das linguagens de programação é que elas têm um alto nível de detalhamento, característica que parece não ser necessária a um algoritmo. Sebasta (2003) afirma que 'a linguagem na qual [programadores] desenvolvem o software impõe limites quanto aos tipos de estruturas de controle, de estruturas de dados e de abstrações que eles podem usar; assim, as formas de algoritmos possíveis de serem construídas

também são limitadas.' As limitações da linguagem de programação podem exigir uma grande transformação do algoritmo para ser possível seu uso, ao invés de uma tradução simples. O grau de minúcia com que o algoritmo deve estar descrito para possibilitar uma tradução para linguagem acessível a um computador parece ser grande para contemplar todas as características que serão exigidas quando o algoritmo for traduzido em uma linguagem de programação. 'Traduzir' pode não ser uma palavra que expressa adequadamente todo processo de transformação de um algoritmo em uma linguagem apropriada a um computador. É possível entender a metáfora, mas talvez não seja uma forma adequada para ensinar pessoas a programar computadores. Essa palavra pode ser entendida como uma grande classe de comportamentos, genérica, que é composta de outras classes de comportamentos, estas sim, expressivas do ponto de vista de caracterizar essa parte do processo comportamental de programar computadores, por serem mais 'microscópicas' do que a metáfora 'traduzir para uma linguagem de programação' representa.

Em síntese, as diferentes definições de algoritmo e de programar computadores encontradas na literatura demonstram haver uma relação próxima entre esses dois conceitos. Tanto para algoritmo quanto para programar computadores é necessário haver um problema a resolver e passos a elaborar para que a solução seja produzida. No caso de programar computadores, é necessário também que uma linguagem de programação seja escolhida para que os passos para a solução do problema sejam aceitos pelo computador. Elaborar um algoritmo é uma etapa de programar computadores, pois o processo denominado de 'tradução' envolve a transformação do algoritmo em um programa de computador. No entanto, ao se referir a esse processo como 'tradução', diversas classes de comportamentos necessários ficam encobertas. Há mais características da tradução que podem ser explicitadas por meio da noção de comportamento, facilitando o ensino e aprendizagem dessa etapa essencial para que um aluno aprenda a programar computadores.

CARACTERÍSTICAS DAS LINGUAGENS DE PROGRAMAÇÃO E SUA RELAÇÃO COM O COMPORTAMENTO DE PROGRAMAR COMPUTADORES

Programar computadores ocorre por meio de uma linguagem de programação específica e tem como resultado um programa de computador. Há grande quantidade e diversidade de linguagens de programação

de computadores. Utilizam-se os termos 'programar computadores' e 'escrever códigos em uma linguagem de programação' como sinônimos em muitos documentos. No entanto, Santos & Costa (2005) argumentam que o estudo das linguagens de programação corresponde a uma parte do estudo de programação, pois são ferramentas que representam o resultado da aplicação 'de conhecimentos que transformam a especificação da solução de um problema em um programa de computador que efetivamente resolve aquele problema' (pg. 3). Para esses autores, antes de estudar linguagens de programação, é necessário estudar os principais paradigmas de programação, aprender conceitos de algoritmo, fundamentos de lógica, para depois codificar em linguagens de programação específicas as resoluções descritas.

Outro tipo de influência das características das linguagens de programação no comportamento de quem programa se refere à capacidade de aprender, aplicar e manter tais linguagens (Pressman, 1995, pg. 681). A depender do tipo de linguagem de programação, a capacidade de utilizá-la, por exemplo, varia em grau de dificuldade. Há classificações das linguagens de programação feita por autores como Pressman (1995) e Sebesta (2003) que ajudam a compreender suas características. Pressman (1995, pp. 678-681) apresenta uma classificação das linguagens de programação baseando-se em características que denomina como 'psicológicas', que considera não mensuráveis, mas se manifestam em todas as linguagens de programação: a *uniformidade* é uma característica que diz respeito às notações ou representações utilizadas em uma linguagem. Por exemplo, uma linguagem de programação pode apresentar um símbolo com duas funções diferentes. A uniformidade se refere à consistência do uso das notações, à arbitrariedade de restrições e suporte de exceções sintáticas e semânticas às regras; a *ambiguidade* se refere a como uma linguagem de programação é percebida e interpretada pelo computador e pelo programador. Considera-se que um computador tem uma restrição muito grande a ambiguidades na linguagem, em comparação com um programador; uma linguagem de programação é considerada *concisa* pela quantidade de informações 'orientadas para o código que devem ser recuperadas novamente da memória humana'. Ou seja, linguagens concisas apresentam poucas palavras-chave, abreviações, tipos de dados, operadores lógicos e aritméticos e funções embutidas; a *localidade* de uma linguagem auxilia o programador no tratamento de exceções no código. Uma linguagem de programação com boa localidade é organizada em módulos ou blocos e coesa. Uma linguagem que suporta ou estimula o manuseio de exceções viola a localidade. Weinberg (1998) afirma que a lo-

calidade se refere à propriedade de um programa em que suas partes relevantes estão dispostas em um mesmo lugar, ou em uma mesma página, o que facilita a visualização para o programador; a *linearidade* de uma linguagem é a característica que facilita a percepção humana, por apresentar operações lógicas em sequência e com poucas ramificações; a característica *tradição* se refere à capacidade de aprendizagem de outras linguagens a partir da linguagem que foi primeiramente aprendida pelo programador. Há linguagens que apresentam semelhanças entre si e que facilitam a aprendizagem, em oposição a linguagens com formatos e origens diferentes, cuja aprendizagem será mais demorada.

Uma linguagem de programação pode ou não apresentar cada uma dessas características 'psicológicas'. Essas características podem facilitar ou dificultar tanto a aprendizagem, o ensino quanto à elaboração de programas de computador. Por exemplo, uma linguagem com pouca uniformidade e muita ambiguidade terá programas escritos de formas menos legíveis para outros programadores, e aumento da propensão a erros. A capacidade de elaboração de problemas em uma linguagem muito linear também pode ficar restrita, embora essa característica facilite quando o problema já está bem detalhado.

As características de uma linguagem de programação tem grande impacto na qualidade e eficiência do processo de 'tradução' (Pressman, 1995, pg. 676). Esse autor escreve que 'ruídos' podem interferir na etapa de 'tradução' do processo de programar computadores de muitas maneiras. Tais interferências se referem às restrições ou complexidades da própria linguagem de programação. Pressman (1995, pg. 676) afirma que "as características de uma linguagem de programação podem influenciar a maneira segundo a qual pensamos, propagando estruturas de dados e desenhos de software desnecessariamente limitados" e restringem "a maneira pela qual nos comunicamos com um computador" (pg. 681). Isso significa que as linguagens de programação restringem o comportamento de quem programa. É necessário identificar as relações entre as classes de comportamentos de programar computadores e as características das linguagens de programação, as características dessa influência e outras classes de comportamentos que podem ser apresentadas para que tais limitações possam ser superadas.

Cada linguagem de programação pode ser classificada em um paradigma. Há muitos paradigmas de programação e muitas linguagens de programação para cada paradigma. Sebasta (2003, pg. 36) apresenta uma categorização sintética das linguagens de programação em quatro classes: as linguagens de programação imperativas, funcionais, lógicas e 'orienta-

das a objeto'. Uma linguagem de programação *imperativa* apresenta ordenação de execução das instruções específica e algoritmos especificados com grandes detalhes. São derivadas do *paradigma orientado para procedimentos*. O *paradigma funcional* se caracteriza por aplicar funções a determinados parâmetros para computar dados. É baseado na elaboração de funções matemáticas descritas como sistemas de equações, como por exemplo: $f(x) = x + 1$. Funções complexas são construídas pela composição de outras funções mais simples e pelo emprego de recursão, que é uma função é definida em função dela mesma².

Há também o *paradigma de programação lógico*, representado especialmente pela linguagem denominada PROLOG, que se baseia em regras em vez de instruções. Por exemplo, em um algoritmo baseado em linguagem imperativa, uma instrução para avaliar se X é par poderia ser escrita como: *calcule o resto da divisão de X por 2, compare com 0. Se for igual, escreva X é par*. Já um programa em PROLOG é formado por um conjunto de regras lógicas seguido por um questionamento lógico. Em um algoritmo baseado no *paradigma de programação lógica*, a regra a ser escrita seria: *X é par se o resto da divisão de X por 2 é igual a 0*.

Por fim, as linguagens de programação *orientada a objeto*, que tem seu uso bastante popularizado, são derivadas das linguagens imperativas. Nas linguagens desse tipo, são criados objetos que representam entidades do mundo real e que se comunicam por meio de envio de mensagens. A computação ocorre por meio do envio de mensagens de um objeto a outro, que provoca a execução de uma resposta do destinatário, possivelmente enviando mensagens a outros objetos. O *paradigma dessas linguagens é denominado paradigma da programação orientada a objetos*. Todos os paradigmas de programação são equivalentes em termos do que possibilitam fazer. Não há *paradigma de programação mais poderoso que outro*.

² Gersting (1995) explica que uma definição recursiva é aquela na qual o item que está sendo definido aparece como parte da definição. Uma função recursiva contém duas partes:

1. Uma base, onde casos simples do item que está sendo definido são dados explicitamente, e
2. Um passo indutivo ou recursivo, onde outros casos do item que está sendo definido são dados em termos dos casos anteriores.

A parte 1 fornece um ponto de partida na medida em que trata alguns casos simples; enquanto a parte 2 permite construir novos casos a partir desses casos simples, para então construir outros casos a partir desses novos, e assim por diante (Gersting, 1995, pg. 67).

Cada linguagem de programação tem suas características próprias que podem facilitar ou não a elaboração de uma tarefa em comparação com uma linguagem de programação pertencente a outro paradigma.

Tratar dos problemas nos processos de ensino e de aprendizagem de programação de computadores por meio da caracterização das classes de comportamentos que constituem essa classe geral e da elaboração de cadeias de comportamentos que viabilizem a elaboração de programas de ensino pode trazer benefícios para os aprendizes e profissionais cientistas de computação, por tornar esse processo comportamental mais bem elucidado e por incluir aspectos que não são tradicionalmente considerados como parte integrante do que é denominado como programar computadores. Para isso, é necessário compreender que aprender a escrever programas de computador e aprender as características das linguagens de programação não bastam para ensinar diferentes pessoas a programar computadores. Portanto, o objetivo dessa pesquisa é realizar a caracterização da classe de comportamentos 'programar computadores' como parte da capacitação profissional do cientista da computação.

II

MÉTODO - PROCESSO DE OBTENÇÃO DE DADOS PARA IDENTIFICAR OS COMPORTAMENTOS DA CLASSE DE COMPORTAMENTOS 'PROGRAMAR COMPUTADORES'

1. FONTES DE INFORMAÇÕES

Para observar (e identificar o que acontece com) a classe de comportamentos 'Programar computadores' foram escolhidas fontes de informação bibliográficas, incluindo livros e artigos. A literatura contém indicações de diversas classes de comportamentos que são necessárias para programar computadores. Há artigos, livros didáticos, tutoriais, livros para ensino de programação, teses e dissertações disponíveis que apresentem conceitos relacionados ao ensino de programação de computadores, procedimentos e técnicas de ensino, pesquisas de verificação de ferramentas de ensino e uma grande quantidade de produção científica a respeito das características da classe geral 'programar computadores'. A escolha das fontes de informação foi feita por meio de buscas em *sites* da internet que incluíam revistas científicas, *sites* de eventos da área da computação e bibliotecas, utilizando uma combinação de diversas palavras-chave, como 'programar computadores', 'linguagem de programação', 'ensino de programação', 'algoritmo', 'Ciências da Computação', 'comportamento', 'programação', 'dificuldade em programação de computadores'. Os livros e artigos selecionados foram aqueles que apresentaram alguma característica do processo básico de programar computadores, de elaboração de algoritmos e dos processos de ensino e de aprendizagem dessa classe de comportamento em seus títulos ou resumos, identificados pelas palavras-chave.

A literatura selecionada foi:

1. DEHNADI, S. **A cognitive study of learning to program in introductory programming courses**. PhD thesis, Middlesex University, 2009.
2. DELGADO, C. A. D. M.; XEXÉO, J. A. M.; SOUZA, I. F.; RAPKIEWICZ, C. E.; PEREIRA JUNIOR, J. C. R. **Identificando competências associadas ao aprendizado de leitura e construção de algoritmos**. In: XXV Congresso da

- SBC - X Workshop de Educação em Informática), São Leopoldo - RS. Anais do XXV Congresso da SBC, v. 1, 2005.
3. MATTOS. M. M. **Construção de abstrações em lógica de programação.** Em XX Congresso Nacional da Sociedade Brasileira de Computação, volume 1. Editora Universitária Champagnat, 2000.
 4. PEREIRA JÚNIOR, J. C. R. **AVEP - um ambiente virtual para apoio ao ensino de algoritmos e programação.** Dissertação (Mestrado) Programa de Pós-Graduação em Engenharia de Produção, Universidade Estadual do Norte Fluminense, Campos dos Goytacazes, 2006.
 5. PEREIRA JÚNIOR, J. C. R.; RAPKIEWICZ, C. E. ; DELGADO, C.; XEXÉO, J. A. M. **Ensino de algoritmos e programação: uma experiência no nível médio.** Anais do XXV Congresso da SBC, São Leopoldo, RS. v. 1, 2005.
 6. WEINBERG G. M. **The Psychology of Computer Programming: Silver Anniversary Edition.** Dorset House; Anl Sub edition, 292 páginas, 1998.
 7. XAVIER, G. M. C. et al. **Estudo dos Fatores que Influenciam a Aprendizagem Introdutória de Programação.** IV ERBASE – Escola regional de computação Bahia-Sergipe. Anais eletrônicos. Feira de Santana: Universidade Estadual de Feira de Santana, 2004.

2. VARIÁVEIS OBSERVADAS

Para observar a classe de comportamentos 'programar computadores' foi necessário definir quais as variáveis relevantes. Sendo o fenômeno de pesquisa uma classe de comportamentos, as variáveis que constituem qualquer unidade comportamental do processo de programar computadores e seus componentes foram observadas:

1. Unidades de comportamentos constituintes da classe geral de comportamentos 'programar computadores';

2. Classes de estímulos antecedentes de unidades de comportamento da classe geral 'programar computadores';
3. Classes de ações de unidades de comportamento da classe geral 'programar computadores';
4. Classes de estímulos consequentes de unidades de comportamento da classe geral 'programar computadores'.

3. SITUAÇÃO E MATERIAIS

O exame de documentos foi feito em ambiente silencioso, com tempo suficiente para retirar informações dos documentos e organizá-las em registros padronizados. Foi utilizado um computador para organizar os documentos em arquivos separados pelo nome do documento. As informações obtidas em cada documento foram organizadas em protocolos de registros de forma a possibilitar a inserção de trechos de informação do documento.

4. PROCEDIMENTO PARA REGISTRAR, ANALISAR E TRATAR DADOS

Foram seis etapas para o registro, análise e tratamento das informações:

- a) Seleção e transcrição de informações que continham indicação das variáveis explicitadas no item 2;
- b) Destaque de classes de comportamentos identificadas nas informações selecionadas;
- c) Decomposição das classes de comportamentos identificadas em classes de comportamentos mais simples ou intermediárias;
- d) Análise das classes de comportamentos identificadas;
- e) Identificação de cadeias de comportamentos;
- f) Organização das cadeias de comportamentos.

a) Seleção e transcrição de informações

Após a leitura das fontes de informação, foi feita uma transcrição para registro das informações de interesse, que foram selecionadas quando apresentavam no texto as variáveis definidas no item 2 (VARIÁVEIS OBSERVADAS). As informações que apresentam tais variáveis foram transcritas para um protocolo de registro em um computador, logo após serem identificadas. Na Figura 7 está apresentado um exemplo do protocolo de registro dos parágrafos selecionados:

Figura 7 Exemplo de protocolo de registro das informações selecionadas

Obra: Pereira Júnior, Rapkiewicz, Delgado e Xexeo - Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio

pg. 7: ... Para programar um computador é necessário utilizar uma especificação formal que nada mais é do que um conjunto de instruções a serem seguidas, as quais possam indicar alguma solução de um problema. Neste caso, trata-se de utilizar uma linguagem que seja bem interpretada, sem ambiguidades, ou seja, uma linguagem formal.

As informações de interesse dos documentos foram transcritas quando identificadas descrições diretas ou indiretas ou quando havia indicações de classes de comportamentos que compõem a classe geral 'programar computadores', componentes das classes de comportamentos, e classes de comportamentos pré-requisitos da classe geral programar computadores. Esse procedimento é baseado no trabalho de diversos pesquisadores que desenvolveram um método para identificar comportamentos a partir de informações em documentos (Botomé, 1975, 1994; Luiz, 2008; De Luca, 2008; Garcia; 2009; Kienen, 2008; Viecili, 2008;).

b) Destaque de classes de comportamentos identificadas nas informações selecionadas

As variáveis que controlaram a seleção e transcrição da informação para o protocolo de registro foram destacadas em negrito, como ilustrados na Figura 8:

Figura 8 Exemplo de um destaque de aspectos específicos da informação

Obra: Pereira Júnior, Rapkiewicz, Delgado e Xexeo - Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio

pg. 7: ... Para programar um computador é necessário **utilizar** uma especificação formal que nada mais é do que **um conjunto de instruções a serem seguidas**, as quais possam **indicar alguma solução de um problema**. Neste caso, trata-se de **utilizar uma linguagem que seja bem interpretada, sem ambiguidades, ou seja, uma linguagem formal**.

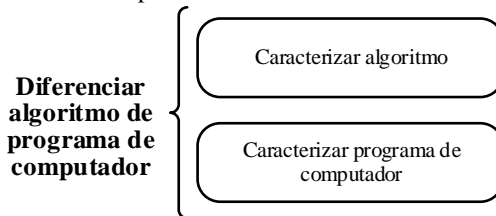
Para identificar no texto as classes de estímulos que constituem a situação antecedente foram considerados aspectos com as quais o sujeito precisa levar em conta para apresentar uma ação. Para identificar ações que compõem a classe de resposta e unidades de comportamento foram selecionados aspectos que indicassem coisas que o sujeito faz, usualmente sinalizado por um verbo, mesmo que substantivado ou apresentado na forma de gerúndio. As classes de estímulos que constituem a situação consequente foram selecionadas a partir da indicação no texto das decorrências das ações do sujeito, sociais, físicas, e imediatas ou a curto, médio e longo prazo. No exemplo da Figura 8, foram destacadas as classes de estímulos antecedentes: 'linguagem bem interpretada, sem ambiguidades, ou seja, uma linguagem formal' e 'conjunto de instruções a serem seguidas'. A classe de comportamento destacada foi 'utilizar uma linguagem formal', e como classe de estímulo consequente foi destacado 'solução de um problema', pois está indicado no texto que, ao utilizar uma linguagem formal (classe de resposta), que seja bem interpretada, sem ambiguidades (classe de estímulos antecedentes), estas devem produzir a solução de um problema (classe de estímulos consequentes).

