UNIVERSIDADE FEDERAL DO ESPIRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM PSICOLOGIA

DOUTORADO EM PSICOLOGIA

HUGO CRISTO SANTANNA

Ação, Computação, Representação

Uma investigação psicogenética sobre o desenvolvimento do Pensamento Computacional

VITÓRIA
2014

HUGO CRISTO SANT'ANNA

Ação, Computação, Representação
Uma investigação psicogenética sobre o desenvolvimento do Pensamento Computacional

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Psicologia, como parte dos requisitos para obtenção do título de Doutor em Psicologia, da Universidade Federal do Espírito Santo.

Orientadora: Prof[a]. Dr[a]. Maria Cristina Smith Menandro

VITÓRIA
2014

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Sant'Anna, Hugo Cristo, 1979-
S232a Ação, computação, representação : uma investigação psicogenética sobre o desenvolvimento do pensamento computacional / Hugo Cristo Sant'Anna. – 2014.
199 f. : il.

Orientador: Maria Cristina Smith Menandro.
Tese (Doutorado em Psicologia) – Universidade Federal do Espírito Santo, Centro de Ciências Humanas e Naturais.

1. Inteligência computacional. 2. Representações Sociais. 3. Ciências cognitivas. 4. Aprendizagem. 5. Designers. I. Menandro, Maria Cristina Smith, 1962-. II. Universidade Federal do Espírito Santo. Centro de Ciências Humanas e Naturais. III. Título.

CDU: 159.9

Ação, Computação, Representação: Uma investigação psicogenética sobre o desenvolvimento do Pensamento Computacional.

Hugo Cristo Sant'Anna

Tese de Doutorado apresentada ao Programa de Pós Graduação em Psicologia, como parte dos requisitos para obtenção do título de Doutor em Psicologia, da Universidade Federal do Espírito Santo.

Aprovada em 30 de maio de 2014 por:

Profa. Dra. Maria Cristina Smith Menandro (orientadora) – UFES

Prof. Dr. Rogério Câmara – UnB

Prof. Dr. Rafael Moura Coelho Pecly Wolter – UERJ

Profa. Dra. Zeidi de Araujo Trindade – UFES

Profa. Dra. Claudia Broetto Rossetti – UFES

Esta tese é dedicada a todos aqueles que se esforçaram para democratizar o acesso aos conhecimentos da Computação, seja pela construção de dispositivos mais baratos e simples de usar, seja pelo desenvolvimento de sistemas operacionais, aplicativos, linguagens de programação, jogos ou qualquer outro tipo de iniciativa que reduziu as barreiras de entrada para os iniciantes. Esta lista é incompleta e injusta, mas coerente com o meu próprio percurso: Seymour Papert, criador da linguagem LOGO; Alan Kay, realizador do conceito de computador pessoal; John G. Kemeny e Thomas E. Kurtz, criadores da linguagem BASIC; Microsoft Corporation, criadora do computador MSX, no qual aprendi a programar com BASIC; Borland Corporation, desenvolvedora dos meus primeiros compiladores de C/C++ e Assembler; Os *demogroups* e a *demoscene*, com agradecimento especial aos grupos Future Crew, Iguana, Electromotive Force, Nooon, Doomsday, Hornet e Sahara Surfers; os programadores Statix, Skal, Pelusa e muitos outros que compartilharam seus conhecimentos e códigos-fonte; *Demosceners* brasileiros (e amigos) do TheEnd e Imago, que fizeram parte da minha juventude do desenvolvimento do meu próprio pensamento computacional – F0x, Zeh, MNC, Ed, Fremen, Lady, Vuds e Simak; Meu amigo-primo-emprestado Rafael Guimarães (RPG), que serviu de mentor e inspiração quando ainda era jovem e não tinha as bases acadêmicas suficientes para desenvolver tudo que desejava (minha primeira zona de desenvolvimento proximal na Computação); John Maeda, fundador do *Aesthetics + Computation Group* no MIT, de onde surgiram projetos que me influenciaram e continuam presentes na minha vida – Design by Numbers e Processing; Mitch Resnick, coordenador do *Lifelong Kindergarten* no MIT, grupo responsável pela linguagem Scratch e pelo kit de robótica educacional LEGO Mindstorms; Meus alunos, de todas as instituições e grupos de pesquisa que fiz ou faço parte, pelo desafio de fazer por vocês o que essas pessoas fizeram por mim.