Em muitos trechos selecionados, não foi possível identificar uma classe de comportamento completa, ou seja, com indicação de uma classe de estímulo antecedente, classe de resposta e classe de estímulo consequentes de unidades de comportamento da classe geral 'programar computadores'. Nessas situações, foi necessário pesquisar em outras fontes que apresentassem informações a respeito de uma mesma classe de comportamento e completar as lacunas identificadas.

c) Decomposição das classes de comportamentos identificadas

Weinberg (1998) considera que estudar a programação de computadores do ponto de vista psicológico exige a decomposição das atividades complexas que as caracterizam em entidades mais simples. Afirma que a natureza iterativa da programação torna sua decomposição um processo sofisticado, sem muitas fronteiras nítidas. Como exemplo da complexidade desse fenômeno, Weinberg afirma que, se for perguntado a programadores o que estão fazendo, um dirá que está 'escrevendo o código', outro que está 'debugando', etc. e ambos poderão estar realizando a mesma tarefa. A decomposição das classes de comportamentos foi feita para organizar essas diferentes 'tarefas' - as classes de comportamentos intermediárias e pré-requisitos da classe 'Programar computadores'. Foram considerados classes de comportamentos pré-requisitos aquelas necessárias para a ocorrência de uma classe de comportamento. As classes de comportamentos intermediárias se referem as classes de comportamentos componentes de uma cadeia e que precisam ser aprendidas para viabilizar a execução de qualquer elo de uma cadeia comportamental. Essa etapa auxiliou na identificação de cadeias de comportamentos e de classes de comportamentos pré-requisitos que constituíram outras cadeias.

Figura 9 Exemplo de procedimento para a etapa 'Decomposição das classes de comportamentos identificadas'



A decomposição de classes de comportamentos possibilita identificar classes de comportamentos mais específicas, como operações envolvidas em uma tarefa, manuseio de equipamentos, aspectos do meio com os quais se lida, procedimentos (como) para fazer o que precisa ser feito. Para fazer a decomposição de uma classe de comportamento é necessário identificar o que o aprendiz precisa estar apto a fazer para conseguir realizar essa classe de comportamento. (Botomé, 1975). No exemplo acima, o aprendiz do comportamento 'diferenciar algoritmo de um programa de computador' precisa ser capaz de caracterizar programa de computador e caracterizar algoritmo.

d) Análise das classes de comportamentos identificadas

A análise das classes de comportamentos identificadas foi feita para explicitar as classes de estímulos que controlam as ações apresentadas e os resultados que devem ser garantidos ao apresentar tais ações, como apresentado na Figura 10.

Figura 10 Exemplo de protocolo de registro para analisar as classes de comportamentos identificadas em conformidade com seus componentes: classe de estímulos antecedentes, classe de resposta e classe de estímulos consequentes

Obra: Pereira Júnior, Rapkiewicz, Delgado e Xexeo - Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio

pg. 7: ... Para programar um computador é necessário **utilizar** uma especificação formal que nada mais é do que um **conjunto de instruções a serem seguidas**, as quais possam **indicar alguma solução de um problema**. Neste caso, trata-se de **utilizar uma linguagem que seja bem interpretada, sem ambiguidades, ou seja, uma linguagem formal**.

Utilizar linguagem formal		
Classe de estímulos antecedentes	Classe de respostas	Classe de estímulos consequentes
Linguagem formal (linguagem bem interpretada, sem ambiguidades) Conjunto de instruções a serem seguidas <i>Problema a resolver</i>	Utilizar linguagem formal	Solução de um problema <i>Consequências a médio prazo: ...</i> <i>Consequências a longo prazo: ...</i>

No exemplo da Figura 10 é necessário, para utilizar linguagem formal, que uma linguagem formal faça parte da classe de estímulos antecedentes e que se produza como uma das consequências a solução de um problema. Ao analisar a classe de comportamento, foi possível complementar informações, como no exemplo, a classe de estímulo antecedente 'problema a resolver', destacada em itálico, que não estava explícita no trecho, porém, como a consequência imediata da classe de comportamento é a solução de um problema, é necessário explicitar tal classe de estímulo como constituinte da situação antecedente. Demais consequências, a médio e a longo prazo, podem ser explicitadas ao identificar outras

fontes de informação que complementem a análise da classe de comportamento.

e) Identificação das cadeias de comportamentos

A elaboração de cadeias de comportamentos (sequências temporais de classes de comportamentos, em que a classe de estímulo consequente da primeira classe de comportamento é a classe de estímulo antecedente da classe de comportamento seguinte) foi realizada a partir da análise e decomposição das classes de comportamentos. Ao identificar classes de estímulos consequentes que poderiam ser classes de estímulos antecedentes de outra classe de comportamento, estas foram encadeadas.

Na Tabela 2, a partir da segunda linha, cada linha representa um elo (uma classe de comportamento) da cadeia de comportamentos. Na primeira coluna estão escritos os números que indicam a ordem das classes de comportamentos. Na segunda coluna estão escritos os nomes das

Tabela 2. Exemplo de procedimento para a etapa 'Identificação das relações de cadeia'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar algoritmo	Características de algoritmos	Caracterizar algoritmo	Algoritmo caracterizado M: Aumento da probabilidade de caracterizar um algoritmo
2	Caracterizar programa de computador	Características de programas de computador Algoritmo caracterizado	Caracterizar programa de computador	Programa de computador caracterizado M: Aumento da probabilidade de caracterizar um programa de computador
3	Diferenciar algoritmo de programa de computador	Algoritmo caracterizado Programa de computador caracterizado	Comparar algoritmo com programa de computador	Algoritmo e programa de computador diferenciados

classes de comportamentos da cadeia. Na terceira coluna estão localizadas as diferentes classes de estímulos que constituem a situação que antecede as classes de respostas, apresentadas na quarta coluna. A última coluna contém as classes de estímulos que constituem a situação consequente. Tal cadeia de comportamento explicita as relações entre classe de estímulo consequente de uma classe de comportamento e classe de estímulo antecedente da classe de comportamento seguinte. As classes de estímulos antecedentes destacadas em negrito são as que mais importam para controlar a classe de resposta e as classes de estímulos consequentes em negrito são aquelas que têm mais probabilidade de serem percebidas pelo indivíduo e tornarem-se classe de estímulos antecedentes à classe de resposta no elo seguinte da cadeia.

Há três tipos de classes de estímulos consequentes, que foram classificados como classe de estímulo a curto prazo (em negrito), médio prazo (M) e longo prazo (L). As consequências a médio e longo prazo são outras que compõem a situação consequente e que também devem controlar a ocorrência da classe de resposta das classes de comportamentos da cadeia, ou estarem associadas como resultados (desejáveis) da ação apresentada.

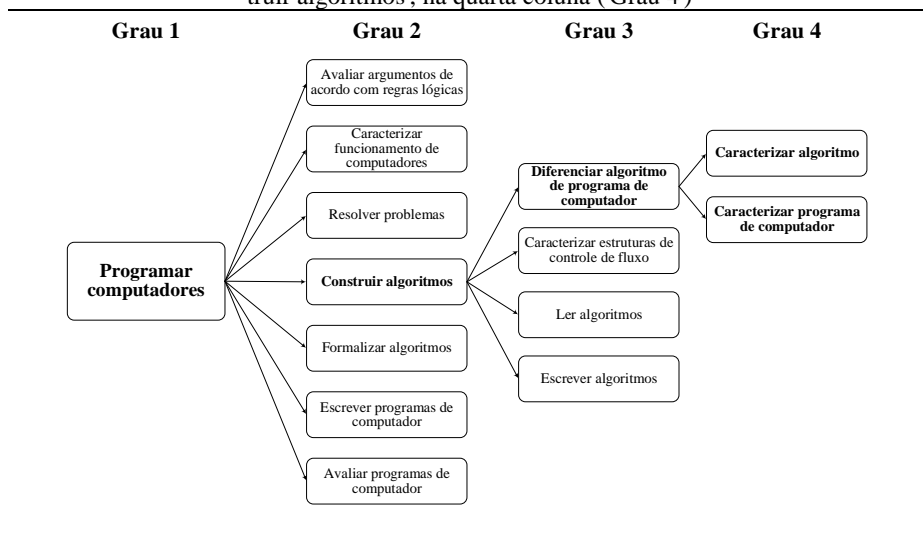
f) Organização das cadeias de comportamentos

Para organizar as cadeias de comportamentos foi necessário separá-las em diferentes graus de abrangência e organizar com base nas fases elaboradas por Delgado e outros (2005). Estes autores identificaram três fases de ensino de algoritmos: 'resolução de problemas'; 'formalização'; e 'construção de algoritmos' que foram utilizadas para organizar as cadeias de comportamentos. Por haver cadeias de comportamentos que não se encaixaram nas fases definidas por Delgado e outros (2005), foi necessário elaborar outras 'fases' intermediárias àquelas proposta por esses autores.

III RESULTADOS

O conceito de classe de comportamento foi utilizado para se referir ao processo 'Programar computadores'. Entender esse processo como uma classe de comportamento humano implica em identificar as características das ações que as pessoas realizam ao programar um computador, das situações em que essas ações ocorrem e as decorrências dessas ações ('consequências'), desde consequências imediatas, a curto, médio e longo prazo. O processo 'Programar computadores' foi definido como uma classe geral de comportamentos, composta por muitas outras classes de comportamentos em sequências e com características diferentes.

Figura 11 Graus de abrangência da classe geral 'Programar computadores' e exemplo de uma sequência de classes de comportamentos denominada 'Construir algoritmos', na quarta coluna ('Grau 4')



A organização da classe geral 'programar computadores' foi feita em quatro níveis ou graus e está ilustrada na Figura 11. Foram identificadas sete grandes categorias (segunda coluna, 'grau 1'), organizadas em uma sequência que representa desde classes de comportamentos consideradas pré-requisitos gerais para programar computadores (como 'Avaliar argumentos de acordo com regras lógicas' e 'Caracterizar funcionamento

de computadores') até classes de comportamentos que representam a avaliação de programas de computador. Na Figura 11 há também um exemplo de uma sequência de classes de comportamentos destacada em negrito, na última coluna.

As sete categorias encontradas foram: (1) Avaliar argumentos de acordo com regras lógicas; (2) Caracterizar funcionamento de computadores; (3) Resolver problemas; (4) Construir algoritmos; (5) Formalizar algoritmos; (6) Escrever programas de computador; (7) Avaliar programas de computador. A organização das classes de comportamentos da classe geral 'Programar computadores' nos três primeiros graus de abrangência está apresentada na Tabela 3.

Tabela 3. Graus de abrangência da classe geral de comportamento 'Programar computadores'

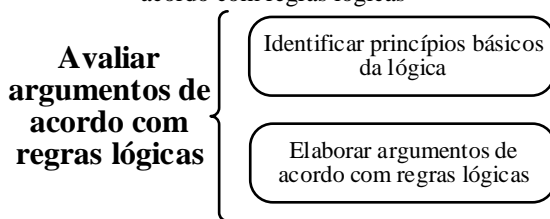
Grau 1	Grau 2	Grau 3
Programar computadores	Avaliar argumentos de acordo com regras lógicas	--
	Caracterizar funcionamento de computadores	--
	Resolver problemas	<ol style="list-style-type: none"> 1. Caracterizar problemas 2. Delimitar escopo do problema 3. Representar problemas 4. Decompor problemas 5. Escrever solução de problemas
	Construir algoritmos	<ol style="list-style-type: none"> 1. Diferenciar algoritmo de programa de computador 2. Caracterizar estruturas de controle de fluxo 3. Ler algoritmos 4. Escrever algoritmos
	Formalizar algoritmos	<ol style="list-style-type: none"> 1. Identificar equivalência de instruções em linguagem natural e de programação 2. Caracterizar descrições formais de problemas 3. Especificar detalhes formais de um algoritmo
	Escrever programas de computador	<ol style="list-style-type: none"> 1. Nomear variáveis 2. Declarar variáveis 3. Atribuir valores às variáveis 4. Caracterizar processo de programas de computadores 5. Projetar programas de computador 6. Escrever códigos em uma linguagem de programação
	Avaliar programas de computador	<ol style="list-style-type: none"> 1. Detectar erros 2. Depurar erros 3. Avaliar qualidade de programas de computador

As sete categorias do grau 2 ilustrada na Tabela 3 estão em uma ordem que obedece a uma possível sequência para o ensino da classe geral de comportamentos denominada 'Programar computadores'. Tais categorias foram decompostas em outras que estão apresentadas na terceira coluna, denominada 'grau 3', sendo cinco categorias em 'Resolver problemas', cinco em 'Construir algoritmos', três em 'Formalizar algoritmos', seis em 'Escrever programas de computador' e três em 'Avaliar programas de computador'. O grau 4 de abrangência da classe geral de comportamentos 'Programar computadores' não está representado nessa tabela. As categorias 'Avaliar argumentos de acordo com regras lógicas' e 'Caracterizar funcionamento de computadores' não apresentam outras subdivisões mas apenas as classes de comportamentos em sequência do quarto grau de abrangência. As próximas seções apresentam as classes de comportamentos que formam cada categoria do grau 2.

1. CATEGORIA DE COMPORTAMENTOS 'AVALIAR ARGUMENTOS DE ACORDO COM REGRAS LÓGICAS'

A primeira categoria de comportamentos do grau de abrangência 2 é denominada 'Avaliar argumentos de acordo com regras lógicas'. Esta categoria pode ser considerada como pré-requisito para programar computadores, pois possibilita desenvolver raciocínio lógico, necessário para elaborar soluções de problemas em uma linguagem de programação.

Figura 12 Sequência de classes de comportamentos de 'Avaliar argumentos de acordo com regras lógicas'



Na Figura 12 as classes de comportamentos da categoria 'Avaliar argumentos de acordo com regras lógicas', referentes ao 4º grau de abrangência da classe geral de comportamentos são apresentadas: 'Identificar princípios básicos da lógica' e 'Elaborar argumentos de acordo com regras lógicas'.

Tabela 4. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Avaliar argumentos de acordo com regras lógicas'

	Classe de comportamentos	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Identificar princípios básicos da lógica	Princípios básicos da lógica	Identificar princípios básicos da lógica	Princípios básicos da lógica identificados
2	Elaborar argumentos de acordo com regras lógicas	Princípios básicos da lógica identificados Conectivos lógicos	Elaborar argumentos de acordo com regras lógicas	Argumentos elaborados de acordo com regras da lógica M: Aumento da probabilidade de escrever sentenças lógicas em programas de computador
3	Avaliar argumentos de acordo com regras lógicas	Argumentos elaborados de acordo com regras da lógica	Avaliar argumentos de acordo com regras lógicas	Argumentos avaliados M: Aumento da probabilidade de desenvolver raciocínio lógico

Na Tabela 4 estão os componentes das classes de comportamentos que se referem a 'Avaliar argumentos de acordo com regras lógicas'. Na primeira classe de comportamento, os princípios básicos da lógica compõem a classe de estímulos da situação antecedente e se referem aos princípios da identidade, da não-contradição e do terceiro excluído³. Há outros princípios que também são importantes para a elaboração de um raciocínio correto que podem ser explorados a partir dessa classe de comportamento. Por considerar que há várias possibilidades de resposta específicas, estas não foram especificadas na coluna 'classe de resposta', sendo

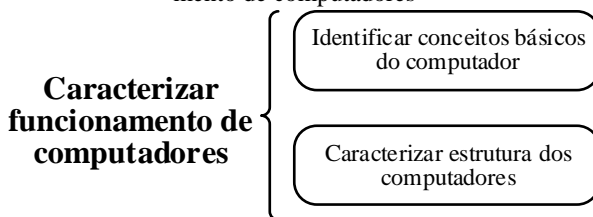
³ O princípio da identidade afirma que 'A' é igual a 'A' e não pode ser 'B': se refere à formação de conceitos, pois representa a identidade das coisas. O princípio da não-contradição afirma que 'A' é igual a 'A' e nunca pode ser 'não-A', refere-se a afirmação de que uma proposição não pode ser verdadeira e falsa ao mesmo tempo. Junto ao princípio da identidade este princípio é essencial para que seja possível formar conceitos. O princípio do terceiro excluído afirma que uma alternativa ou é verdadeira ou falsa. Ou 'A' é 'x' ou 'A' não é 'x' e não há uma terceira possibilidade (Paixão, 2007)

que o nome da classe de comportamento representa de forma geral a classe de resposta. Com a identificação desses princípios, é possível elaborar argumentos de acordo com regras lógicas. Tal classe de comportamento tem como consequência a médio prazo o aumento da probabilidade de escrever sentenças lógicas em programas de computador. A ocorrência da última classe de comportamento depende das duas primeiras classes de comportamentos ocorrerem e a execução de toda a cadeia aumenta a probabilidade de desenvolver raciocínio lógico, condição necessária para resolver problemas, construir e formalizar algoritmos e escrever programas de computador, outras categorias a serem apresentadas a seguir.

2. CATEGORIA DE COMPORTAMENTOS 'CARACTERIZAR FUNCIONAMENTO DE COMPUTADORES'

Caracterizar o funcionamento de computadores é essencial, pois este é o objeto onde ocorre o processo de programar computadores. Esta é a segunda categoria de comportamentos do grau 2 de abrangência da classe geral de comportamentos 'programar computadores' e está representada pelo conjunto de classes de comportamentos da Figura 13.

Figura 13 Sequência de classes de comportamentos de 'Caracterizar funcionamento de computadores'



A primeira classe de comportamento é 'Identificar conceitos básicos do computador' seguido da classe 'Caracterizar estrutura dos computadores'. A cadeia de comportamentos formada a partir desse conjunto está apresentada na Tabela 5.

O primeiro elo da cadeia é a classe de comportamento 'Identificar conceitos básicos do computador'. Esta classe de comportamento tem como classe de estímulo antecedente 'Conceitos básicos do computador'. Um computador é dividido em duas partes básicas, o *hardware* e o *software*. O *software* é o programa de computador propriamente dito, por

exemplo, sistemas operacionais (Linux, Windows, ...), programas ou aplicativos, como os editores de texto (Microsoft Word, Open Office, ...), jogos, navegadores de internet (Mozilla Firefox, Google Chrome, Internet Explorer, ...). O *hardware* é a parte física do computador, formada por componentes eletrônicos, circuitos integrados, placa-mãe, processador, memória, disco rígido entre outros. Ao identificar esses conceitos, a relação de dependência entre hardware e software fica mais evidente. O software necessita da parte física, denominada hardware, para existir. É o software que torna possível a comunicação das diferentes partes físicas do hardware.

Tabela 5. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar funcionamento de computadores'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Identificar conceitos básicos do computador	Conceitos básicos do computador	Identificar conceitos básicos do computador	Conceitos básicos do computador identificados M: Aumento da probabilidade de identificar a relação de dependência entre hardware e software
2	Caracterizar estrutura dos computadores	Conceitos básicos do computador identificados Diferentes componentes da estrutura de um computador	Caracterizar estrutura dos computadores	Estrutura dos computadores caracterizada
3	Caracterizar funcionamento de computadores	Estrutura dos computadores caracterizada Características do funcionamento do computador	Caracterizar funcionamento de computadores	Funcionamento de computadores caracterizado M: Aumento da probabilidade de diferenciar uso do computador como editor ou executor de programas

Na segunda classe de comportamento, a classe de estímulos que constitui a situação antecedente é formada pela classe de estímulo consequente do elo anterior e por 'Diferentes componentes da estrutura de um

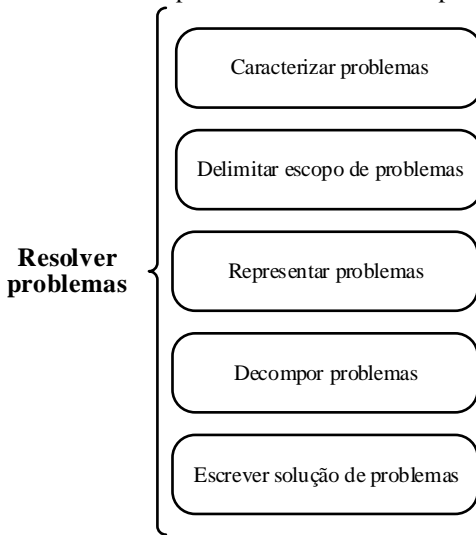
computador', por exemplo, o processador, memória, dispositivos removíveis de armazenamento de dados (*pen drives, HDs externo, DVDs, etc.*) e dispositivos de entrada e saída (*mouse, teclado, impressora, monitor, etc.*) (Horstmann, 2003), todos componentes do *hardware*, a parte física do computador. A partir dessas classes de estímulos, a classe de resposta pode ser quaisquer respostas que produzam como estímulo consequente a curto prazo a caracterização da estrutura dos computadores.

Com os conceitos do computador identificados e sua estrutura caracterizada, é possível 'Caracterizar funcionamento de computadores', terceiro comportamento da cadeia. Este comportamento contém como estímulos antecedentes o estímulo consequente do elo anterior e o estímulo 'Características do funcionamento do computador'. Em termos do seu funcionamento, o computador pode ser definido como uma máquina programável, que executa ordens e armazena dados (Boratti, 2004). Em um computador, o processador controla e movimenta dados, localiza e executa programas e realiza operações aritméticas. A memória, onde ocorre o armazenamento de dados, pode ser caracterizada como um conjunto de posições com uma identificação (um nome) e com um determinado valor armazenado (Boratti, 2004). Além da memória, há os dispositivos de entrada, que são os meios por onde se insere dados, e os dispositivos de saída, que são os meios por onde o computador fornece os dados (Horstmann, 2003). Um tipo de decorrência a médio prazo desse comportamento é a diferenciação entre utilizar um computador para editar programas, ou seja, com a função de programá-lo, criar programas e utilizá-lo para executar programas (usar programas prontos).

3. CATEGORIA DE COMPORTAMENTOS 'RESOLVER PROBLEMAS'

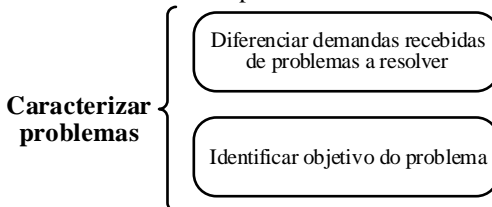
Resolver problemas se caracteriza por mais que apenas a emissão de uma resposta que produz uma solução. É uma complexa cadeia de comportamentos, sendo que esses comportamentos formam passos necessários para tornar mais provável a ocorrência da solução do problema (Skinner, 1974, pg. 98). Em relação à categoria 'Resolver problemas' há diversas cadeias de comportamento ('passos') identificadas que aumentam a probabilidade de chegar a solução de problemas em geral.

Figura 14 Classes de comportamentos de 'Resolver problemas'



Na Figura 14 estão organizadas as categorias do 3º grau de abrangência de 'Resolver problemas'. Cada categoria foi criada e nomeada orientando-se pela união de classes de comportamentos com características semelhantes, indicadas por expressões sinônimas e pela similaridade das consequências constituintes das expressões. Inicia-se pela categoria 'Caracterizar problemas', 'Delimitar escopo de problemas' e 'Representar problemas'. Em seguida vem a categoria 'Decompor problemas' e a última categoria, 'Escrever solução de problemas'.

Figura 15 Sequência de classes de comportamentos de 'Caracterizar problemas'



Na Figura 15 está a sequência de classes de comportamentos de 'Caracterizar problemas', que se inicia com a classe de comportamento 'Diferenciar demandas recebidas de problemas a resolver', seguido da classe de comportamento 'Identificar objetivo do problema'.

Tabela 6. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar problemas'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Diferenciar demandas recebidas de problemas a resolver	Diferentes demandas apresentadas como problemas a resolver	Diferenciar demandas recebidas de problemas a resolver	Diferenciação entre demanda e problema M: Aumento da probabilidade de derivar, a partir de demandas, problemas a resolver L: Aumento da probabilidade de identificar problemas expressos em diferentes formas
2	Identificar objetivo do problema	Diferenciação entre demanda e problema	Identificar objetivo do problema	Objetivo do problema identificado L: Aumento da probabilidade de selecionar problemas relevantes para resolver
3	Caracterizar problemas	Objetivo do problema identificado Características nucleares do problema	Caracterizar problemas	Características nucleares do problema caracterizados M: Aumento da probabilidade de identificar o que é nuclear em um problema L: Aumento da probabilidade de produzir solução adequada para um problema

Na Tabela 6 estão apresentadas as classes de comportamentos com os componentes que constituem os elos da cadeia 'Caracterizar problemas'. A primeira classe de comportamento dessa cadeia apresenta como classe de estímulos que constituem a situação antecedente 'Diferentes demandas apresentadas como problemas a resolver'. Demanda é diferente de problema, pois naquela o problema não está bem explicitado. As possibilidades de resolução do problema dependem de uma identificação do que importa e é preciso fazer, mais do que resolver uma demanda apresentada pelo usuário como problema pronto. A classe de resposta desse comportamento pode ser quaisquer respostas que produzam como consequência a 'Diferenciação entre demanda e problema', além de produzir a médio prazo o 'Aumento da probabilidade de derivar, a partir de demandas, problemas a resolver' e a longo prazo o 'Aumento da probabilidade de identificar problemas expressos em diferentes formas'. Após diferenciar demanda de problema ocorre a classe de comportamento 'Identificar

objetivo do problema', que produz o aumento da probabilidade de selecionar problemas relevantes para resolver. A terceira classe de comportamento se refere à caracterização do problema, por exemplo, os dados do problema a serem obtidos para realizar cálculos, os resultados esperados dos cálculos, etc. Como estímulos consequentes são apresentados 'Características nucleares do problema caracterizados', a médio prazo o 'Aumento da probabilidade de identificar o que é nuclear em um problema' e o 'Aumento da probabilidade de produzir solução adequada para um problema', a longo prazo.

Na Tabela 7 está apresentada a classe de comportamento 'Delimitar escopo do problema'. O 'escopo' do problema se refere ao alcance e validade de expressões e variáveis do problema, seus limites.

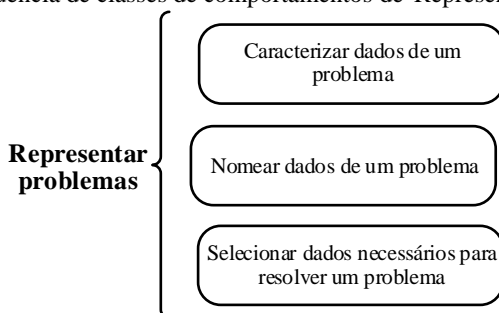
Tabela 7. Classe de comportamento 'Delimitar escopo do problema' e seus componentes

Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
Delimitar o escopo do problema	Requisitos do problema	Delimitar o escopo do problema	Escopo do problema delimitado M: Diminuição da probabilidade de erros nas etapas seguintes de elaboração da solução do problema L: Aumento da probabilidade de decidir como decompor o problema

Na classe de estímulos antecedentes está 'Requisitos de um problema', que se refere às condições e exigências que necessitam ser garantidas para resolver um problema. Os requisitos de um problema ajudam na definição do que será resolvido e o que não será. Por exemplo, para projetar uma máquina de ressonância magnética, não é preciso projetar uma maca, porém, uma maca é indispensável para levar os pacientes até a máquina, é, portanto, um requisito a ser considerado para projetar a máquina. Essa classe de comportamento produz como consequência a diminuição da probabilidade de erros nas etapas seguintes de elaboração da solução do problema e o aumento da probabilidade de decidir como decompor o problema, facilitando a separação correta do problema em subproblemas.

A sequência da Figura 16 contém classes de comportamentos relacionados à representação de um problema e inicia com a caracterização e nomeação dos dados de um problema, em seguida ocorre a seleção dos dados necessários para resolver um problema.

Figura 16 Sequência de classes de comportamentos de 'Representar problemas'



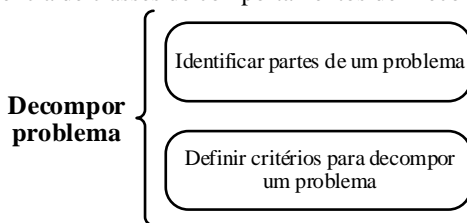
Os componentes das classes de comportamentos da cadeia 'Representar problemas' estão apresentados na Tabela 8. Caracterizar dados de um problema contém como classe de estímulos antecedentes os tipos básicos de dados de um problema (dados de entrada, dados de saída e dados a serem calculados). Após a caracterização dos dados do problema ocorre sua nomeação. É necessário dar nomes para estruturar a solução do problema e ser possível manipular os dados. Por exemplo, um problema que tem em seu enunciado uma referência à quantidade de passagens vendidas em um ônibus poderá ter esse dado nomeado como 'lotação do ônibus'. A seguir os dados que importam para resolver um problema são selecionados. Por exemplo, em um problema para calcular o valor do salário líquido de um funcionário os dados necessários para resolver o problema seriam o valor do salário bruto, as horas extras, etc. e os dados irrelevantes poderiam ser o nome, o sexo, a cor do cabelo do funcionário. É possível que em um enunciado não estejam claros quais dados são necessários para produzir a solução do problema. Por isso, a seleção dos dados necessários aumenta a probabilidade de analisar problemas e de representá-los em seus aspectos essenciais. Após a seleção do que importa para resolver um problema, é necessário representar tal problema. Há diferentes possibilidades de representar um problema, por exemplo, por meio de fluxogramas, em linguagem natural (como texto descritivo), em linguagem matemática, em uma representação gráfica etc.

Tabela 8. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Representar problemas'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar dados de um problema	Diferentes tipos de dados de um problema	Caracterizar dados de um problema	Dados do problema caracterizados
2	Nomear dados de um problema	Dados do problema caracterizados	Nomear dados de um problema	Dados de um problema nomeados
3	Selecionar dados necessários para resolver um problema	Dados de um problema nomeados	Selecionar dados necessários para resolver um problema	Dados necessários para a resolução do problema selecionados M: Aumento da capacidade de analisar problemas M: Aumento da probabilidade de representar o problema em seus aspectos essenciais
4	Representar problemas	Dados necessários para a resolução do problema selecionados	Representar problemas de diferentes formas	Representações de problemas construídas M: Aumento da probabilidade de construir a solução de um problema

Na Figura 17 está apresentada a sequência 'Decompor problema'. Para decompor um problema é necessário que as partes do problema sejam identificadas (primeira classe de comportamento) e que os critérios para decompor um problema sejam definidos (segunda classe de comportamento).

Figura 17 Sequência de classes de comportamentos de 'Decompor problema'



A Tabela 9 contém os componentes das classes de comportamentos dessa cadeia.

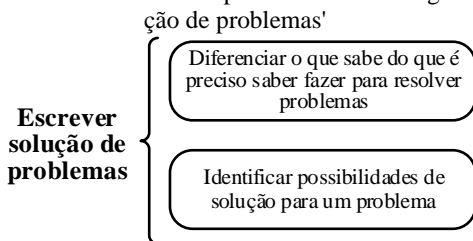
Tabela 9 Conjunto das três classes de comportamentos e seus componentes da cadeia 'Decompor problemas'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Identificar partes de um problema	Problema a resolver	Identificar partes de um problema	Partes de um problema identificadas
2	Definir critérios para decompor um problema	Partes de um problema identificadas Diferentes critérios que orientam a decomposição de um problema	Definir critérios para decompor um problema	Critérios para decompor problema definidos M: Aumento da probabilidade de identificar o que fazer para resolver um problema
3	Decompor problema	Critérios para decompor problema definidos	Decompor problemas em problemas de menor complexidade, com partes pequenas, únicas e não ambíguas que são resolvidas de forma independente	Problema decomposto em subproblemas M: Diminuição da complexidade do problema M: Aumento da probabilidade de resolver cada parte do problema de forma independente

A primeira classe de comportamento, 'Identificar partes de um problema', apresenta como classe de estímulo antecedente 'problema a resolver' e é pré-requisito para a classe de comportamento denominado 'Definir

critérios para decompor um problema'. Tais critérios se referem à definição de unidades de um problema (partes pequenas, únicas e não ambíguas), que podem ser representadas e resolvidas em uma única instrução. Como consequência a médio prazo há um aumento da probabilidade de identificar o que fazer para resolver um problema. A última classe de comportamento se refere ao próprio processo de decomposição de um problema em problemas de menor complexidade (ou 'subproblemas'). A partir da ocorrência dessa classe de comportamento, há uma diminuição da complexidade do problema e aumento da probabilidade de resolução de cada parte dele de forma independente. Depois de ter o problema caracterizado, delimitado, representado e decomposto, é possível a ocorrência da sequência 'Escrever solução de problemas', apresentada na Figura 18, representada pelas classes de comportamentos: 'Diferenciar o que sabe do que é preciso saber fazer para resolver problemas' e 'Identificar possibilidades de solução para um problema'.

Figura 18 Sequência de classes de comportamentos da categoria 'Escrever solução de problemas'



A Tabela 10 contém os classes de comportamentos e seus componentes, evidenciando as relações em cadeia, em que o estímulo consequente do comportamento anterior é estímulo antecedente do próximo comportamento.

Ao diferenciar o que se sabe do que é preciso saber para resolver problemas há um aumento da probabilidade de identificar lacunas e ampliar o repertório para resolver problemas. É possível desenvolver diferentes soluções para um mesmo problema, logo identificar essas possibilidades aumenta a variabilidade de soluções que poderão ser desenvolvidas (segunda classe de comportamento). A classe de comportamento 'Escrever solução de problemas' se refere a problemas com diferentes graus de complexidade, desde aqueles que exigem cálculos matemáticos simples, como equações de primeiro grau, problemas que exigem a elaboração de condições lógicas, ou que considerem diferentes decisões a tomar, elaborar estruturas de repetição, equações, etc. Ao escrever solução para

tais tipos de problemas, há um aumento da autonomia para elaborar soluções e abstrair uma solução elaborada para outros contextos.

Tabela 10. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Escrever solução de problemas'

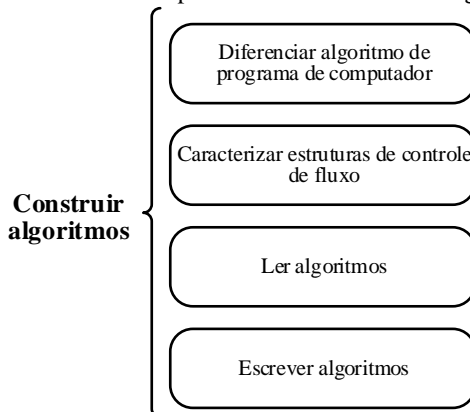
	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Diferenciar o que sabe do que é preciso saber fazer para resolver problemas	Enunciado de um problema Comportamentos que o indivíduo já apresenta para resolver problemas Comportamentos a apresentar para resolver problemas	Diferenciar o que sabe do que é preciso saber fazer para resolver problemas	Diferenciação do que se sabe do que é preciso saber fazer para resolver problemas Aspectos importantes para resolver o problema destacados M: Aumento da probabilidade de identificar lacunas no próprio repertório de resolução de problemas L: Aumento da probabilidade de ampliar repertório para resolver problemas
2	Identificar possibilidades de solução para um problema	Diferenciação do que se sabe do que é preciso saber fazer para resolver problemas Aspectos importantes para resolver o problema destacados Diferentes soluções para os problemas	Identificar possibilidades de solução para um problema	Possibilidades de solução para um problema identificadas M: Aumento da probabilidade de desenvolver soluções variadas
3	Escrever solução de problemas	Diferentes possibilidades de solução para um problema identificadas Diferentes graus de complexidade de um problema	Escrever solução de problemas de diferentes graus de complexidade	Problemas de diferentes graus de complexidade escritos M: Aumento da autonomia na elaboração de soluções M: Aumento da probabilidade de construir abstrações com a solução elaborada

A categoria 'Resolver problemas' é considerada um pré-requisito para a categoria 'Construir algoritmos'. Sem desenvolver as cadeias de caracterizar problemas, delimitar seu escopo, representar, decompor e escrever solução de problemas, é pouco provável construir algoritmos, pois uma parte do que caracteriza o processo de 'Construir algoritmos' é a explicitação dos passos necessários para resolver problemas, que será apresentado na próxima seção.

4. CATEGORIA DE COMPORTAMENTOS 'CONSTRUIR ALGORITMOS'

Um algoritmo é compreendido como uma sequência de instruções, em geral pequenas, simples que contém as etapas ou 'passos' para produzir a solução de um problema (Boratti, 2004, Forbellone & Eberspächer, 2005, Sipser, 2005, Ascencio & Campos, 2007, entre outros). Construir algoritmos é uma classe geral de comportamentos que pode ser subdividida em várias classes de comportamentos de menor complexidade e com características semelhantes.

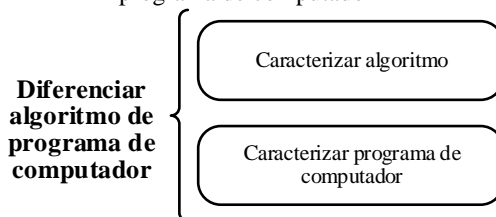
Figura 19 Classes de comportamentos de 'Construir algoritmos'



Tal categoria é composta das classes de comportamentos 'Diferenciar algoritmo de programa de computador', 'Caracterizar estruturas de controle de fluxo', que se refere às diversas estruturas que alteram o fluxo de execução de um algoritmo, 'Ler algoritmos' e 'Escrever algoritmos',

apresentadas na Figura 19. Para ser possível programar computadores é necessária a ocorrência dessa etapa, pois nela o aluno torna-se capaz de explicitar de forma mais organizada o que é preciso fazer para estruturar a solução de um problema de forma mais próxima de um programa de computador.

Figura 20 Sequência de classes de comportamentos de 'Diferenciar algoritmo de programa de computador'



Na Figura 20 está apresentada a sequência de classes de comportamentos denominada 'Diferenciar algoritmo de programa de computador'. Para um aluno ser capaz de diferenciar um algoritmo de um programa de computador, é necessário ocorrer caracterizar o que é um algoritmo e o que é um programa de computador.

Tabela 11. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Diferenciar algoritmo de programa de computador'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar algoritmo	Características de algoritmos	Caracterizar algoritmo	Algoritmo caracterizado M: Aumento da probabilidade de caracterizar um algoritmo
2	Caracterizar programa de computador	Características de programas de computador Algoritmo caracterizado	Caracterizar programa de computador	Programa de computador caracterizado M: Aumento da probabilidade de caracterizar um programa de computador
3	Diferenciar algoritmo de programa de computador	Algoritmo caracterizado Programa de computador caracterizado	Diferenciar algoritmo de programa de computador	Algoritmo e programa de computador diferenciados

A Tabela 11 contém as três classes de comportamentos e seus componentes dessa cadeia. Para caracterizar algoritmo, na classe de estímulos antecedentes estão suas características, que se referem a sua estrutura básica, sua função e características. Para caracterizar um programa de computador, as características dos programas necessitam ser descritas, que são, por exemplo, as palavras reservadas de uma determinada linguagem de programação, paradigma de programação utilizado no programa, a função ou objetivo de um programa de computador etc. Tanto a caracterização de algoritmo quanto o de programa de computador são classe de estímulos para fazer a diferenciação desses processos.