Dedico esta tese principalmente à minha mãe, Leni Cristo. Mesmo sem ter a menor ideia do que eu faria com um computador aos oito anos de idade, ela acreditou em um potencial que muitos ainda não acreditam, quase 30 anos depois. Ao contrariar as oportunidades tradicionalmente oferecidas às crianças da época, minha mãe deu início à construção de absolutamente tudo o que me tornei. Obrigado, mãe.

Agradeço primeiramente aos alunos do Programa de Pós-Graduação em Psicologia (PPGP) da UFES, que me receberam com toda paciência do mundo em 2005 e transformaram radicalmente minhas trajetórias profissional e acadêmica. Nos últimos nove anos aprendi a admirar e respeitar a pesquisa e os pesquisadores da área de Psicologia, encontrando em cada um dos meus colegas (muitos deles agora *meus amigos*) uma inspiração para os meus próprios objetivos. Optei por não mencionar explicitamente nenhum nome para que os agradecimentos contemplem todos vocês, indiscriminadamente.

O segundo agradecimento precisa ser feito à minha orientadora, Profa. Maria Cristina Smith Menandro, que sempre acreditou e apostou nas minhas ideias. A fagulha inicial para o desenvolvimento do trabalho que realizo hoje, articulando Design e Psicologia, surgiu nas aulas da "Cris" no mestrado (2005-2007). No doutorado, minha orientadora se fez presente da maneira mais importante: dando conselhos e fazendo apontamentos precisos nas horas certas; com o incentivo para seguir em frente numa pesquisa que era inteiramente nova para mim; e, principalmente, com uma confiança no meu trabalho que foi fundamental para que eu conseguisse chegar até aqui. Muito obrigado.

O terceiro agradecimento é direcionado aos demais professores do PPGP que, cada um a seu modo, estão presentes nesta tese. Se aprendi a gostar de tudo na Psicologia foi porque tive excelentes professores sempre dispostos a ouvir minhas questões e a dividir o que sabiam comigo, pacientemente. Cada disciplina cursada e trabalho entregue significou muito na minha formação e espero que esta tese faça justiça ao tempo que cada um de vocês investiu em mim ao longo dos últimos nove anos. Serei eternamente grato a todos.

Por fim, agradeço a todos que resistiram bravamente às minhas infinitas ausências, omissões e períodos de mau humor durante a realização deste trabalho, principalmente minha família e amigos. Concluir o doutorado foi uma das coisas mais difíceis e mais importantes que já fiz na vida e, por causa dele, tive que abandonar ou abrir mão de muitas outras coisas igualmente ou até mais importantes.

Amo todos vocês <3

"I'm not an atomic playboy."
– Second Reality (Future Crew, 1993)

"Is everybody in? Everybody in? The ceremony is about to begin."
– Verses (Electromotive Force, 1994)

Sant'Anna, H. C. (2014) Ação, Computação, Representação: Um estudo psicogenético sobre o desenvolvimento do Pensamento Computacional. Tese de Doutorado. Programa de Pós-Graduação em Psicologia. Universidade Federal do Espírito Santo, Vitória, ES, 199 pp.