Tabela 12. Classe de comportamento 'Caracterizar estruturas de controle de fluxo' e seus componentes

Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
Caracterizar estruturas de controle de fluxo	Diferentes estruturas de controle de fluxo Conhecimento de como elaborar sequência de instruções	Caracterizar estruturas de controle de fluxo	Estruturas de controle de fluxo caracterizadas M: Aumento da probabilidade de identificar quando usar cada estrutura M: Aumento da probabilidade de elaborar algoritmos concisos e eficientes M: Aumento da probabilidade de compreender o algoritmo escrito

Na Tabela 12 estão apresentados os componentes da classe de comportamento 'Caracterizar estruturas de controle de fluxo'. Na classe de estímulos da situação antecedente há os estímulos 'diferentes estruturas de controle de fluxo' e 'conhecimento de como elaborar sequência de instruções'. As principais estruturas de controle de fluxo são: estruturas de seleção, de repetição, de sequência. O controle de fluxo se refere à como as instruções escritas são executadas, em que ordem, quais repetições e decisões a serem executadas. Cada estrutura de controle de fluxo apresenta diferentes funções. Por exemplo, as estruturas de seleção têm como função ajudar na decisão entre duas ou mais possibilidades de continuidade da execução do algoritmo, ou seja, quando há a possibilidade de executar um ou outro conjunto de instruções. No caso das estruturas de repetição, estas servem para executar um bloco de instruções em uma quantidade qualquer de vezes, o que produz a diminuição do tamanho do

algoritmo escrito. Ao caracterizar as estruturas de controle de fluxo há um aumento da probabilidade de identificar quando o uso de cada estrutura é adequado, aumento da probabilidade de elaborar algoritmos concisos e eficientes e de compreender o algoritmo escrito.

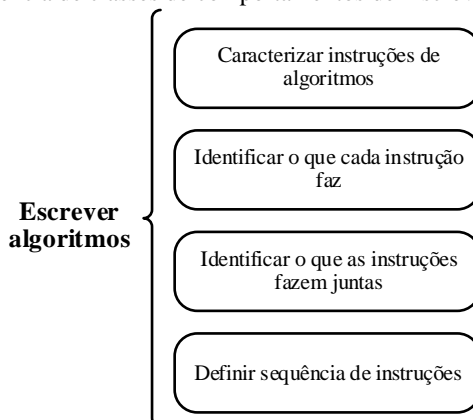
Tabela 13 Classe de comportamento 'Ler algoritmos' e seus componentes

Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
Ler algoritmos	Instruções de um algoritmo	Ler instruções de um algoritmo	Algoritmo lido M: Aumento da probabilidade de escrever algoritmos M: Aumento da probabilidade de interpretar algoritmos

A Tabela 13 apresenta o comportamento denominado 'Ler algoritmos', que tem como estímulo antecedente 'Instruções de um algoritmo'. A classe de resposta 'Ler instruções de um algoritmo' tem como estímulo consequente a curto prazo 'Algoritmo lido' e a médio prazo 'Aumento da probabilidade de escrever algoritmos' e 'Aumento da probabilidade de interpretar algoritmos'.

'Escrever algoritmos' é a última sequência de classes de comportamentos da categoria 'Construir algoritmos' e está apresentada na Figura 21.

Figura 21 Sequência de classes de comportamentos de 'Escrever algoritmos'



A primeira classe de comportamento da sequência é 'Caracterizar instruções de algoritmos', a segunda classe de comportamento é 'Identificar o que cada instrução faz', seguido das classes de comportamentos 'Identificar o que as instruções fazem juntas' e 'Definir sequência de instruções'. Na Tabela 14 os componentes dessas classes de comportamento são apresentados.

Tabela 14. Conjunto das cinco classes de comportamentos e seus componentes da cadeia 'Escrever algoritmos'

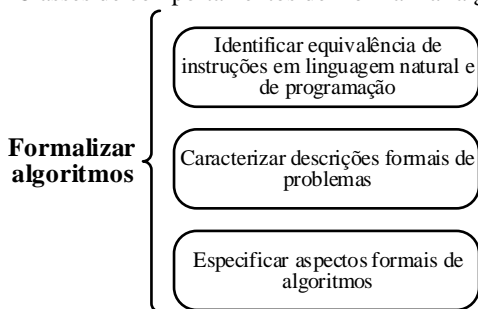
	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar instruções de algoritmos	Tipos de instruções	Caracterizar instruções de algoritmos	Instruções de algoritmos caracterizadas
2	Identificar o que cada instrução faz	Instruções de algoritmos caracterizadas Atividades de cada instrução	Identificar o que cada instrução faz	Conhecimento do que cada instrução faz M: Aumento da probabilidade de identificar erros no algoritmo M: Aumento da probabilidade de ler adequadamente um algoritmo
3	Identificar o que as instruções fazem juntas	Conhecimento do que cada instrução faz	Identificar o que as instruções fazem juntas	Conhecimento do que as instruções fazem juntas M: Aumento da clareza do processo de execução de um algoritmo M: Aumento da probabilidade de caracterizar uma sequência de instruções
4	Definir sequência de instruções	Conhecimento do que as instruções fazem juntas Problema caracterizado Requisitos do problema identificados Problema decomposto Instruções de um algoritmo	Definir sequência de instruções pequenas, únicas e não ambíguas	Sequência de instruções definida M: Aumento da probabilidade de entender as estruturas de controle de fluxo
5	Escrever algoritmos	Sequência de instruções definida Necessidade de descrever soluções que sejam entendidas por qualquer pessoa	Escrever algoritmos	Algoritmo construído

Para caracterizar instruções em um algoritmo (primeira classe de comportamento) é necessário conter na situação antecedente os diferentes tipos de instruções. Entre esses estão instruções do tipo declaração referencial (afirmações de fato: 'Isso é X') e instruções do tipo declaração conotativa (dar instruções: 'Encontre a inversa desta matriz'). Na segunda classe de comportamento, ao identificar o que cada instrução faz, há como classe de estímulos consequentes a médio prazo um aumento da probabilidade de identificar erros no algoritmo e de ler adequadamente um algoritmo. Após a identificação do que cada instrução faz, ocorre a classe de comportamento 'Identificar o que as instruções fazem juntas'. Essa classe de comportamento produz como classe de estímulo consequente um aumento da clareza do processo de execução de um algoritmo e aumento da probabilidade de caracterizar uma sequência de instruções. Para 'Construir algoritmos' é necessário definir uma sequência de instruções pequenas, únicas e não ambíguas. Como classe de estímulos que compõem a situação antecedente dessa classe de comportamento estão alguns estímulos da classe 'Resolver problemas'. A classe de estímulo consequente a médio prazo dessa classe de comportamento se refere ao aumento da probabilidade de entender as estruturas de controle de fluxo, pois estas se referem às diferentes formas que as sequências de instruções são arranjadas. Após as instruções terem sido definidas, é possível 'Escrever algoritmos' cuja classe de resposta pode quaisquer respostas que produzam o algoritmo construído.

5. CATEGORIA DE COMPORTAMENTOS 'FORMALIZAR ALGORITMOS'

Terminada a categoria 'Construir algoritmos', a próxima etapa está representada pela categoria 'Formalizar algoritmos'. Ao ter os passos para a solução de um problema elaborados em forma de algoritmo, é necessário transformá-los em uma linguagem mais próxima das linguagens de programação, pelo processo denominado 'formalização'. Nesse processo são aplicadas regras que eliminam ambiguidades e imprecisões, de forma a tornar mais fácil escrever um programa de computador.

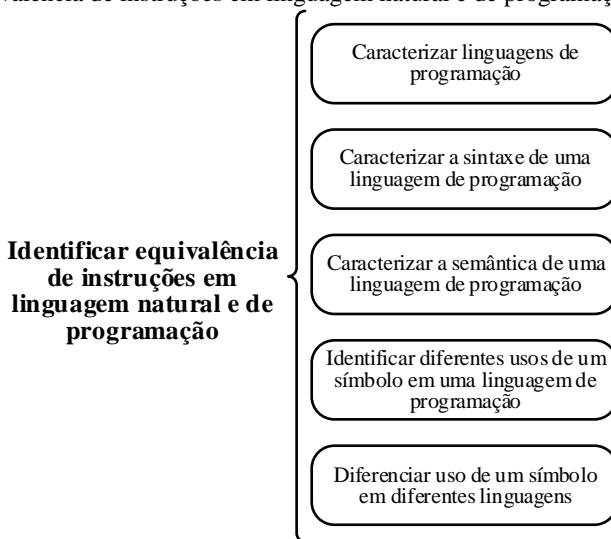
Figura 22 Classes de comportamentos de 'Formalizar algoritmos'



Na Figura 22 há estão representadas as classes de comportamentos pertencentes à classe geral 'Formalizar algoritmos'. A formalização de um algoritmo passa pelas etapas de identificação da equivalência de instruções em linguagem natural e de programação, caracterização de descrições formais e, por fim, pela especificação dos aspectos formais de algoritmos.

Na Figura 23 estão contidos as classes de comportamentos da sequência 'Identificar equivalência de instruções em linguagem natural e de programação'. Essa sequência tem como primeira classe de comportamento 'Caracterizar linguagem de programação'. Em seguida ocorrem as classes de comportamentos referentes à caracterização sintática e semântica das linguagens de programação. 'Identificar diferentes usos de um símbolo em uma linguagem de programação' e 'Diferenciar uso de um símbolo em diferentes linguagens' são as próximas classes de comportamentos.

Figura 23 Sequência de classes de comportamentos da categoria 'Identificar equivalência de instruções em linguagem natural e de programação'.



Na Tabela 15 estão apresentados os componentes das classes de comportamentos da cadeia 'Identificar equivalência das instruções escritas em linguagem natural e de programação'. A primeira classe de comportamento se refere à caracterização das linguagens de programação. Estas podem ser caracterizadas de diferentes pontos de vista: quanto ao seu grau de abstração, à estrutura de tipos, ao paradigma de programação, quanto à geração, etc. Toda linguagem de programação pertence a um ou mais paradigmas de programação. Por exemplo, no paradigma de orientação a objetos existem as linguagens de programação Java, C++, *Small-talk*; o paradigma estruturado contém as linguagens de programação C, Pascal; o paradigma de programação lógico é representado pela linguagem Prolog. Paradigmas de programação se referem às diferentes formas de estruturar o raciocínio e elaborar programas de computador. Cada paradigma difere nos conceitos básicos utilizados para programar computadores (como objetos, funções, variáveis, restrições, etc.) e são classificados em funcional, estruturado, não estruturado, sequencial, lógico, imperativo, orientado a objetos, entre outros. A sintaxe de uma linguagem de programação (segunda classe de comportamento) se refere à disposição das palavras em uma frase e a das frases no discurso. É a relação lógica das frases entre si, a estrutura gramatical da linguagem. Já a semântica

Tabela 15. Conjunto das seis classes de comportamentos e seus componentes da cadeia 'Identificar equivalência das instruções escritas em linguagem natural e de programação'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar linguagens de programação	Linguagens de programação mais comuns Paradigmas de programação	Caracterizar linguagens de programação	Linguagens de programação caracterizadas
2	Caracterizar a sintaxe de uma linguagem de programação	Linguagens de programação caracterizadas Características da sintaxe de uma linguagem de programação	Caracterizar a sintaxe de uma linguagem de programação	Sintaxe de uma linguagem de programação caracterizada M: Aumento da probabilidade de utilizar as linguagens de programação M: Aumento da probabilidade de identificar a importância da formalização
3	Caracterizar a semântica de uma linguagem de programação	Sintaxe de uma linguagem de programação caracterizada Vocabulário da linguagem de programação / semântica	Caracterizar a semântica de uma linguagem de programação	Semântica de uma linguagem de programação caracterizada M: Aumento da probabilidade de utilizar as linguagens de programação M: Aumento da probabilidade de identificar a importância da formalização
4	Identificar diferentes usos de um símbolo em uma linguagem de programação	Semântica de uma linguagem de programação caracterizada Símbolo com diferentes funções em uma linguagem de programação	Identificar diferentes usos de um símbolo em uma linguagem de programação	Usos de um símbolo em uma linguagem de programação identificados M: Aumento da probabilidade de entendimento do que cada símbolo faz e de suas funções
5	Diferenciar uso de um símbolo em diferentes linguagens	Usos de um símbolo em uma linguagem de programação identificados Diferentes linguagens (natural, matemática, lógica, de programação) Símbolo que pode ter função diferente em outras linguagens	Diferenciar uso de um símbolo em diferentes linguagens	Uso de símbolos em diferentes linguagens diferenciado M: Aumento da probabilidade de identificar erros conceituais ao programar computadores
6	Identificar equivalência das instruções escritas em linguagem natural e de programação	Uso de símbolos em diferentes linguagens diferenciado Linguagens de programação Linguagem natural	Identificar equivalência das instruções escritas em linguagem natural e de programação	Equivalência entre instruções escritas em linguagem natural e escritas em uma linguagem de programação identificada M: Aumento da probabilidade de 'conversão' entre as diferentes linguagens

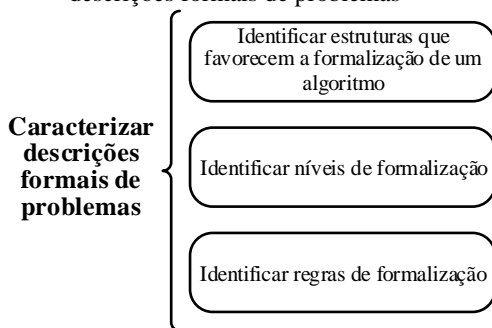
(terceira classe de comportamento) se refere ao vocabulário da linguagem, o significado de uma palavra, frase ou expressão em um determinado contexto. Essas classes de comportamentos apresentam como consequência a médio prazo o aumento da probabilidade de utilizar as linguagens de programação e de identificar a importância da formalização dos algoritmos.

Em relação a quarta classe de comportamento, os símbolos de uma linguagem de programação podem ser utilizados com diferentes funções. Por exemplo, o símbolo '+' serve para concatenar duas expressões em '3' + i; tem função de adição em $3 + i$ e serve como incremento em $i++$. Nesses exemplos, considerando que i é uma variável com valor 4, em uma determinada linguagem o resultado na concatenação poderia ser 34, na adição poderia ser 7 e no incremento seria 5. Tal classe de comportamento apresenta como consequência o aumento da probabilidade de entendimento do que cada símbolo faz e de suas possíveis funções. Um mesmo símbolo pode ter diferentes funções a depender da linguagem de programação utilizada (quinta classe de comportamento). Por exemplo, na linguagem matemática, o símbolo '=' significa igualdade e na linguagem de programação pode significar atribuição. A expressão 'A = 3' resulta em 3 na linguagem matemática. Isso significa que em qualquer equação em que 'A' aparece, pode ser substituído por 3 que o resultado não é alterado. Na linguagem de programação Java, 'A = 3' significa que uma variável nomeada como 'A' contém o valor 3 apenas enquanto outro número não for atribuído ao 'A'. Como consequência a médio prazo dessa classe de comportamento há um aumento da probabilidade de identificar erros conceituais ao programar computadores.

A última classe de comportamento (Identificar equivalência das instruções escritas em linguagem natural e de programação) tem como classe de estímulos antecedentes 'Uso de símbolos em diferentes linguagens diferenciado', 'Linguagens de programação' e 'Linguagem natural'. As instruções escritas em linguagem natural podem ter uma forma mais próxima de frases com sujeito, verbo e complemento. Por exemplo: *Some 10 vezes o valor de A; Armazene o resultado na variável 'soma de A'*. Já a mesma instrução em linguagem de programação pode apresentar características mais próximas de sentenças lógicas e matemáticas: *for (int i = 0; i < 10, i++) { somaA = a++; }*. Essa classe de comportamento têm como classe de estímulo consequente a médio prazo o aumento da probabilidade de 'conversão' entre essas linguagens.

A sequência que contém as classes de comportamentos relacionados ao processo de formalização de um algoritmo está apresentada na Figura 24.

Figura 24 Sequência de classes de comportamentos da categoria 'Caracterizar descrições formais de problemas'



A identificação de estruturas que favorecem a formalização de um algoritmo, dos níveis de formalização e das regras de formalização são as classe de comportamentos que compõem a sequência 'Caracterizar descrições formais de problemas' da Figura 24.

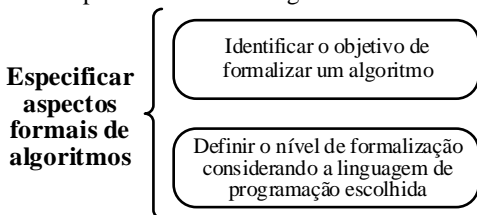
Na Tabela 16, a primeira classe de comportamento, ao se referir a estruturas 'rígidas', que favorecem a formalização de uma linguagem, há por exemplo os blocos de repetição, por facilitar a concisão de um código, a divisão do problema em subproblemas, que aumenta a probabilidade de identificar e eliminar ambiguidades e refinar o algoritmo. Os níveis de formalização das linguagens (segunda classe de comportamento) vão desde o nível considerado menos formalizado (linguagem natural), passando pelo pseudocódigo ou algoritmo, até o nível das linguagens de programação. Há algumas regras de formalização que ilustram a terceira classe de comportamento: regras de compacidade (um programa pequeno tende a ser mais bem compreendido que um grande); uniformidade; ausência de ambiguidade; concisão; precisão de uma linguagem de programação. Há um aumento da probabilidade de escrever códigos legíveis, organizados e corretos ao identificar as regras de formalização. A última classe de comportamento da Tabela 16 implica em considerar aspectos como a precisão, concisão e ausência de ambiguidade para caracterizar descrições formais de problemas.

Tabela 16. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Caracterizar descrições formais de problemas'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Identificar estruturas que favorecem a formalização de uma linguagem	Estruturas 'rígidas' de formalização de algoritmos	Identificar estruturas que favorecem a formalização de uma linguagem	Estruturas para escrever formalmente identificadas M: Aumento da probabilidade de eliminar ambiguidades e refinar o algoritmo
2	Identificar níveis de formalização	Estruturas para escrever formalmente identificadas Diferentes níveis de formalização	Identificar níveis de formalização	Níveis de formalização identificados
3	Identificar regras de formalização	Níveis de formalização identificados Regras de formalização	Identificar regras de formalização	Regras de formalização identificadas M: Aumento da probabilidade de escrever códigos legíveis, organizados e corretos
4	Caracterizar descrições formais de problemas	Regras de formalização identificadas Problemas descritos formalmente	Caracterizar descrições formais de problemas	Descrições formais de problemas caracterizadas

Na sequência da Figura 25 ocorre o processo de 'tradução', que é uma transformação das instruções escritas em formato de um algoritmo para instruções próximas da linguagem de programação escolhida. Para especificar aspectos formais de algoritmos é necessário que o objetivo de formalizar algoritmos esteja identificado e o nível de formalização esteja definido.

Figura 25 Sequência de classes de comportamentos da categoria 'Especificar aspectos formais de algoritmos'



Na Tabela 17 estão apresentadas as classes de comportamentos dessa cadeia.

Tabela 17. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Formalizar algoritmos'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Identificar o objetivo de formalizar um algoritmo	Características de algoritmos que justificam sua formalização	Identificar o objetivo de formalizar um algoritmo	Objetivo de formalizar um algoritmo identificado
2	Definir o nível de formalização considerando a linguagem de programação escolhida	Objetivo de formalizar um algoritmo identificado Linguagem de programação escolhida Diferentes níveis de formalização	Definir o nível de formalização considerando a linguagem de programação escolhida	Nível de formalização para escrever a solução do problema definido
3	Especificar aspectos formais de algoritmos	Nível de formalização para escrever a solução do problema definido Equivalência entre instruções escritas em linguagem natural e escritas em uma linguagem de programação identificada Regras de formalização identificadas	Especificar aspectos formais de algoritmos	Aspectos formais de algoritmos especificados Instruções aproximadas de instruções escritas em uma linguagem de programação M: Aumento da probabilidade de desenvolver programas utilizando apenas a linguagem de programação, sem ser necessário transformar algoritmos em uma linguagem de programação M: Aumento da probabilidade de eliminar passagens vagas ou ambíguas

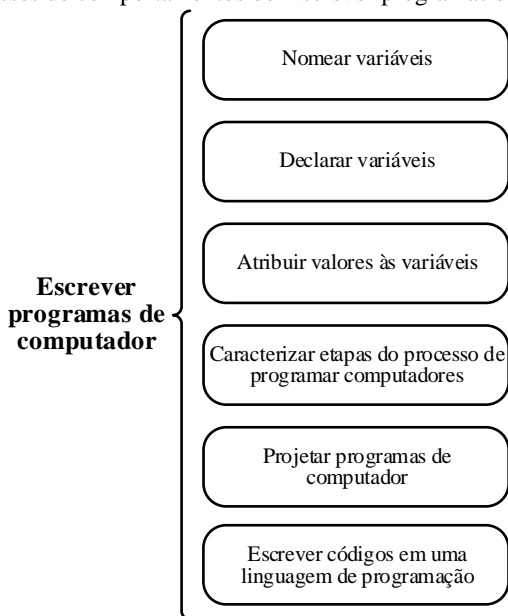
O objetivo de formalizar um algoritmo é aumentar a precisão e concisão para aproximar-se de um programa de computador. Após ter claro qual o objetivo de formalizar algoritmos é necessário definir o nível de formalização, que deve ser mais próximo possível do nível de uma linguagem de programação. Quando o nível de formalização estiver definido, ocorre a transformação das instruções escritas em algoritmos para instruções escritas em uma linguagem de programação por meio da especificação dos aspectos formais do algoritmo. Para isso é necessário que seja identificada a equivalência entre instruções escritas em linguagem natural e escritas em uma linguagem de programação (classe de comportamento que ocorre na cadeia 'Identificar equivalência de instruções em linguagem natural e de programação'), que as regras de formalização (precisão, rigidez, concisão, interpretação única, blocos, repetições, divisão em casos) estejam explicitadas (classe de comportamento da cadeia 'Caracterizar descrições formais de problemas') e que o nível de formalização para escrever a solução do problema esteja definido. Ao realizar essa transformação há um aumento da probabilidade de desenvolver programas utilizando apenas a linguagem de programação, sem ser necessário transformar algoritmos em uma linguagem de programação e um aumento da probabilidade de eliminar passagens vagas ou ambíguas.

6. CATEGORIA DE COMPORTAMENTOS 'ESCREVER PROGRAMAS DE COMPUTADOR'

Escrever programas de computador é a categoria de comportamentos que representa o processo de programar computadores mais explicitamente. Para escrever programas de computador é necessário que todas as classes de comportamentos das categorias anteriores tenham sido aprendidos.

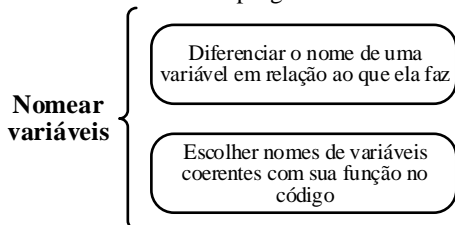
Na Figura 26 é possível observar as categorias que compõem 'Escrever programas de computador'. Essas categorias contém classes de comportamentos relacionados às variáveis (Nomear variáveis, Declarar variáveis e Atribuir valores às variáveis), Caracterizar etapas do processo de programar computadores, Projetar programas de computador e 'Escrever códigos em uma linguagem de programação'.

Figura 26 Classes de comportamentos de 'Escrever programas de computador'



As próximas três sequências e cadeias se referem a aspectos específicos relacionados a variáveis. Entende-se variável, para a programação de computadores, como uma posição de memória que armazena um determinado valor ou expressão (Boratti, 2004). Tais posições de memória existem somente em tempo de execução, sendo que enquanto se está desenvolvendo o programa, a variável é identificada por um nome e um tipo, mas ainda não é alocado um espaço na memória para sua criação.

Figura 27 Sequência de classes de comportamentos da categoria 'Nomear variáveis de um programa'



Um dos requisitos para declarar variáveis é nomeá-la. Essa classe de comportamento se constitui em uma sequência, apresentada na Figura

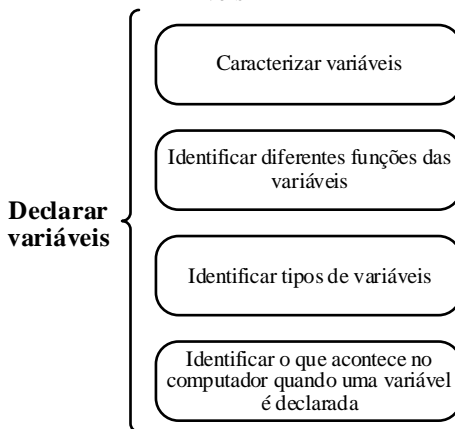
27. Nomear variáveis de um programa de computador é um processo comportamental que exige a diferenciação do nome de uma variável do que é possível fazer com ela. A partir dessa diferenciação, é feita a escolha do nome de forma a ser coerente com o que ela faz no código.

Tabela 18. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Nomear variáveis de um programa'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Diferenciar o nome de uma variável em relação ao que ela faz	Diferentes funções das variáveis identificadas Nomes de variáveis com diferentes significados possíveis	Diferenciar o nome de uma variável em relação ao que ela faz	Nome de uma variável diferenciado em relação ao que ela faz M: Diminuição da probabilidade de considerar que o computador 'compreende' o que a variável faz a partir do significado da palavra M: Aumento da probabilidade de identificar inconsistências que o sujeito apresenta a respeito do funcionamento do computador
2	Escolher nomes de variáveis coerentes com sua função no código	Nome de uma variável diferenciado em relação ao que ela faz Necessidade de criar uma variável	Escolher nomes de variáveis coerentes com sua função no código	Nomes de variáveis escolhidos de forma coerente com sua função no código
3	Nomear variáveis	Nomes de variáveis escolhidos de forma coerente com sua função no código Regras de uniformidade Linguagens de programação que não permitem uso de palavras-chave ou reservadas para declarar variáveis	Nomear variáveis respeitando as regras de uniformidade	Variáveis nomeadas de acordo com as regras de uniformidade M: Leitura do programa facilitada M: Diminuição da probabilidade de equívocos na leitura dos programas

A primeira classe de comportamento se refere à diferenciação entre o que é o nome de variável e o que a variável faz, pois é comum em alunos a tentativa de atribuir um nome significativo para as variáveis esperando que o computador 'entenda' o nome e 'perceba' as intenções do aluno (Putnam e outros, 1986, citado em Dehnadi, 2009). No entanto, é relevante escolher nomes de variáveis que sejam coerentes com sua função no código (segunda classe de comportamento), ou seja, nomes significativos. A diferença está em considerar que apenas a escolha do nome fará com que o computador faça algo coerente com o nome dado. Outra classe de comportamento importante é nomear variáveis considerando as regras de uniformidade da linguagem de programação escolhida. Por exemplo, na maioria das linguagens não é possível nomear uma variável com uma palavra reservada (por exemplo, em Java, as palavras-chave *if*, *main*, *while*, *public*, entre outras, são utilizadas para outras funções naquela linguagem). Logo para nomear as variáveis é necessário respeitar as regras de uniformidade, por exemplo, utilizando palavras diferentes das palavras-chave.

Figura 28 Sequência de classes de comportamentos da categoria 'Declarar variáveis'



A Figura 28 contém a sequência de classes de comportamentos denominada 'Declarar variáveis'. O termo 'declarar' se refere à criação de uma variável em um código de programação. Para declarar variáveis é necessário caracterizá-las, identificar suas diferentes funções e tipos e identificar o que acontece no computador quando é feita uma declaração de variáveis.

Tabela 19. Conjunto das cinco classes de comportamentos e seus componentes da cadeia 'Declarar variáveis'

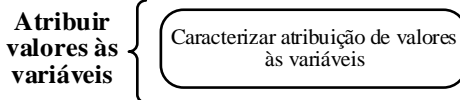
	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar variáveis	Características de variáveis Memória do computador	Definir características das variáveis	Variáveis caracterizadas
2	Identificar diferentes funções das variáveis	Variáveis caracterizadas Funções das variáveis	Identificar diferentes funções das variáveis	Diferentes funções das variáveis identificadas
3	Identificar tipos de variáveis	Diferentes funções das variáveis identificadas Tipos de variáveis de programação	Identificar tipos de variáveis	Tipos de variáveis identificados
4	Identificar o que acontece no computador quando uma variável é declarada	Tipos de variáveis identificados Memória de um computador	Identificar o que acontece no computador quando uma variável é declarada	Eventos que acontecem no computador quando uma variável é declarada identificados
5	Declarar variáveis	Eventos que acontecem no computador quando uma variável é declarada identificados Necessidade de criar uma variável para resolver um problema	Definir o tipo e o nome da variável	Variável declarada e 'alocada' em uma posição de memória

Na Tabela 19 o conjunto das classes de comportamentos e seus componentes estão apresentados. A primeira classe de comportamento se refere à caracterização do que é variável. Variável é uma forma de representar características de um fenômeno que tem valores que variam. Por exemplo, a característica 'cor de um objeto' pode ter os valores 'vermelho', 'verde', 'rosa', 'azul', etc. ou os valores do intervalo de frequência de onda do raio de luz. Nesse exemplo, a característica 'cor de um objeto' é denominada variável, pois tem valores que mudam em uma graduação. Em um

computador, esses valores das variáveis necessitam ser guardados em algum local. O conceito de variável tem relação com a memória de um computador, pois a criação de uma variável depende de haver um espaço de memória reservado no computador para sua criação e armazenamento dos valores. Variável, na programação de computadores, é um espaço da memória que tem como objetivo armazenar dados em um período de tempo. Esse espaço ainda não está 'preenchido' por um valor, é apenas um local reservado para referenciar a variável declarada (ou criada). A segunda classe de comportamento se refere às diferentes funções que uma variável pode adquirir em um programa de computador. Alguns exemplos são a função de contador de repetição, seleção, soma, inicialização, atribuição, acumulação, variáveis de entrada e variáveis de saída, etc.

A terceira classe de comportamento se refere à identificação dos tipos de variáveis de programação. Estas podem ser do tipo 'inteiro', 'real', 'caractere' e 'cadeia de caracteres'. Há outros tipos de variáveis em diferentes linguagens de programação, porém, esses são os tipos mais básicos. Em relação a quarta classe de comportamento, quando uma variável é declarada, um espaço de memória é reservado no computador para armazená-la. Declarar ou criar uma variável (última classe de comportamento) implica em definir seu tipo e seu nome. Essa classe de comportamento tem como classe de resposta possível 'definir tipo e nome da variável', e como consequência imediata, a variável é alocada em uma 'posição' de memória no computador.

Figura 29 Classe de comportamento da categoria 'Atribuir valores às variáveis'



Após a declaração da variável e sua nomeação, é necessário atribuir valores às variáveis. Esta categoria é representada pela classe de comportamento da Figura 29, denominada 'Caracterizar atribuição de valores às variáveis'.

Na Tabela 20 estão apresentados os componentes das classes de comportamentos dessa cadeia. A atribuição de valores nas variáveis se refere ao armazenamento de um valor no espaço de memória reservado após a variável ter sido declarada. No computador, o valor é armazenado na memória e permanece até ser explicitamente alterado, seu conteúdo apagado ou a máquina desligada. Uma posição de memória do computador é preenchida com o valor, porém, esta posição de memória pode ter

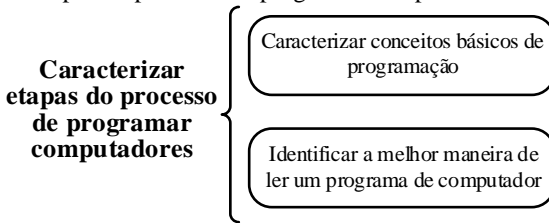
seu valor alterado. A caracterização do processo de atribuição de valores nas variáveis produz como consequência um aumento da probabilidade de entender que variáveis armazenam valores que mudam. Ao atribuir valores as variáveis é necessário que estes sejam compatíveis com o tipo de variável que foi declarada (se a variável 'A' é do tipo inteiro, por exemplo, não deve atribuir a esta variável o caractere 'c' ou '0.97'). Há uma

Tabela 20 Conjunto das duas classes de comportamentos e seus componentes da cadeia 'Atribuir valores às variáveis'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar atribuição de valores nas variáveis	Descrição de como ocorre o processo de atribuição de valores a uma variável Posições de memória de um computador Valor atribuído a uma variável que pode mudar a depender de alguma instrução específica	Identificar o que acontece no computador quando valores são atribuídos a uma variável	Atribuição de valores nas variáveis caracterizado M: Aumento da probabilidade de entender que variáveis armazenam valores que mudam
2	Atribuir valores às variáveis	Atribuição de valores nas variáveis caracterizado Variáveis de diferentes tipos Valores de diferentes tipos	Atribuir valores compatíveis com o tipo da variável declarada	Valores compatíveis atribuídos a uma variável declarada M: Diminuição da probabilidade de equívocos a respeito de variáveis

diminuição da probabilidade de equívocos a respeito de variáveis, como considerar que uma variável pode conter mais de um valor por vez. A atribuição de valores às variáveis pode apresentar problemas caso não haja clareza de que os valores não são fixos ou imutáveis, pois os valores mudam quando houver no código algo que produza essa alteração.

Figura 30 Sequência de classes de comportamentos da categoria 'Caracterizar etapas do processo de programar computadores'



A Figura 30 contém os comportamentos da sequência 'Caracterizar etapas do processo de programar computadores'. Esta sequência é formada pelas classes de comportamentos 'Caracterizar conceitos básicos de programação' e 'Identificar a melhor maneira de ler um programa de computador'.

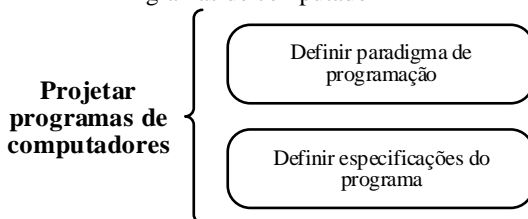
Tabela 21. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Caracterizar etapas do processo de programar computadores'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar conceitos básicos de programação	Conceitos básicos de programação	Caracterizar conceitos básicos de programação	Conceitos básicos de programação de computadores caracterizados
2	Identificar a melhor maneira de ler um programa de computador	Entendimento do que as linhas de código fazem juntas Conceitos básicos de programação de computadores caracterizados Diferentes maneiras de ler um programa de computador	Identificar a melhor maneira de ler um programa de computador	Melhor maneira de ler um programa de computador identificada
3	Caracterizar etapas do processo de programar computadores	Melhor maneira de ler um programa de computador identificada Etapas do processo de programar computadores	Caracterizar etapas do processo de programar computadores	Etapas do processo de programar computadores caracterizadas

				M: Aumento da probabilidade de entender como o programa resolve o problema
--	--	--	--	--

Em relação aos conceitos básicos de programação, estes se referem aos conceitos de variáveis, método, instruções, funções, etc.; Para ocorrer a segunda classe de comportamento é necessário que haja um entendimento do que as linhas de código fazem juntas (estímulo referente à cadeia 'Construir algoritmos') e que se conheçam diferentes maneiras de ler um programa de computador (por exemplo, ler do início ao final, ler por blocos de instruções, etc.). Para a ocorrência da terceira classe de comportamento é necessário a identificação da melhor maneira de ler um programa de computador e o conhecimento das etapas do processo de programar computadores (por exemplo, as etapas de escrita, de compilação, de execução de um código). Esse classe de comportamento tem como consequência a médio prazo o aumento da probabilidade de entender como o programa resolve o problema.

Figura 31 Sequência de classes de comportamentos da categoria 'Projetar programas de computador'



A Figura 31 representa a sequência de classes de comportamento referente ao projeto de programas de computador. As classes de comportamento dessa categoria se referem à definição do paradigma de programação e das especificações do programa. Os componentes de tais classes de comportamentos estão apresentados na Tabela 22.

O projeto de um programa de computador se inicia com a definição do paradigma de programação. Após a ocorrência dessa classe de comportamento, há a definição das especificações do programa. Especificações são requisitos a cumprir, uma definição do que se espera que um programa de computador faça e de seu âmbito de aplicação (os limites do

programa, o que ele vai resolver, onde começa e onde termina). Essa definição aumenta a probabilidade de que os programas elaborados tenham objetivos bem definidos e que sejam formados por códigos relevantes. As classes de estímulos consequentes a curto prazo das duas primeiras classes de comportamentos são classes de estímulos antecedentes para a última classe de comportamento.

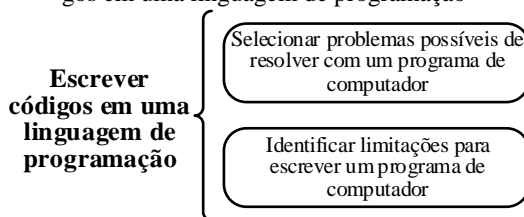
Tabela 22. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Projetar programas de computador'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Definir paradigma de programação	Diferentes paradigmas de programação	Definir paradigma de programação	Paradigma de programação definido
2	Definir especificações do programa	Paradigma de programação definido Especificações para elaborar um programa de computador	Definir especificações do programa	Especificações do programa definidas M: Aumento da probabilidade de elaborar programas sem códigos soltos e sem objetivo
3	Projetar programas de computador	Estrutura básica de um programa de computador Regras de construção de um programa Paradigma de programação escolhido Especificações do programa definidas	Projetar programas de computador	Estrutura que descreve em linhas gerais o projeto do programa elaborada M: Diminuição da probabilidade de ter um programa com partes não planejadas

Ao projetar o programa de computador, considerando o paradigma de programação e as especificações do programa, há diminuição da probabilidade de ter um programa com partes não planejadas e uma estrutura que descreve o projeto do programa é uma consequência imediata dessa classe de comportamento.

Após o projeto do programa de computador elaborado, na Figura 32 são apresentadas as classes de comportamentos da sequência 'Escrever códigos em uma linguagem de programação'.

Figura 32 Sequência de classes de comportamentos da categoria 'Escrever códigos em uma linguagem de programação'



Conforme apresentado na Figura 32, a primeira classe de comportamento necessária para escrever códigos em uma linguagem de programação é 'Selecionar problemas possíveis de resolver com um programa de computador', seguida da classe de comportamento 'Identificar limitações para escrever um programa de computador'.

Tabela 23. Conjunto das três comportamentos e seus respectivos componentes da cadeia 'Escrever programas de computador'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Selecionar problemas possíveis de resolver com um programa de computador	Diferentes tipos de problemas a resolver	Selecionar problemas possíveis de resolver com um programa de computador	Problemas possíveis de resolver com um programa de computador selecionados
2	Identificar limitações para escrever um programa de computador	Problemas possíveis de resolver com um programa de computador selecionados Limitações para escrever um programa de computador	Identificar limitações para escrever um programa de computador	Limitações para escrever programas de computador identificadas
3	Escrever códigos em uma linguagem de programação	Limitações para escrever programas de computador identificadas Regras de formalização identificadas Etapas do processo de programar computadores identificadas	Escrever códigos em uma linguagem de programação	Programa de computador escrito

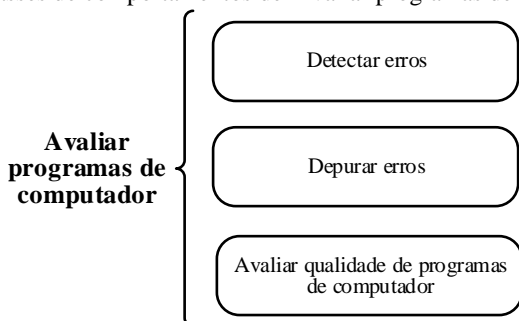
Há problemas que não são computáveis ou programáveis, e precisam ser transformados em problemas programáveis ou descartados como um problema computacional. Após a seleção de problemas possíveis de se resolver por meio de um programa de computador, é necessário identificar as limitações para escrever um programa de computador, por exemplo, limitações de máquina, de linguagem, do programador, das especificações do usuário. Com essas classes de comportamentos apresentados, é possível a ocorrência da classe de comportamento 'Escrever códigos em uma linguagem de programação'. Essa classe de comportamento apresenta como classe de estímulos que compõem a situação antecedente 'Limitações para escrever programas de computador identificadas', 'Regras de formalização identificadas' (pertencente à cadeia 'Caracterizar descrições formais de problemas') e 'Etapas do processo de programar computadores identificadas', da cadeia 'Caracterizar etapas do processo de programar computadores'.