O Pensamento Computacional ou *Computational Thinking* (CT) é definido como um conjunto de habilidades e competências tradicionalmente atribuídas a profissionais, estudantes e pesquisadores da Ciência da Computação, mas que nos últimos anos vem se tornando um conhecimento acessível a mais pessoas por meio de iniciativas que buscam popularizá-lo. Esta tese partiu de uma triangulação entre a Teoria das Representações Sociais na abordagem estrutural, a Psicologia Sócio-Histórica e a perspectiva funcional da Epistemologia Genética, visando elaborar uma proposta pedagógica construcionista para o desenvolvimento do CT entre os estudantes de Design do ensino superior. A pesquisa foi realizada em três etapas, sendo a primeira dedicada à caracterização do relacionamento dos estudantes com os computadores; ao mapeamento das formas por meio das quais aqueles dispositivos auxiliam na resolução de problemas de Design; e à investigação das representações sociais (RS) dos estudantes acerca dos Princípios da Computação. Foram entrevistados 86 estudantes, sendo 23 destes programadores. As análises prototípica e de conteúdo dos dados indicaram a prevalência das RS e práticas instrumentais envolvendo a Computação, objetivada e ancorada pelo computador como meio que a materializa e a produz. A segunda etapa partiu desses resultados para elaborar a abordagem pedagógica centrada na linguagem *RocketSocket*, cuja principal característica é contemplar diferentes perfis de aprendizes dos Princípios da Computação. A linguagem explora a narrativa de um foguete que pode ser programado para coletar estrelas e desviar de asteroides no espaço como estratégia para discutir conceitos, práticas e perspectivas do CT. A terceira etapa consistiu no estudo de caso prototípico que experimentou o potencial da linguagem *RocketSocket* com dois estudantes de Design, sendo um programador e outro não. Os dados foram analisados segundo a perspectiva das microgêneses cognitivas, observando como os conhecimentos dos participantes sobre a Computação e os computadores foram selecionados e avaliados quanto à sua pertinência funcional ao longo do processo de construção dos algoritmos que controlam o foguete. A última parte da tese discute os dois estudos de forma integrada no âmbito da triangulação teórica inicial, buscando refletir sobre o processo de concepção da linguagem a partir das RS dos estudantes e suas possibilidades de contribuição para a popularização do CT.

Palavras-chave: Pensamento Computacional, representações sociais, microgêneses, aprendizagem, Design Computacional.

Sant'Anna, H. C. (2014) Action, Informatique, Représentation: Une enquête psychogénétique sur le développement de la Pensée Computationnelle. Thèse de doctorat. Programme de Post-Graduation en Psychologie. Université Fédérale de Espírito Santo, Vitória, Brésil, 199 pp.

La Pensée Computationnelle ou Computational Thinking (CT) est définie comme un ensemble de capacités et compétences traditionnellement attribuées aux professionnels, étudiants et chercheurs de la Science Informatique, mais qu'aux dernières années est devenue une connaissance accessible à plus de gens grâce à des initiatives qui cherchent à la populariser. Cette thèse a été basée sur une triangulation entre la Théorie des Représentations Sociales dans l'approche structurelle, la Psychologie Socio-historique et la perspective fonctionnelle de l'Épistémologie Génétique, en visant à élaborer une proposition pédagogique constructiviste pour le développement du CT entre les étudiants de Design dans l'enseignement supérieur. La recherche a été réalisée en trois étapes , la première dédiée à la caractérisation de la relation des étudiants avec des ordinateurs; l'identification des moyens par lesquels ces dispositifs aident pour résoudre les problèmes de Design; et la recherche sur les représentations sociales (RS) des étudiants sur les Principes de l'Informatique. 86 étudiants ont été interrogés , entre eux 23 sont programmeurs. Les analyses prototypiques et de contenu des données ont indiqués la prévalence des RS et pratiques instrumentales impliquant l'Informatique, objectivée et ancrée par l'ordinateur comme un moyen que la matérialise et que la produite. La deuxième étape a été basée sur ces résultats pour élaborer une approche pédagogique fondée sur le langage RocketSocket , dont la principale caractéristique est de considérer les différents profils des apprentis des Principes de l'Informatique. Le langage explore la narrative d'une fusée qui peut être programmée pour collecter des étoiles et des astéroïdes dans l'espace comme une stratégie qui vise à discuter des concepts , pratiques et perspectives du CT. La troisième étape a été basée sur l'étude du cas prototypique qui a connu le potentiel du langage RocketSocket avec deux étudiants de Design, un programmeur et l'autre non. Les données ont été analysés selon la perspective des microgèneses cognitives, en observant comment les connaissances des participants sur l'informatique et sur les ordinateurs ont été sélectionnés et évalués selon leur pertinence fonctionnelle lors de la construction d'algorithmes qui contrôlent la fusée. La dernière partie de la thèse traite de deux études d'une manière intégrée au sein de la triangulation théorique initiale, en cherchant à réfléchir sur le processus de conception du langage à partir des RS des étudiants et leur possibilités de contribution à la popularisation du CT.