Após a escrita de um código em uma linguagem de programação, é necessário avaliar se o programa de computador produzido contém erros e é necessário corrigir tais erros, classes de comportamentos a serem apresentados na última categoria, denominada 'Avaliar programas de computador'.

7. CATEGORIA DE COMPORTAMENTOS 'AVALIAR PROGRAMAS DE COMPUTADOR'

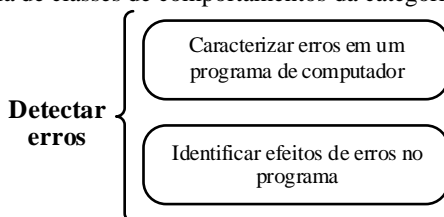
A categoria de comportamentos denominada 'Avaliar programas de computador' ocorre após a escrita de um programa de computador, e trata de classes de comportamentos relacionadas a detecção e depuração de erros.

Figura 33 Classes de comportamentos de 'Avaliar programas de computador'



As classes de comportamentos que compõem a categoria 'Avaliar programas de computador' são 'Detectar erros', 'Depurar erros' e 'Avaliar qualidade de programas de computador', conforme representado na Figura 33.

Figura 34 Sequência de classes de comportamentos da categoria 'Detectar erros'



A Figura 34 contém as classes de comportamentos da sequência 'Detectar erros', que se refere à identificação de erros que resultam da execução do programa de computador, geralmente detectados por quem utiliza o programa. Para detectar erros em um programa de computador (terceira classe de comportamento) é necessário 'Caracterizar erros em um programa de computador' e 'Identificar efeitos de erros no programa'.

A primeira classe de comportamento se refere à caracterização dos erros em um programa de computador que surgem durante a elaboração ou execução do programa. Erros podem ser sintáticos ou semânticos. Os erros sintáticos resultam de problemas na escrita do código, por exemplo, usar palavras reservadas da linguagem de programação para nomear variáveis, não fechar parênteses, aspas, chaves, etc. Erros semânticos podem

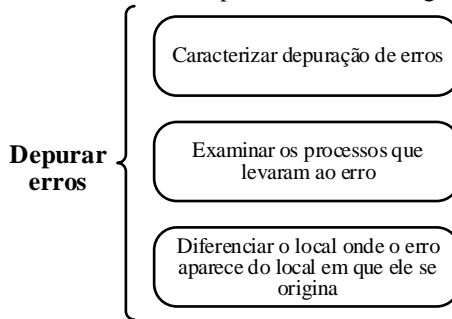
Tabela 24. Conjunto das três classes de comportamentos e seus componentes da cadeia 'Detectar erros'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar erros em um programa de computador	Problemas durante a elaboração de um programa Definição de erro Erro em um programa ao ser executado	Caracterizar erros em um programa de computador	Erros em um programa de computador caracterizados
2	Identificar efeitos de erros em um programa	Erros em um programa de computador caracterizados Diferentes efeitos no programa que cada erro pode ocasionar	Identificar efeitos de erros em um programa	Efeitos de erros em um código identificados M: Aumento da probabilidade de aprender de que forma erros se manifestam no código de um programa
3	Detectar erros em um programa de computador	Efeitos de erros em um código identificados Erro em um programa ao ser executado	Testar o programa elaborado para detectar erros	Erros em um programa detectados M: Aumento da probabilidade de 'tratar' erros no código M: Aumento da visibilidade dos erros do programa

ser considerados como 'erros de raciocínio'. Alguns tipos de erros semânticos podem ser detectados automaticamente, mas outros só podem ser detectados durante a execução, por meio de testes. Nesse caso, esses erros são nomeados como erros de execução, porém, são erros semânticos igualmente. Um exemplo de erro semântico é o loop infinito, quando o programa não para de ser executado. A segunda classe de comportamento é 'Identificar efeitos de erros em um programa' e considera os diferentes efeitos no programa que cada erro pode ocasionar (por exemplo, uma parada abrupta no programa em execução, resultados inconsistentes, o desligamento do computador, etc.). A consequência de mais interesse nessa classe de comportamento é o aumento da probabilidade de aprender de que forma erros se manifestam no código de um programa. A terceira classe de comportamento é denominado 'Detectar erros em um programa de computador'. Essa classe de comportamento apresenta como classe de resposta 'Testar o programa elaborado para detectar erros'. Além de detectar os erros, há um aumento da visibilidade dos erros do programa e da

probabilidade de 'tratar' erros no código, sendo que esse tratamento pode ser mais bem explicitado na cadeia a seguir (Depurar erros).

Figura 35 Sequência de classes de comportamentos da categoria 'Depurar erros'



Na Figura 35 estão contidas as classes de comportamentos da sequência 'Depurar erros', sendo a primeira denominada 'Caracterizar depuração de erros'. Após essa caracterização é feito um exame dos processos elaborados que levaram ao erro (segunda classe de comportamento) e uma diferenciação entre a origem do erro e o local onde o erro aparece (terceira classe de comportamento).

Depurar erros é uma palavra utilizada na área de programação de computadores para se referir ao processo de correção de um erro de execução ou de elaboração em um programa de computador. A classe de comportamento referente ao exame dos processos que levaram a ocorrência do erro (segunda classe de comportamento) tem como classe de estímulo antecedente o erro detectado, referente à cadeia anterior, e diversas consequências relevantes para o programador: ao identificar o que foi escrito ou feito de errado, a busca por um culpado pelo erro (geralmente se considera que o 'culpado' é o próprio computador, que não está 'funcionando') é eliminada e as causas do erro emergem. Outra consequência importante é a diminuição das reclamações a respeito da ocorrência do erro, comportamento que não ajuda a resolver o erro detectado e aumenta a indisposição do programador para escrever um programa. Com isso há um aumento da probabilidade de identificar comportamentos inadequados ao programar o computador e do programador aperfeiçoar seus comportamentos. A terceira classe de comportamento se refere à diferenciação entre o local onde o erro aparece e o local em que ele se origina. Essa diferenciação considera que a origem ou causa do erro não está necessariamente onde o erro surge. A quarta classe de comportamento é 'Depurar erros'. Para a ocor-

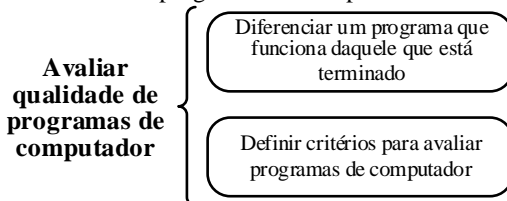
rência dessa classe de comportamento os erros em um programa (referente à cadeia 'Detectar erros') fazem parte da situação antecedente. Uma resposta possível para tal classe de comportamento é 'Corrigir erros encontrados rastreando o código ou utilizando outras técnicas'. Essa classe de comportamento tem como consequência a possibilidade de aperfeiçoamentos no código serem feitos.

Tabela 25. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Depurar erros'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Caracterizar depuração de erros	Definição de depuração Características do processo de depurar	Caracterizar depuração de erros	Depuração de erros caracterizada
2	Examinar os processos que levaram ao erro	Depuração de erros caracterizada Erro em um programa detectado	Identificar o que foi escrito ou feito de errado	Processos que levaram ao erro examinados M: Diminuição da probabilidade de procurar um culpado pelo erro e esconder as causas do problema / Aumento da probabilidade de resolver o problema ao invés de reclamar de sua ocorrência L: Aumento da probabilidade de identificar comportamentos inadequados ao programar o computador L: Aumento da probabilidade do programador aperfeiçoar seus comportamentos
3	Diferenciar o local onde o erro aparece do local em que ele se origina	Processos que levaram ao erro examinados Local do código onde o erro aparece Origem ou causa do erro	Diferenciar o local onde o erro aparece do local em que ele se origina	Diferenciação entre local onde o erro aparece e local que causa o erro (origem) M: Aumento da probabilidade de encontrar e corrigir erros no programa
4	Depurar erros	Diferenciação entre local onde o erro aparece e local que causa o erro (origem) Erros em um programa detectados	Corrigir erros encontrados rastreando o código ou utilizando outras técnicas	Programa de computador depurado Erros corrigidos Aperfeiçoamentos realizados

Na Figura 36 as classes de comportamentos da sequência 'Avaliar qualidade de programas de computador' são apresentadas. Essa sequência se inicia com 'Diferenciar um programa que funciona daquele que está terminado' e depois 'Definir critérios para avaliar programas de computador'. Essa categoria é a última da classe geral 'programar computadores'.

Figura 36 Sequência de classes de comportamentos da categoria 'Avaliar qualidade de programas de computador'



A primeira classe de comportamento se refere às definições de quando o programa está funcionando a depender das especificações de um programa e de seus usuários, pois a partir do que os usuários apresentam como problema a resolver e das especificações existentes o programa funciona ou não. Diferenciar um programa que funciona daquele que está terminado implica em considerar que um programa que foi executado está 'terminado' ou 'aparentemente funcionando' e um programa que fornece as saídas corretas para cada entrada é um programa que funciona de fato. Há um aumento da probabilidade de elaborar programas que funcionem corretamente ao fazer tal diferenciação. Definir critérios para avaliar programas de computador é a segunda classe de comportamento da cadeia e tem como classe de estímulo antecedente 'Critérios de qualidade para avaliar programas de computador'. Entre os critérios de qualidade possíveis pode-se indicar a eficiência, eficácia, custo disponível para elaborar o programa, atendimento às especificações, valor para as pessoas, etc. Com a definição dos critérios de avaliação de programas de computador ocorre um aumento da probabilidade de avaliar um programa de computador de forma adequada e de selecionar critérios relevantes para avaliar programas de programas de computador, é necessário levar em conta os critérios de avaliação de programas de computador para considerar se o problema é ou não resolvido por meio do programa de computador. Com isso, obtém-se mais perspectiva a respeito do que foi feito e aumenta a aprendizagem a respeito dos determinantes do sucesso da elaboração de um programa de computador. A avaliação do programa de computador proporciona ao sujeito a identificação das dificuldades que teve ao elaborar um

programa de computador e o questionamento a respeito do que determinou o sucesso do programa.

Tabela 26. Conjunto das quatro classes de comportamentos e seus componentes da cadeia 'Avaliar qualidade de programas de computador'

	Classe de comportamento	Classe de estímulos da situação antecedente	Classe de resposta	Classe de estímulos da situação consequente
1	Diferenciar um programa que funciona daquele que está terminado	Programa que foi possível executar Programa que fornece as saídas corretas para cada entrada Definições de quando o programa está funcionando a depender das especificações e dos usuários	Diferenciar um programa que funciona daquele que está terminado	Programa que funciona diferenciado daquele que está terminado M: Aumento da probabilidade de elaborar programas que funcionem corretamente
2	Definir critérios para avaliar programas de computador	Programa que funciona diferenciado daquele que está terminado Critérios de qualidade para avaliar programas de computador	Definir critérios para avaliar programas de computador	Critérios de avaliação de programas de computador definidos M: Aumento da probabilidade de avaliar um programa de computador de forma adequada M: Aumento da probabilidade de selecionar critérios relevantes para avaliar programas de computador
3	Avaliar qualidade de programas de computador	Critérios de avaliação de programas de computador definidos	Avaliar qualidade de programas de computador	Avaliação da qualidade do programa em termos da resolução do problema M: Obtenção de mais perspectiva a respeito do que foi feito M: Aprendizagem a respeito dos determinantes do sucesso da elaboração do programa

IV DISCUSSÃO

Este trabalho possibilitou identificar características específicas da classe geral de comportamentos 'programar computadores', apresentar esse fenômeno como um processo comportamental e organizar as classes de comportamentos que a compõem de forma a explicitar relações de encaqueamento entre elas. A organização das informações em função das classes de estímulos antecedentes, classes de respostas e classes de estímulos consequentes torna mais provável o planejamento de condições de ensino e o desenvolvimento de pessoas para programar computadores.

A maioria dos artigos que tratam de problemas no ensino e na aprendizagem de programação de computadores propõem soluções para tais problemas mesmo sem uma caracterização das classes de comportamentos que compõem a classe geral 'programar computadores'. Tais soluções são baseadas na elaboração de técnicas de ensino, ambientes de desenvolvimento e aprendizagem, linguagens de programação específicas, entre outros. Há muitos supostos de que os problemas se relacionam a uma dificuldade de abstração, de raciocínio, mas sem clareza do que essas dificuldades significam e sem verificar se o problema está nessas dificuldades. Por meio da caracterização do processo de programar computadores há maior facilidade na identificação das classes de comportamentos mais difíceis de ensinar, bem como torna mais claro identificar se as dificuldades de alunos estão em uma classe de comportamento ainda pouco clara em termos do que produz como consequência, quais classes de estímulos devem ser levadas em conta, se é uma classe de comportamento cujos elos anteriores são muitos básicos, não fornecendo uma aprendizagem gradual ou outras possibilidades de explicação para tais dificuldades.

A sistematização da literatura a respeito das estratégias adotadas para resolver problemas em ensino de algoritmo e programação de computadores feita por Pereira Junior (2006) evidencia que, em relação às ferramentas elaboradas, muitas não apresentam integração com estratégias pedagógicas. Dehnadi (2009) afirma que os ambientes de desenvolvimento ensinam de forma parcial e restringem a aprendizagem a um nível superficial e inadequado, não facilitam o entendimento dos mecanismos de programação que são a base das 'ações'. Essas abordagens não enfatizam, por exemplo, o ensino de conceitos essenciais de programação. Ambos os autores destacam a necessidade de considerar mais do que elaboração de ferramentas ou ambientes de desenvolvimento para resolver

as dificuldades em programar computadores. Ao explicitar as diversas classes de comportamentos que compõem a classe geral programar computadores é possível criar ferramentas e ambientes de desenvolvimento mais completos em termos do ensino de programação de computadores.

Há muitas classes de comportamentos que compõem a classe geral programar computadores que necessitam ser ensinados para garantir a aprendizagem efetiva desse complexo processo comportamental. No entanto, o ensino de programação de computadores muitas vezes se inicia diretamente com a escrita de programas de computador, sem considerar as diferenças de repertórios dos alunos, que mesmo que já saibam programar computadores, não aprenderam de maneira formal e podem ter dificuldades em apresentar algum comportamento, podem apresentar comportamentos parcialmente eficazes, ou mesmo ações que parecem se relacionar a alguma classe de comportamento necessária e no entanto, tratam-se mais de classes de comportamentos supersticiosas e irrelevantes para programar computadores. Muitas vezes essa aprendizagem não formal indica ao professor, de maneira equivocada, que o aluno sabe programar. A explicitação das cadeias de comportamentos das categorias anteriores à categoria 'escrever programas de computador' como parte do processo de programar computadores aumenta a probabilidade de que o ensino de programação de computadores seja mais eficaz, por ensinar ao aluno outros aspectos que também fazem parte desse processo, como ilustrados pelas classes de comportamentos das categorias 'Avaliar argumentos de acordo com regras lógicas', 'Caracterizar funcionamento de computadores', 'Resolver problemas' e 'Construir algoritmos'. Tais categorias facilitam o ensino de conceitos fundamentais de programação, formas de resolver problemas, elaboração de diferentes soluções em formato de algoritmos, etc. O ensino dessa classe geral ocorre muito antes da escrita de um programa de computador, viabilizando a aprendizagem de classes de comportamentos que são parte do processo mas que não são explicitados como tal.

As classes de comportamentos das categorias 'Avaliar argumentos de acordo com regras lógicas', 'Caracterizar funcionamento de computadores', 'Resolver problemas' e 'Construir algoritmos' são desconsiderados como objetivos de ensino explícito em cursos de Ciências da Computação (Delgado e outros, 2005). Com isso, aumenta a dificuldade do professor em ensinar o aluno a programar computadores assim como ocorre uma tendência a buscar outras explicações para as dificuldades, muitas vezes enfatizando 'causas internas' como uma forma de racionalização e esquiva de suas responsabilidades de ensino. Ao explicitar as cadeias de comportamentos e o que se deve obter como consequência em cada classe de

comportamento, há uma diminuição da probabilidade do professor seguir o modelo tradicional de ensino, que prioriza atividades do aluno, informações, intenções ou ações do professor como o que deve ser ensinado em vez de classes de comportamentos mais próximos das classes de comportamentos profissionais e relevantes para a sociedade. Por meio da explicitação das classes de estímulos consequentes de cada classe de comportamento, é mais provável que o aluno obtenha o feedback adequado para cada classe de comportamento aprendida, e que o professor tenha melhores condições de avaliar o desempenho do aluno bem como a adequação de seus programas de ensino.

É necessário que o professor seja capaz de detectar o que os alunos aprenderam como consequência das condições de ensino. A avaliação do desempenho do aluno e dessas condições possibilita estabelecer uma relação de dependência entre o que é ensinado e o que é aprendido. O que o aluno passa a fazer é consequência daquilo que foi planejado pelo professor. Por exemplo, ao elaborar um programa de ensino o professor percebe que o aluno desconhece as características de um algoritmo e que não é capaz de elaborar argumentos lógicos, classes de comportamentos essenciais para escrever códigos em qualquer linguagem de programação e elabora seu programa de ensino com atividades que ajudem ao aluno a caracterizar um algoritmo e elaborar argumentos lógicos. Tornando explícito diversas classes de comportamentos componentes do processo de programar computadores, os resultados possibilitam a elaboração de programas de ensino e tornam possível também a avaliação do desempenho do aluno e das etapas críticas no programa de ensino, das lacunas a preencher para tornar o ensino mais gradual, do ritmo de aprendizagem do aluno e da necessidade de fornecer consequências relevantes para que as classes de comportamentos ocorram e se mantenham.

De acordo com Weinberg (1998) a programação de computadores é uma forma de comunicação entre homens e máquinas. O autor afirma que é preciso reconhecer o foco desse fenômeno não como simples manipulação de símbolos, mas como comportamento humano. A explicitação de cadeias de comportamentos da classe geral 'programar computadores' apresenta em minúcia sua complexidade, que muitas vezes é compreendida em outros termos que não um processo comportamental, conduzido por pessoas. Ao estudar do ponto de vista de comportamento humano, a programação de computadores passa a ser vista como algo possível de ser ensinado e aprendido, com lacunas ainda a expor, mas não como algo misterioso ou impossível de desvendar. O programador controla seu próprio comportamento em relação à máquina ao programar um computador. A observação de seu comportamento produz mais clareza do que foi feito

para o computador apresentar um determinado resultado, dos erros cometidos, dos aperfeiçoamentos que devem ser feitos, das possibilidades de escrever um código, além de mais facilidade em projetar o programa, em estruturar algoritmos, detectar os erros e avaliar os determinantes do bom funcionamento do programa.