Mots-clés: Pensée Computationnelle, représentations sociales, microgènese, apprentissage, Design Computationnelle.

Sant'Anna, H. C. (2014) Action, Computing, Representation: A psychogenetic investigation about the development of Computational Thinking. Doctoral Thesis. Post-Graduation Program in Psychology, Federal University of Espírito Santo, Vitória, Brazil, 199 pp.

Computational Thinking (CT) is defined as a set of skills and competencies usually ascribed to Computer Science professionals, students and researchers, but in the recent years has become an accessible knowledge to more people through initiatives that seek to popularize it. This thesis was based on a triangulation between the Social Representations Theory in the structural approach, the Sociohistorical Psychology and the functional perspective of Genetic Epistemology, aiming to elaborate a constructionist pedagogic approach for the development of CT among undergraduate Design students. The research was conducted in three stages, the first dedicated to the characterization of the relation between the students and the computers; the mapping of the ways in which those devices help in Design problem solving; and investigated the social representations of the students about Computing Principles. A total of 86 students participated in the study, 23 of these programmers. Prototypical and content analysis indicated the prevalence of instrumental representations and practices involving the Computing, objectified and anchored by the computer as a means which produces and makes it concrete. The second stage of the research departed from those results to develop a pedagogical approach based on the RocketSocket programming language, whose main feature is to consider different profiles of learners of the CT principles. The language explores a narrative of a rocket which can be programmed to collect stars and avoid asteroids in space as a strategy to discuss concepts, practices and perspectives of the CT. The third stage consisted of a prototypical case study that experienced the potential of the RocketSocket language with two undergraduate Design students, a programmer and a non-programmer. Data were analyzed from the cognitive microgenetic perspective, observing how the participants' knowledge about Computing and computers were selected and evaluated for their functional relevance during the construction of the algorithms that controlled the rocket. The last part of the thesis discusses the two studies in a integrated view within the initial theoretical triangulation, aiming to reflect on the process of designing the language inspired by the students' social representations and its potential contribution to the popularization of CT.

Keywords: Computational Thinking, social representations, microgenesis, learning, Computational Design.

Um dos pensadores que influenciou a realização desta tese afirmou certa vez que a "revolução da Computação ainda não aconteceu" (Kay, 2007). Pode parecer uma frase dramática, mas após 14 anos envolvido com a educação tecnológica, passei a observar um fenômeno curioso de ampliação expressiva no acesso aos computadores acompanhado de uma diminuição não menos relevante do conhecimento sobre esses dispositivos. Quando tive acesso aos computadores pela primeira vez, por volta dos oito anos de idade em meados dos anos 1980, aqueles eram equipamentos para especialistas: caros, difíceis de encontrar, operar e manter. Nas rodas de conversas dos amigos dos meus pais, videogames eram coisas de crianças, computadores eram coisas de adultos.

Ainda assim, comecei cedo a utilizar os computadores de uma forma que hoje quero acreditar que foi a ideal. Fui o feliz proprietário de um microcomputador pessoal MSX[2], projetado especificamente para crianças e equipado com jogos infantis (como era de se esperar), mas também com uma linguagem de programação orientada a programadores iniciantes. Aprender a programar significava, para mim, poder levar meus interesses para dentro daquela máquina: gostava de desenhar no papel e passei a desenhar no computador; aprendi a tocar instrumentos musicais e também experimentei música no computador; me divertia com jogos e videogames e passei a sonhar com a possibilidade de ganhar a vida desenvolvendo meus próprios jogos.

Ao longo da adolescência, acompanhei a popularização dos computadores entre meus colegas de escola, apesar dos amigos programadores terem sido raros. Muitas vezes conversei sobre programação com professores, irmãos mais velhos e pais de colegas de turma, por falta de interlocutores da minha idade. Meu interesse pela programação de computadores se intensificou nesse período, ampliando meu conhecimento para áreas como animação digital, computação gráfica, música eletrônica e criação de jogos.

Quando finalmente escolhi uma carreira e entrei no curso superior, fui um dos três únicos na minha turma do curso de Design da Universidade Federal do Espírito Santo (UFES) que

[1] A numeração dos capítulos está apresentada em binário, iniciando em zero, opção realizada em função da temática da tese.
[2] O MSX foi um computador pessoal comercializado pela Microsoft no Japão, alguns países da Europa e Brasil entre 1983 e 1990.

sabiam programar. Um dos colegas estava na segunda graduação, sendo que a primeira havia sido em Ciência da Computação. O outro havia cursado o ensino médio em Processamento de Dados na antiga Escola Técnica, atual Instituto Federal do ES. Mais uma vez, eu parecia solitário no interesse pela programação fora dos atividades acadêmicos.

Minha condição de autodidata proporcionou-me um emprego antes mesmo da matrícula na UFES. As habilidades de programador e a experimentação com os processos da máquina desde a pré-adolescência me proporcionaram uma entrada rápida no mercado de trabalho. A maioria dos meus colegas de turma na graduação utilizava o computador profissionalmente e nas tarefas acadêmicas, porém muito distantes da programação. Lembro-me de diversos episódios onde desejei fazer algo que um determinado aplicativo da área de Design não oferecia e acabei construindo minha própria solução. Considerando os prós e contras da participação dos computadores no meu desenvolvimento pessoal, acadêmico e profissional, resolvi perguntar se a minha história poderia ser explicada pelo *simples acesso* precoce aos computadores.

O argumento que defendo ao longo deste trabalho é que se houve precocidade, foi do acesso a um tipo específico de computador *preparado* para me transformar em algo além de um *usuário*. Como mencionei no breve histórico acima, tive colegas de escola que tiveram o mesmo acesso a computadores que eu e que ainda assim permaneceram como *consumidores* de programas criados por outras pessoas. Não é por acaso que os programas de inclusão digital brasileiros, baseados principalmente no aumento da oferta dos dispositivos[3], não parecem ser suficientes para criar uma geração de pessoas que *pensam com o computador*, ao invés de deixarem *o computador pensar por elas*.

A possibilidade de criar oportunidades ao mesmo tipo de acesso aos computadores que eu tive, mesmo que tardias, foram a motivação deste trabalho. Como docente da área de Design, tenho interesse em ampliar as habilidades de projeto dos meus alunos, deslocando os computadores da situação de meras ferramentas para a extensão do seus intelectos, como sugeriu Douglas Engelbart[4] há mais de meio século. Espero que os produtos finais da tese, especialmente a linguagem *RocketSocket*, extrapolem as limitações do estudo e possam contribuir para que pessoas de qualquer perfil, não apenas designers, tenham a chance de descobrir como utilizar a Computação a seu favor.

---

[3] Ver programas e ações em http://www.mc.gov.br/inclusao-digital/acoes-e-programas. Acesso em 20 de setembro de 2013.
[4] Engelbart, D. (1962). *Augmenting Human Intellect: A Conceptual Framework.* SRI Summary Report AFOSR-3223. Disponível em http://dougengelbart.org/pubs/augment-3906.html. Acesso em 23 de setembro de 2013.

Esta tese discute o processo de pesquisa e desenvolvimento de uma abordagem para a aprendizagem de princípios fundamentais da Computação por estudantes de Design[5] do ensino superior. A discussão que se desenvolve nos capítulos que se seguem inscreve-se no universo de pesquisas que visam, por um lado, repensar a utilização dos computadores em diversos contextos de aprendizagem, e por outro, investigar como aprendizes com diferentes histórias de vida e interesses apropriam-se dessas tecnologias na construção das suas próprias estratégias de resolução de problemas em diferentes situações.

A organização deste texto foi estruturada visando oferecer ao leitor uma revisão das teorias e autores que deram suporte às pesquisas realizadas, para em seguida analisar e discutir seus resultados. A introdução se inicia com um breve histórico dos usos dos computadores pessoais por designers, tendo um importante fio condutor: o *Construcionismo*, teoria sobre a construção do conhecimento utilizando computadores, com raízes nos trabalhos de Jean Piaget, Seymour Papert, Jerome Bruner e Alan Kay. Na sequência são apresentados os conceitos de *Computação* e *Pensamento Computacional* e *Design Computacional*, visando discutir habilidades e competências de projeto e resolução de problemas a serem desenvolvidas por qualquer pessoa na interação com os computadores.