Programar computadores não se restringe apenas a 'escrever um programa de computador', pois este representa apenas a parte final e observável do processo, quando a classe geral de comportamento já está consolidado. Nesse sentido, 'programar não é apenas o comportamento humano, é um comportamento humano complexo' (Weinberg, 1998). Ao considerar as categorias de avaliar argumentos de acordo com regras lógicas, caracterizar computadores, resolver problemas, construir e formalizar algoritmos e avaliar programas de computador como parte da classe geral de comportamento 'programar computadores' e também como parte de um programa de ensino dessa classe geral, evidencia-se a complexidade desse fenômeno comportamental. Essa pesquisa contribui ao ampliar o conhecimento a respeito da classe geral 'programar computadores' ao considerar tais categorias como parte essencial desse processo.

Em relação às dificuldades identificadas nessa pesquisa, é possível afirmar que o processo de programar computadores pode não ocorrer de forma linear como as cadeias de comportamentos apresentadas podem sugerir. As cadeias de comportamentos e as relações entre as cadeias explicitam as classes de comportamentos que compõem a classe geral, mas nem sempre explicitam como estas classes de comportamentos ocorrem, qual a ordem ou sequência de apresentação delas. Por isso é necessário diferenciar o que é um programa de ensino de programar computadores do que é o processo de programar computadores ocorrendo. Não é possível visualizar, por exemplo, durante a ocorrência do processo de programar computadores, o aluno elaborando argumentos (cadeia 'Avaliar argumentos de acordo com regras lógicas'), identificando conceitos básicos do computador (cadeia 'Caracterizar funcionamento de computadores') ou caracterizando dados de um problema (Resolver problemas), etc.

A sequência das classes de comportamentos se refere ao processo de aprendizagem e de elaboração de condições de ensino e não ao processo de programar computadores em si, em que algumas classes de comportamentos das categorias provavelmente não ocorrem, ocorrem parcialmente ou concomitantemente ao processo de programar computadores. Por exemplo, as classes de comportamentos de caracterização, em geral, não ocorrem durante a escrita de um programa de computador. São classes de comportamentos pré-requisitos que ocorrem de forma encoberta. Isso não significa que o processo de programar computadores se restringe

apenas à escrita de um programa de computador, mas sim que há muitos comportamentos encobertos, e que a explicitação dessas classes de comportamentos evidencia uma provável sequência de ensino para programar computadores.

Em relação às sequências de comportamentos, o que deve ser ensinado antes ou depois não está consolidado na literatura. As classes de comportamentos das categorias foram organizados em sequências, porém nem sempre há uma relação sequencial bem explícita. A organização feita foi baseada no critério de graus crescentes de complexidade das classes de comportamentos e de relações de pré-requisitos. Há também a possibilidade de, em algumas situações, a ordem das sequências de algumas classes de comportamentos não ser evidente, ou pode ser possível que tais classes de comportamentos ocorram de forma concomitante, dificultando o estabelecimento de uma sequência determinada. É necessário portanto a verificação experimental de quais as sequências adequadas para o ensino de tais classes de comportamentos, que facilitam a aprendizagem para o aluno das classes de comportamentos de cada categoria apresentada.

Em relação às classes de respostas das classes de comportamentos, na maioria das cadeias estas apresentam o mesmo nome da classe de comportamento, pois não foi possível explicitar todas as variações que podem produzir os resultados indicados em cada classe de comportamento. A nomeação de classe de respostas como equivalente ao nome da classe de comportamento significa que estas podem ser de tão variadas formas que sua explicitação tem como consequência determinar que apenas por meio de uma resposta específica a classe de comportamento ocorre. No entanto, uma mesma classe comportamento pode ocorrer por meio de diversas respostas. A riqueza de respostas que é possibilitada ao aluno como forma de aprender uma determinada classe de comportamento torna a explicitação prévia de uma classe de resposta uma condição restritiva para sua aprendizagem, por aumentar a probabilidade de que se compreenda que esta resposta é a única possível de ser apresentada para que uma determinada classe de comportamento ocorra. Da mesma forma, a explicitação de quais respostas são as mais adequadas para a ocorrência de uma dada classe de comportamento é uma questão a ser verificada experimentalmente.

A literatura examinada apresentou ênfase em alguns dos aspectos de programar computadores em detrimento de outros. Em todas as cadeias de comportamentos apresentadas, é possível ainda encontrar mais classes de comportamentos, com um grau maior de especificidade, que torne mais gradual e evidente o processo de programar computadores. O nível de

detalhamento depende do que a literatura apresentou como informação que possibilitou derivar classes de comportamentos e organizá-los em uma sequência comportamental. Caso outras literaturas tivessem sido utilizadas, poderiam fornecer mais informações a respeito de possíveis classes de comportamentos que tornam as sequências de ensino mais graduais. É provável que, ao apresentar as sequências e cadeias em um programa de ensino, ocorram acréscimos de classes de comportamentos intermediárias identificadas posteriormente ou de classes de estímulos antecedentes e consequentes que tornem mais claro a classe de comportamento a ser ensinada.

Cada categoria elaborada apresentou dificuldades e contribuições específicas para elucidar o processo de programar computadores. No que se refere à categoria 'Avaliar argumentos de acordo com regras lógicas', a identificação dos princípios básicos da lógica, a elaboração do raciocínio e a capacidade de avaliar argumentos (classes de comportamentos relacionados ao que é denominado na literatura como 'raciocínio lógico') contribuem para pensar de forma lógica e estruturada facilitando a ocorrência das classes de comportamentos das demais categorias que compõem a classe geral 'programar computadores'. Navega (2005) apresenta a relação entre as regras da lógica e a capacidade de elaborar argumentos. A lógica tem uma grande importância na elaboração do raciocínio, pois possibilita a construção de argumentos válidos (matematicamente corretos). Ao construir um algoritmo ou escrever um programa de computador há a necessidade de aplicar regras lógicas, utilizar tabelas-verdade (tipo de tabela matemática usada em Lógica para descrever e avaliar a validade de expressões lógicas), elaborar condições de decisão (como nas estruturas de seleção) etc. A construção de argumentos produz soluções para problemas de forma mais organizada e estruturada, aspectos de grande importância para elaborar algoritmos e escrever programas de computador. O computador faz exatamente o que for instruído, logo, se uma solução é lógica (correta), sua representação computacional também será. Por isso a cadeia apresentada nessa categoria pode ser considerada uma cadeia de comportamentos pré-requisitos para resolver problemas, construir e formalizar algoritmos, escrever e avaliar programas de computador.

É comum o relato de professores de que falta capacidade de abstração e raciocínio, mencionados como aspectos fundamentais para a aprender a programar computadores (Santos & Costa, 2006; Hinterholz Jr 2009, citado em Prietch & Pazeto, 2010). Porém, a identificação das classes de comportamentos relacionados a tais capacidades não estão claros. Na literatura, estão explicitados em forma de grandes classes, inviabilizando o esclarecimento das classes de comportamentos necessárias

para o aluno aprender a programar computadores. As classes de comportamentos da categoria 'Avaliar argumentos de acordo com regras lógicas' podem contribuir para desenvolver as capacidades de abstração e raciocínio, e embora também seja ainda uma categoria não tão específica, avança ao propor uma sequência de ensino de classes de comportamentos básicos e ao incluí-la como parte do processo de programar computadores, iniciando pela identificação dos princípios básicos da lógica, o que dá condições ao aluno de elaborar argumentos de acordo com esses princípios e posteriormente, de avaliá-los.

Em relação à capacidade de abstração, indicada como um dos problemas de quem não consegue programar computadores, há poucas classes de comportamentos indicadas na literatura pesquisada que possibilitam maior clareza do que caracteriza a abstração ao programar computadores (Santos & Costa, 2006; Hinterholz Jr 2009, citado em Prietch & Pazeto, 2010). É importante diferenciar o comportamento da operação denominada abstração. Abstração não é um comportamento em si, se refere a uma operação que ocorre por meio de diversos comportamentos, assim como 'aprendizagem' e 'generalização'. Boratti (2004) considera que as operações de abstração evidenciam a forma que o ser humano organiza e modela o mundo. Trata-se, portanto, de operações de representação de objetos. Quando se utiliza a palavra abstração, geralmente se encobre um fenômeno pouco claro: por exemplo, se diz comumente que o aluno com dificuldade em programar computadores tem essa dificuldade por não conseguir abstrair. Mas é preciso identificar que coisas ele precisa abstrair, o grau de dificuldade, a relação dessa dificuldade com outras características e quais comportamentos possibilitam ele desenvolver tal capacidade. Quando essas características são especificadas, provavelmente deixa de ser necessário se referir à abstração para indicar o que ocorre. As classes de comportamentos relacionados a caracterização e representação de problemas, da categoria 'Resolver problemas' são exemplos de classes de comportamentos que contribuem para desenvolver tal capacidade, pois para que eles ocorram é necessário selecionar propriedades específicas dos estímulos a serem caracterizados e representados e descartar as demais propriedades que não contribuem para a caracterização e representação.

As classes de comportamentos da categoria 'Caracterizar funcionamento de computadores' também podem ser consideradas pré-requisitos por serem genéricos em relação à classe de comportamento 'Programar computadores', ou seja, não são caracterizadores do processo de programar computadores e sim classes de comportamentos amplas que podem ser necessárias para a ocorrência de outras classes de comportamentos.

Por exemplo, para 'Consertar computadores' é necessário caracterizar estrutura e funcionamento dos computadores, duas classes de comportamentos também pré-requisitos para programar computadores. Se um aluno não sabe caracterizar um computador dificilmente será capaz de programá-lo. As classes de comportamentos dessa categoria contribuem para que o aluno compreenda o que é um computador, objeto onde ele realiza a programação, e sua função ao programá-lo, facilitando a ocorrência de classes de comportamentos que se relacionam à interação com o computador. A diferenciação do uso do computador como editor ou executor de programas auxilia ao aluno a delimitar seu âmbito de trabalho, já que programar computadores envolve editar (ou criar) programas e não apenas utilizá-lo (executar programas), como faz um usuário comum. Considerando que tais classes de comportamentos se referem a aspectos básicos, não podem ser desprezados sob o risco de comprometer toda a cadeia de comportamentos envolvida na programação de computadores. Esses dados contribuem para o desenvolvimento de classes de comportamentos básicos dos alunos, que são pré-requisitos necessários para que outras classes de comportamentos mais complexas da classe geral programar computadores possam ocorrer.

No que se refere à categoria 'Resolver problemas', esta envolve diversas sequências de classes de comportamentos desde a sua caracterização até a solução elaborada para o problema. Ao resolver um problema, nem sempre este é caracterizado. As classes de comportamentos da sequência 'Caracterizar problemas' auxiliam o desenvolvimento da solução de problemas por tornar mais claro quais os aspectos importantes a serem resolvidos. Por exemplo, a classe de comportamento 'diferenciar demandas recebidas de problemas a resolver' contribui para que o problema seja extraído a partir de demandas, pois demandas de diferentes pessoas podem ser compreendidas como problemas prontos para serem resolvidos. Demandas não caracterizam o que é necessário solucionar e sim o que surge como problema para uma pessoa. Diferenciar uma demanda de um problema, este sim possível de ser resolvido, é uma aprendizagem importante para produzir soluções corretas. Se um programador aceita um problema sem perceber que nele existem dados implícitos, demanda no lugar de um problema claramente expresso, sem definir o objetivo do problema a resolver, não é possível que seu programa funcione corretamente ou que produza a solução esperada. É provável que durante a elaboração da solução, ocorram erros diversos que estão em sua origem, na caracterização deficiente do que é o problema a resolver.

Delimitar escopo do problema é outra classe de comportamento da categoria 'Resolver problemas' que contribui para a resolução do mesmo.

A falta de delimitação do escopo do problema tem como consequência uma análise precária do mesmo e uma falta de clareza a respeito de qual é o problema a resolver, pois é pelo escopo que se definem os limites do problema. Sem essa delimitação, há dificuldade em planejar o que será necessário para resolver o problema. A delimitação do que é o problema complementa o que é feito ao caracterizá-lo, e facilita o início de sua decomposição. Fazer a delimitação do escopo do problema é avançar em direção à decomposição, pois antes de decompor é preciso ter claro quais são as partes do problema e o que é central no problema a ser resolvido. As classes de comportamentos da sequência 'Representar problemas' possibilitam aumentar a compreensão e análise do problema e a estruturação de sua solução. Uma representação precária do problema tende a dificultar a decomposição e a resolução do mesmo. Já o processo de decomposição de problemas, da cadeia 'Decompor problemas' possibilita a diminuição da complexidade do mesmo, torna mais evidente como cada parte do problema se relaciona e facilita sua resolução e posterior construção de algoritmos. Esses dados ajudam a superar dificuldades em programar computadores, pois raramente são ensinados os comportamentos da categoria 'Resolver problemas', pelo pressuposto de que os alunos já sabem fazê-lo ou de que essas classes de comportamentos serão aprendidas de forma indireta ao escrever um programa de computador.

Em relação à categoria 'Construir algoritmos' é necessário considerar que estes são a base dos programas de computador. Sem o conhecimento (explícito ou implícito) de como construí-los, a elaboração de um programa de computador torna-se muito mais difícil. A diferenciação entre algoritmo e programas de computador proporciona mais clareza acerca do que é necessário para programar computadores. Um algoritmo não deve ser confundido com um programa de computador, pois embora apresentem semelhanças, aquele não tem todas as características de um programa escrito em uma linguagem de programação. O algoritmo é mais simples e mais facilmente compreendido pelas pessoas, por não exigir o uso das regras sintáticas e semânticas de uma linguagem de programação, sendo um passo intermediário essencial para a aprendizagem de programar computadores.

As instruções são o que formam os algoritmos. Por meio das instruções de um algoritmo os dados são alterados produzindo a solução do problema. Aprender a elaborar instruções facilita ao aluno organizar e estruturar cada etapa para produzir a solução de um problema em forma de algoritmo. As classes de comportamentos da cadeia 'Construir algoritmos' propiciam um entendimento mais completo do que faz um algoritmo e de

como as instruções interagem entre si para produzir a solução do problema. Dentre os tipos de instruções, estão aquelas que representam estruturas de controle de fluxo. O conhecimento a respeito dessas estruturas possibilita ao aluno organizar as instruções de forma a potencializar a leitura, elaboração e execução do algoritmo. Conhecer as estruturas básicas que servem para controlar o andamento da sequência das instruções de um algoritmo também proporciona um aumento da probabilidade de aprender a programar computadores. Outro aspecto que favorece o entendimento de como elaborar um algoritmo é a classe de comportamento ler algoritmos (Weinberg, 1998). Para elaborar algoritmos é necessário aprender a ler algoritmos seguindo suas instruções. Instruções de algoritmos não possibilitam realizar uma leitura linear do mesmo e por isso, nem sempre a melhor maneira de ler um algoritmo é linear. A construção de algoritmos exige clareza acerca das instruções, das estruturas de controle de fluxo, de como descrever essas instruções e estruturas para que se produza a solução do problema de forma mais próxima de um programa de computador.

Para que a transformação de algoritmos em programas de computador ocorra de maneira mais gradual, o processo de formalizar um algoritmo torna-se essencial como condição de ensino. Nesse processo, diversos critérios de formalização são atendidos, como aprimoramento do algoritmo, maior precisão e eliminação de ambiguidades, entre outros. Esse processo não é uma simples tradução de um algoritmo para uma linguagem de programação, pois as características da linguagem de programação escolhida interferem na própria solução elaborada em forma de algoritmo, que poderá sofrer novas alterações para que seja possível sua transformação em uma linguagem de programação. Ao escolher uma linguagem de programação e iniciar a formalização, detalhes que ainda não tinham sido considerados podem ser evidenciados, pois nas linguagens de programação há muito mais detalhamento a ser feito, sendo necessário especificar exatamente como o código deve ser escrito. O paradigma e a linguagem de programação determinam, em partes, como será elaborado o algoritmo e que transformações serão feitas com o algoritmo para transformá-lo em um programa escrito em uma determinada linguagem de programação.

Embora a categoria 'formalizar algoritmos' avance em relação ao processo de transformação de um algoritmo para um programa de computador, é provável que mais classes de comportamentos intermediários sejam necessários para tornar mais explícito como ocorre essa transformação. A classe de comportamento denominada formalizar algoritmos ainda não torna totalmente evidente os aspectos necessários para ocorrer

a transformação de algoritmo para programa de computador, como classe de estímulos antecedentes e consequentes que explicitem o que se deve levar em conta e o que obter ao fazer tal transformação. No entanto, as classes de comportamentos explicitadas nessa categoria auxiliam a tornar a etapa antes denominada como 'tradução' um pouco mais gradual e clara, facilitando essa aprendizagem que produz grande parte das desistências e constatações de fracasso escolar.

É importante considerar também que, na prática, a elaboração do algoritmo e sua formalização podem ser realizadas conjuntamente. Para algoritmos mais simples (e para programadores experientes) muitas vezes a linguagem de programação é utilizada como ferramenta para representar logicamente a solução de forma direta e o algoritmo puro e completo não aparece explicitamente no processo. Isso não significa que as etapas de elaboração e formalização de algoritmos não são executadas, mas sim que esses podem ser processos encobertos juntamente com as classes de comportamentos de escrever programas de computador.

Weinberg (1998) afirma que nas universidades, em vez de tentar ensinar os princípios, ou pelo menos linguagens de programação contrastantes, ou uma linguagem de programação que represente as principais possibilidades, as escolas ensinam a programar em uma única linguagem simples e artificial. Com isso o estudante aprende a escrever algum tipo de programa rapidamente, mas à custa de limitar seu crescimento futuro como programador. A caracterização das linguagens de programação é importante por reduzir as limitações ao desenvolver programas e aumentar a variedade de processos intelectuais pela aprendizagem de novas construções de linguagens (Sebesta, 2003). Essa caracterização explicitada na cadeia 'Identificar equivalência de instruções em linguagem natural e de programação' possibilita ao aluno aprender não apenas uma determinada linguagem de programação, mas sim os princípios que não estão necessariamente vinculados a uma determinada linguagem. Priorizar a aprendizagem de uma linguagem de programação em específico aumenta a probabilidade de desvio dos comportamentos-objetivos relacionados a programar computadores para construir programas específicos naquela linguagem, com a expectativa de que ocorra generalização dos comportamentos aprendidos, mas sem proporcionar reais condições para que isso seja possível. As classes de comportamentos das cadeias de 'Formalizar algoritmos' são outros aspectos que proporcionam compreensão para aprender a programar computadores que não apenas a aprendizagem de uma linguagem de programação específica.

Aprender a programar apenas orientando-se por uma linguagem de programação específica pode trazer problemas e dificuldades na aprendizagem. McIver (2001) afirma que há um impacto considerável da primeira linguagem de programação na aprendizagem do principiante. A linguagem na qual um programa é desenvolvido limita os tipos de estruturas de controle, de dados e abstrações que se pode usar, fazendo com que as formas de algoritmos possíveis de serem construídas também sejam limitadas (Sebesta, 2003). A escolha de uma determinada linguagem de programação influencia o nível e as características da formalização a ser realizada e em como o problema será resolvido. Essas características necessitam ser consideradas ao formalizar um algoritmo. Por isso, embora possa ser forçoso aprender por meio de alguma linguagem de programação, é necessário considerar a classe de comportamento a ser aprendida, classes de comportamentos básicas e genéricas para garantir flexibilidade na aprendizagem do que é nuclear em programar computadores. Nesse sentido, outras classes de comportamentos além daquelas relacionadas com o estudo das linguagens de programação auxiliam a eliminar as restrições impostas pela necessidade de estudar por meio de uma linguagem de programação, como aquelas referentes a caracterização do processo de formalização e do processo de programar computadores. Não importa se uma ou outra linguagem de programação ou ambiente de programação é escolhido, mas quais as classes de comportamentos de programar computadores devem ser ensinadas, seja com que linguagem for.