Ao final da introdução, propõe-se uma abordagem situada para o Construcionismo fundada na articulação teórica entre a Psicologia Sócio-Histórica de Vygotsky e colaboradores, a Epistemologia Genética do grupo de Jean Piaget e a perspectiva estrutural das representações sociais, desenvolvida por Jean-Claude Abric. Tal articulação busca fornecer subsídios para a investigação das representações sociais elaboradas pelos estudantes sobre a Computação, bem como a relação dessas representações com as práticas de projeto daquele grupo. O segundo capítulo apresenta formalmente os objetivos e justificativas da tese, seguidos da apresentação do método no capítulo três. O quarto capítulo detalha procedimentos, instrumentos e apresenta os resultados da pesquisa sobre as representações sociais da Computação elaboradas pelos estudantes de Design.

O quinto capítulo parte dos resultados dessa pesquisa e propõe uma abordagem pedagógica para os Princípios da Computação de orientação construcionista, cujo produto principal é uma linguagem de programação denominada *RocketSocket*. O sexto capítulo relata e discute um estudo de caso da linguagem com estudantes de Design, para finalmente chegar à integração teórica que encerra a tese no capítulo sete.

[5] Ao longo desta tese, a grafia *Design* corresponde às áreas e subáreas do conhecimento envolvidas (Design Computacional, Design de Interfaces), enquanto *design* refere-se ao ato de projetar (o design de uma interface ou de um programa).

1.1 – Computação Pessoal, Construcionismo e Design

Mesmo antes da popularização dos computadores pessoais nos anos 1980[6], Seymour Papert (Papert e Solomon, 1971; Papert 1971) e Alan Kay (Kay, 1972) investigaram aplicações potenciais dos computadores no desenvolvimento cognitivo de crianças. Esses autores e seus colaboradores vislumbraram, há pouco mais de 40 anos, que o aumento na disponibilidade dessas tecnologias tanto nas residências quanto em ambientes de trabalho ou espaços de educação formal ofereceria uma série de novas oportunidades para que as pessoas guiassem seus interesses e desenvolvessem seus próprios projetos. Moggridge (2007) reforça tais oportunidades ao sugerir que um computador pode ser considerado a partir de seis paradigmas: como *inteligência autônoma* que auxilia seus usuários em tarefas; como *ferramenta* para desenvolvermos nossos trabalhos; como *meio de expressão*; como *sistema vivo* ou de *simulação de vidas*; como *veículo* para transitarmos nas infraestruturas computacionais e informacionais que nos cercam; ou como parte da dinâmica do *sistema da moda*, marcada por relações grupais e identitárias mediadas pela propriedade e fluência em determinados códigos ou tecnologias.

O paradigma do computador como *meio de expressão* contempla, principalmente, as formas por meio das quais os usuários se apropriam dos recursos do sistema para se expressarem: sistemas operacionais[7], linguagens de programação, pacotes de aplicativos, dispositivos de entrada e saída de dados, conectividade e comunicação, entre outros. A maior parte dessas formas de interação foram criadas e experimentadas entre os anos 1950 e 1980, seguindo duas principais vias (p.77): a *low-road* ou interação baseada em texto (Fig. 1.1); e a *high-road* ou interação baseada em telas e ponteiros (Fig. 1.2).

[6] Para uma visão detalhada da popularização dos computadores pessoais ver Ceruzzi (2003), capítulos 7 e 8.
[7] Um glossário para termos técnicos utilizados neste trabalho encontra-se disponível no Anexo I.

Figura 1.1 – Exemplo da *low-road*: Teletipo modelo 33 (1963), utilizado para a interação com computadores de tempo compartilhado. As interações eram baseadas no envio de instruções ao computador central pelo teclado do terminal e recebimento assíncrono de respostas pela impressora. Esse modelo de interação ainda existe nos consoles das variantes do Linux ou no interpretador de linhas de comando (*prompt*) do Microsoft Windows.

Figura 1.2 – Exemplo da *high-road*: Sketchpad, desenvolvido por Ivan Sutherland em 1963, utilizava uma caneta ótica e botões especiais para a entrada de dados e as respostas síncronas do computador eram visualizadas na tela. Sistemas operacionais gráficos contemporâneos, baseados em janelas, ícones e uso de mouse como o Microsoft Windows 7 e Apple MacOS X são descendentes da mesma linhagem.