Para escrever programas de computador, o conhecimento de variáveis é fundamental. A importância das classes de comportamentos relacionadas à variáveis se evidencia quando o indivíduo não é capaz de caracterizar uma variável, por exemplo. Sem conseguir caracterizar uma variável, não é possível elaborar instruções, armazenar informações, entre outras classes de comportamentos que ocorrem ao programar computadores. O processo de nomear variáveis tem implicações na escrita de um programa de computador. O nome de uma variável deve ser significativo para facilitar o entendimento do código do programa e da função da variável (Sebesta, 2003). Variáveis em programas de computador se referem a um estado que se altera a depender das instruções do algoritmo ou código de programação. O desconhecimento do que é uma variável pode ocasionar problemas no processo de programar computadores, já que é um conceito que está, explícita ou implicitamente, presente nesse processo. Esse conceito auxilia na análise dos problemas, entendimento de como funciona a memória de um computador e armazenamento de dados e na discriminação de estímulos em um código.

A importância de diferenciar o nome atribuído a uma variável em relação ao que é função da variável se refere a um equívoco que alunos cometem ao programar computadores. Putnam e outros (1986, citado em Dehnadi, 2009) ao examinar equívocos dos alunos do ensino médio ao escrever em uma linguagem de programação, observa que estes tentam usar nomes significativos para as variáveis que criam, na expectativa de que o computador leia, entenda o nome e perceba suas intenções. Alunos iniciantes costumam pensar que, ao nomear uma variável com um determinado nome, este por si só irá fazer aquilo que o nome indica, ou que o computador vai 'entender' o que é para fazer, apenas pelo nome ser 'significativo'. Atribuir um nome significativo não implica na variável fazer algo correspondente ao nome dado. A classe de comportamento 'Diferenciar o nome de uma variável em relação ao que ela faz' auxilia na elucidação desse tipo de confusão que alunos iniciantes costumam incorrer.

A avaliação dos programas de computador é fundamental para corrigir erros e aperfeiçoar o comportamento do programador. Há muitas formas de avaliar programas de computador, e para fazê-lo não é suficiente medir a porcentagem de códigos estranhos ou periféricos. O estudo da programação como atividade humana exige uma avaliação mais complexa dos programas, em que muitas variáveis necessitam ser consideradas, mais do que apenas afirmar que um programa é 'bom' ou 'ruim' (Weinberg, 1998). Weinberg (1998) conclui que é raro encontrar dois programas semelhantes que possibilitam uma comparação em minúcia e uma conclusão de qual deles é superior em todas as características. Para este autor, os principais critérios para avaliar um programa se referem a quanto ele é compreendido por outros e quanto ele não precisa de manutenção. A elaboração de critérios bem explícitos do que será considerado um bom programa auxilia em sua elaboração, correção e manutenção.

Embora ainda possa ser mais detalhado conforme literatura, as cadeias de comportamentos organizadas nas sete categorias apresentadas tornam o processo de programar computadores mais claro e mais completo, pois diversas classes de comportamentos foram organizadas em um processo contínuo que expressa cada etapa de programar computadores. As cadeias de comportamentos evidenciam que o estudo das linguagens de programação corresponde a uma parte do estudo de programação e não a sua totalidade (Santos e Costa, 2005, pg. 3). Embora se observe mais explicitamente as classes de comportamentos referentes à escrita de programas de computador quando se está programando, as cadeias de comportamentos anteriores a essa categoria são tão importantes para que um programa de computador seja escrito quanto são as classes de comportamentos referentes ao uso de uma linguagem de programação. A partir

dessa organização há diminuição da probabilidade de serem formulados falsos objetivos de ensino, pois fica evidente a relação entre o que o aluno apresenta como repertório inicial e o que ele precisa apresentar como comportamento-objetivo. O ensino das classes de comportamentos relacionados a essas categorias possibilita um entendimento mais amplo do processo de programar computadores, com a caracterização de conceitos essenciais, e também possibilita avaliar se os alunos aprenderam a programar computadores.

Em relação ao programador profissional, a importância da observação do próprio comportamento e de considerar que o comportamento humano é parte principal na programação de computadores se evidencia no que Weinberg (1998) observou em seus alunos: sem a introspecção do aluno, problemas, dificuldades e insights não são observados, e como consequência, os problemas enfrentados ao programar um computador são compreendidos como erros do computador. O programador não é capaz de olhar para seu próprio comportamento e corrigi-lo. Ao contrário, ele reclama do computador, que não está fazendo o que ele escreveu para ser feito. Nesse sentido, o desenvolvimento de profissionais críticos em relação ao seu próprio comportamento é um dos efeitos de um programa de ensino que considera programar computadores um conjunto complexo de comportamentos.

Esse trabalho contribui como uma possibilidade de fornecer melhores condições de ensino por meio de explicitação de diversas classes de comportamentos que compõem a classe geral programar computadores. Os resultados apresentados evidenciam o que Weinberg (1998) afirma: que bons programadores são feitos, não nascem; portanto devemos voltar a atenção para a produção ou processo de formação de programadores. Tal constatação possibilita observar a classe geral de comportamentos 'Programar computadores' como possível de ser aprendido e ensinado, e não como algo que algumas pessoas têm e outras não têm possibilidade de aprender, como deficiências ou habilidades inatas. As sequências de classes de comportamentos apresentadas contribuem para a construção de programas de ensino mais completos, graduais, que diminuem a probabilidade do aluno errar, fracassar e desistir de aprender a programar computadores. Nas sequências de cada categoria são considerados desde classes de comportamentos de caracterização e classes de comportamentos que favorecem a aprendizagem de habilidades básicas até classes de comportamentos consideradas típicas do ensino de programar computadores. Com isso, grande parte das dificuldades de aprendizagem em programar computadores são trabalhadas pontualmente, e são garantidas

melhores condições para que alunos com diferentes repertórios e habilidades prévias possam aprender a programar computadores.

REFERÊNCIAS

ABBAGNANO, N. **Dicionário de filosofia**. 5. ed. São Paulo: Martins Fontes, 2007.

AGUIAR R. S., OEIRAS, J. Y. Y. **Laboratório de Linguagens Formais - LabLF**. Laboratório de Informática na Educação da Universidade Federal do Pará, Revista Brasileira de Informática na Educação, v. 18, n. 1, 2010.

ASCENCIO, A. F. G., CAMPOS, E. A. V. **Fundamentos da Programação de Computadores - Algoritmos, Pascal, C/C++ e Java**. 2a Edição, Ed. Pearson/Prentice Hall, São Paulo, 2007, 448p.

AVIZ JÚNIOR, A. A. **A Aprendizagem de algoritmos: relato de uma experiência no Curso de Tecnologia em Informática do CEFET/PA**. Dissertação (Mestrado) - Programa de Pós-Graduação em Educação em Ciências e Matemáticas, Núcleo Pedagógico de Apoio ao Desenvolvimento Científico, Universidade Federal do Pará, Belém, 2007.

BORATTI, I. C. **Programação Orientada a Objetos em Java**. Visual Books, 2007.

BOTOMÉ, S. P. **A noção de comportamento**. Em H.P. De Moraes Feltes e U. Zille. Filosofia: Diálogo de Horizontes. Porto Alegre: EDIPU-CRS, p. 685-708, 2001.

BOTOMÉ, S. P. **Ensino Programado**. Texto não publicado. Tradução e adaptação feita para uso didático, em 1970, de um texto de 1961, sem identificação de autoria ou editora da Teaching Machines Incorporated - A division of Croler, Incorporated, 1961.

BOTOMÉ, S. P. **O problema dos falsos 'objetivos de ensino'**. Em S. P. Botomé. Objetivos de ensino, necessidades sociais e tecnologia educacional. Trabalho premiado no Concurso Nacional de Monografias sobre Tecnologia Educacional - II Concurso Roquete Pinto de Monografias, promovido pela FUNTEVE do Ministério da Educação, p. 102-122, 1985.

BOTOMÉ, S. P. **Um procedimento para encontrar os comportamentos que constituem as aprendizagens envolvidas em um objetivo de**

ensino. São Carlos (SP): Universidade Federal de São Carlos. Não publicado. 1975.

BOTOMÉ, S. P. Repertório de entrada dos aprendizes para um programa de ensino. Texto escrito para uso interno no módulo sobre ensino do curso de Especialização em Arquitetura e Urbanismo, realizado na Universidade de Caxias do Sul, 1996, a partir de uma versão testada com estudantes de Psicologia da PUC de São Paulo, 1977.

BOTOMÉ, S. P., KUBO, O. M. Objetivos de ensino como problema no ensino superior. Texto não publicado, elaborado como material didático para uso interno no Programa de Pós-graduação em Psicologia da Universidade Federal de Santa Catarina, 2003.

BOTOMÉ, S. P., KUBO, O. M., SOUZA, D. G. Generalidade e especificidade da linguagem e sua relação com classes de ações humanas. (Texto didático) Universidade Federal de Santa Catarina, Florianópolis, 2004.

BRASIL. MINISTÉRIO DE EDUCAÇÃO. Diretrizes Curriculares dos cursos de Bacharelado em Ciência da Computação, Engenharia de Computação, Engenharia de Software e Sistemas de Informação e dos cursos de Licenciatura em Computação. 2003. Disponível em: http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view_category&catid=36

BRASIL. MINISTÉRIO DE EDUCAÇÃO. Sinopse da Educação Superior. 2009.

BROOKSHEAR, J. G. Ciência da Computação: uma visão abrangente. 7ª edição, Porto Alegre: Artmed, 2003, 512 p.

CATANIA, A. C. Aprendizagem: comportamento, linguagem e cognição (4ª ed). Porto Alegre: Artmed. 1999, original publicado em 1998.

CRISTOVÃO, H. M. Aprendizagem de Algoritmos num Contexto Significativo e Motivador: Um Relato de Experiência. Anais do XXVIII Congresso da SBC - WEI – Workshop sobre Educação em Computação, 2008.

DEHNADI, S. **A cognitive study of learning to program in introductory programming courses.** PhD thesis, Middlesex University, 2009.

DELGADO, C. A. D. M.; XEXÉO, J. A. M.; SOUZA, I. F.; RAPKIEWICZ, C. E.; PEREIRA JUNIOR, J. C. R. **Identificando competências associadas ao aprendizado de leitura e construção de algoritmos.** In: XXV Congresso da SBC - X Workshop de Educação em Informática, São Leopoldo RS. Anais do XXV Congresso da SBC, v. 1, 2005.

DUARTE, A., MOREIRA, H., MELLO, T. S. **Competitividade como Fator Motivacional para o Estudo de Computação.** Anais do XXI SBIE, Simpósio Brasileiro de Informática na Educação (SBIE), João Pessoa, 2010. Disponível em: http://www.ccae.ufpb.br/sbie2010/anais/Artigos_Resumidos_files/75021_1.pdf

ESTEVES, M., ANTUNES, R., MORGADO, L., MARTINS P., FONSECA, B. **Contextualização da aprendizagem da programação: estudo exploratório no Second Life®.** Conferência IADIS Ibero-Americana WWW/Internet, 2007.

FOLHA ONLINE. **Matemática e ciências da computação têm alta taxa de abandono.** 2009. Folha.com. Acesso em 25/09/2011. Disponível em <http://www1.folha.uol.com.br/folha/educacao/ult305u546576.shtml>

FOLLETTE, W. C., NAUGLE, A. E., LINNEROOTH, P. J. **A functional alternative to traditional assessment.** Tradução de Maly Delitti. M. J. Dougher (Ed.), Clinical Behavior Analysis, Reno, NV: Context Press, p. 99-125, 1999.

FORBELLONE A., EBERSPACHER, H. **Lógica de programação.** 3ª Edição, Editora Pearson Prentice Hall, 2005, 232 p.

GERSTING, J. L. **Fundamentos Matemáticos para a Ciência da Computação.** 3ª Edição, LTC Editora, 1995, 538 p.

GUERREIRO, P. **A Mesma Velha Questão: Como Ensinar Programação?** Memória do Quinto Congreso Iberoamericano de Educación Superior en Computación, México City, Ed. UNAM, 1986.

HEGENBERG, L. **Dicionário de Lógica**. São Paulo: EPU, 1995.

HERNANDEZ, C. C., SILVA, L., SEGURA, R. A., SCHIMIGUEL, J., LEDÓN, M. F. Pa., BEZERRA, L. N. M., SILVEIRA, I. F. **Teaching Programming Principles through a Game Engine**. CLEI Electronic Journal, v. 13, n. 2, paper 3, 2010.

HINTERHOLZ JR, O. **Tepequém: uma nova Ferramenta para o Ensino de Algoritmos nos Cursos Superiores em Computação**. In: XVII WEI, Bento Gonçalves/RS, 2009.

HORSTMANN, C. **Conceitos de computação com o essencial de JAVA**. Porto Alegre: Bookman, 3ª edição, 2003.

JABŁONOWSKI, J. **A Case Study in Introductory Programming**. International Conference on Computer Systems and Technologies, p.131-137, 2007.

JESUS, A.; BRITO, G. S. **Concepção de ensino-aprendizagem de algoritmos e programação de computadores: a prática docente**. I ENI-NED - Encontro Nacional de Informática e Educação, Vol. 9, n. 16, 2009.

KIENEN, N., WOLFF, S. **Administrar comportamento humano em contextos organizacionais**. In: Psicologia: organizações e trabalho. v. 2, n.2, p. 11-37, 2002.

KNUTH, D. E. (1973). **The Art of Computer Programming. Volume 1: Fundamental Algorithms**. Addison-Wesley, 3ª ed., 1997.

KUBO, O. M., BOTOMÉ, S. P. **Ensino-Aprendizagem: uma interação entre dois processos comportamentais**. Interação, Curitiba, v. 5, p. 123-132, 2001.

LIMA, M. R., LEAL, M. C. **Uso da linguagem logo no ensino superior de programação de computadores**. Revista Eletrônica Multidisciplinar Pindorama do Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA, n. 01 - Ano I - Ago. 2010.

MATTOS, M. M. **Construção de abstrações em lógica de programação.** Em XX Congresso Nacional da Sociedade Brasileira de Computação, v. 1. Editora Universitária Champagnat, 2000.

MENEZES, W. S., COELLO J. M. A. **Formação de Grupos em Ambientes de Aprendizado Colaborativo de Programação Usando Estilos de Aprendizagem e Elementos da Teoria do Conflito Sócio-cognitivo.** Anais do XXVI Congresso da SBC - XIV Workshop sobre Educação em Computação, 2006.

MILLENSON, J. R. **Princípios de análise do comportamento** (A. A. Souza & D. Rezende, Trads.). Brasília, DF: Editora Coordenada, 1975.

OLIVEIRA, A. B.; BORATTI, I. C. **Introdução a programação: algoritmos.** Visual Books, 2004, 146 p.

PAIXÃO, W. **Aprendendo a raciocinar: lógica para iniciantes.** São Paulo, Humanitas, 2007.

PEREIRA JÚNIOR, J. C. R. **AVEP - um ambiente virtual para apoio ao ensino de algoritmos e programação.** Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Estadual do Norte Fluminense, Campos dos Goytacazes, 2006.

PEREIRA JÚNIOR, J. C. R.; RAPKIEWICZ, C. E.; DELGADO, C.; XEXÉO, J. A. M. **Ensino de algoritmos e programação: uma experiência no nível médio.** Anais do XXV Congresso da SBC, São Leopoldo, RS. v. 1, 2005.

PEREIRA JÚNIOR, J. C. R., RAPKIEWICZ, C. E. **O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma visão crítica da pesquisa no Brasil.** In: I Workshop de Educação em Computação RJ/ES, 2004, Vitória - ES. I Workshop de Educação em Computação RJ/ES, v. 1. 2004.

PINHEIRO, M. C. **Uma Experiência no Ensino de Lógica de Programação para Cursos de Engenharia Usando o Pascal e o Logo.** II Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG), Poços de Caldas, 2003.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo, Makron Books, 1995, 1028 p.

PRIETCH, S. S., PAZETO, T. A. **Estudo sobre a Evasão em um Curso de Licenciatura em Informática e Considerações para Melhorias**. WEIBASE/ ERBASE, Maceió/AL, 2010.

ROCHA, R. **Utilização da robótica pedagógica no processo de ensino aprendizagem de programação de computadores**. Dissertação (Mestrado em Educação). Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG. Belo Horizonte, 2006.

SAID, R. **Curso de lógica de Programação**. São Paulo: Digerati Books, 2007, 144 p.

SANTOS, R. P., COSTA, H. A. X. **Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática**. INFOCOMP - JOURNAL OF COMPUTER SCIENCE, Lavras/MG – Brasil, v. 5, n. 1, p. 41-50, 2006.

SANTOS, R., COSTA, H. A. X. **TBC-AED e TBC-AED/WEB: Um Desafio no Ensino de Algoritmos, Estruturas de Dados e Programação**. In: IV WEIMIG – IV Workshop em Educação em Computação e Informática do Estado de Minas Gerais, Varginha/MG – Brasil. CD dos Anais do IV Workshop em Educação em Computação e Informática do Estado de Minas Gerais, v. 1. 2005.

SEBESTA, R.W. **Conceitos de linguagem de programação**. 5ª edição. Porto Alegre: Bookman, 2003, 638 p.

SILVA FILHO, R. L. L.; MOTEJUNAS, P. R.; HIPÓLITO, O.; LOBO, M. B. C. M. **A evasão no ensino superior brasileiro**. Cadernos de Pesquisa, São Paulo, v. 37, n. 132, p. 641-659, set./dez. 2007. Disponível em: <http://www.scielo.br/pdf/cp/v37n132/a0737132.pdf>

SIPSER, M. **Introduction to the Theory of Computation**. Brooks/Cole Pub Co. 2a Edição 2005.

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. Diretoria de Educação da Sociedade Brasileira de Computação, 2010. Acesso em

25/09/2011. Disponível em: http://www.sbc.org.br/index.php?option=com_content&view=article&id=130:descricao&catid=190:chapas-diretoria&Itemid=140

SOUZA, C. M. **VisuAlg - Ferramenta de Apoio ao Ensino de Programação**. Revista TECCEN - vol. 2 - n. 2 - set. 2009.

UNIVERSIDADE FEDERAL DE SANTA CATARINA. **Boletim de dados 2007**. Departamento de Integração e Estatística, 2007. Disponível em: http://www.pip.ufsc.br/arquivos/BOLETIM_DADOS_2007.pdf

VARGAS, K. S. **Ferramenta para apoio ao ensino de Introdução à Programação**. Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau, Blumenau, 2005.

WEINBERG, G. M. **The Psychology of Computer Programming: Silver Anniversary Edition**. New York: Dorset House Publishing, 1998, 292 p.

WEIZENBAUM, J. **Computer Power and Human Reason: From Judgement to Calculation**. San Francisco: W. H. Freeman, 1976.

XAVIER, G. M. C. e outros. **Estudo dos Fatores que Influenciam a Aprendizagem Introdutória de Programação**. IV ERBASE – Escola regional de computação Bahia-Sergipe. Anais eletrônicos. Feira de Santana: Universidade Estadual de Feira de Santana, 2004.

YNOGUTI, C. A. **Uma metodologia para o ensino de algoritmos**. In: Global Congress On Engineering And Technology Education, 2005, Santos, Brasil, 4 p.

ZABALA, A. **A prática educativa. Como ensinar**. Porto Alegre: ArtMed, 1998.