Ainda nos anos 1950 alguns profissionais criativos associaram-se a cientistas da computação em centros de pesquisa para investigar o potencial expressivo dos computadores. No entanto, conforme argumenta Manovitch (2013), essas iniciativas geravam programas restritos aos interesses desses artistas e que eram executados em tipos distintos de computadores. Foi apenas a partir dos anos 1970, com os esforços realizados pela equipe do Xerox Palo Alto Research Center (PARC), que o computador pessoal ganhou tanto sua aparência quanto potencial contemporâneos: interface gráfica com janelas sobrepostas e ícones, conectividade em rede, programas para manipulação de diversas mídias – textos, imagens, sons e animações – e uma linguagem de programação simples (Smalltalk, Fig. 1.3) que permitia ao usuário criar seus próprios programas.

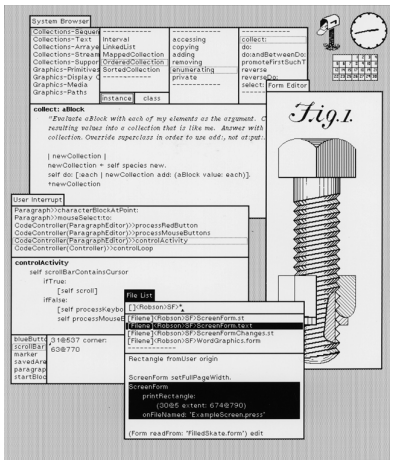


Figura 1.3 – Interface gráfica janelas sobrepostas e ícones desenvolvidos pelo PARC (1974).

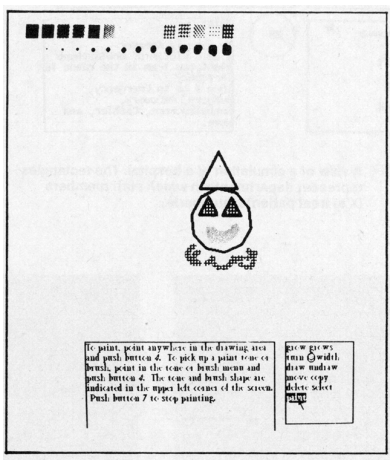


Figura 1.4 – Programa de desenho desenvolvido por uma criança em Smalltalk (1977).

A visão de Alan Kay, líder do grupo do Xerox PARC responsável por essa concepção, era de que o computador pessoal deveria ser um *metameio* (Kay e Goldberg, 1977, p. 31): "*o computador, enquanto meio, poderia ser qualquer outro meio, desde que os métodos de visualização e incorporação apropriados fossem oferecidos suficientemente*". Para Kay, a simulação seria a noção central da sua visão para a computação pessoal.

Uma das fontes de inspiração formalmente citadas por Kay e Goldberg (p. 32) para a sua proposta do computador como metameio foram as pesquisas de Seymour Papert com crianças programando a linguagem LOGO que deram origem à abordagem pedagógica denominada Construcionismo (Harel e Papert, 1991). Essa abordagem compartilha da conotação construtivista da aprendizagem como construção de estruturas de conhecimento (Piaget, 2003) independentemente das circunstâncias onde o aprendizado ocorre, somando a ideia de que isso aconteceria de forma especialmente positiva em contextos onde o aprendiz estaria conscientemente engajado na construção de objetos. Em outros termos, a disponibilidade do computador-metameio permitiria que o aprendiz perseguisse seus próprios objetivos, construindo o conhecimento enquanto interage com a máquina ao solucionar problemas ou durante a realização de atividades do seu interesse.

Essa perspectiva, que pensa o computador como meio de expressão em uma *cultura da computação*, se opõe radicalmente às tentativas de integração dos computadores no processo de aprendizagem pela via dos laboratórios de informática. Papert (1994) argumenta que a oferta de conhecimentos práticos mínimos sobre a computação nesses laboratórios teria o mesmo efeito que a oferta de conhecimentos mínimos sobre leitura, escrita e literatura no ensino de línguas – a formação de *analfabetos funcionais*. Kay (1989) complementa esse argumento ao defender que 1) a habilidade de *leitura* em um meio significaria que alguém consegue *acessar* materiais e ferramentas produzidas por outras pessoas; e que 2) a habilidade de *escrita* em um meio significaria que alguém consegue *criar* materiais e ferramentas para outras pessoas. Papert e Kay, cada um a seu modo e com objetivos distintos, estavam diretamente interessados em oferecer um contexto fértil e as ferramentas apropriadas para que pessoas comuns, e não apenas especialistas, pudessem se expressar livremente utilizando os computadores.

No caso dos estudantes de Design do ensino superior, grupo de interesse deste trabalho, a ideia do computador-metameio para a expressão pessoal e realização de projetos é fundamental. Desde o lançamento do primeiro modelo do Macintosh em 1984 pela Apple, os computadores mudaram radicalmente a prática profissional no Design (Meggs, 1997). Num primeiro momento, estudantes e profissionais acostumados a processos artesanais e

semiartesanais de preparação de modelos para reprodução industrial adotaram as simulações computacionais de páginas impressas, objetos tridimensionais, ilustrações, animações e partituras para facilitar o desenvolvimento de projetos. Nesse período, designers eram em geral consumidores dos programas disponíveis para os computadores pessoais. Ao contrário do que defendiam Alan Kay e Seymour Papert, as linguagens e ambientes de programação desse período são pouco amigáveis e de difícil aprendizado[8], cabendo aos programadores e especialistas da área de informática o papel de construir as ferramentas que seriam utilizadas pelos demais usuários, incluindo os designers.

Figura 1.5 – MacPaint, programa de desenho *bitmap* (1984)

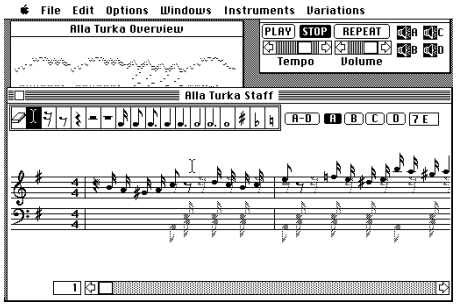


Figura 1.6 – MusicWorks, programa de edição partituras musicais (1984)

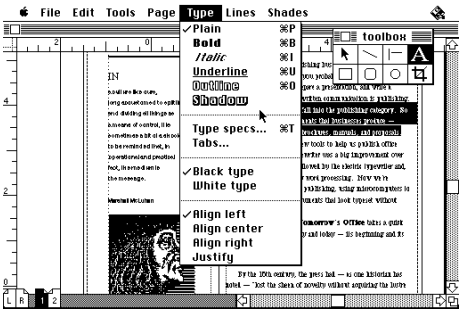


Figura 1.7 – Aldus Pagemaker, utilizado na edição de projetos gráficos (1987).

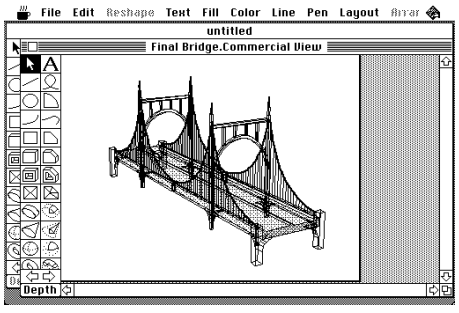


Figura 1.8 – Mac3D, utilizado na modelagem de objetos 3D (1984).

De meados dos anos 1980 ao fim dos anos 1990, aplicativos foram especialmente desenvolvidos com o intuito de empoderar designers e artistas para a escrita com os computadores, dentre os quais destacam-se: os sistemas de autoria multimídia Director[9] (Canter, 1986) e Flash; HyperCard, um ambiente de programação lançado em 1987 pela Apple para "pessoas comuns" (Winograd, 1996); a linhagem de linguagens de programação visual para multimídia e música eletrônica iniciada com o Patcher de Miller Puckett (IRCAM, 2009); e o ambiente Design by Numbers (Maeda, 2001).

[8] Nos primeiros anos o desenvolvimento de programas para os computadores pessoais era baseado em linguagens de máquina como Assembly ou de baixo nível como C e Pascal. Essas linguagens demandavam mais conhecimento técnico sobre a operação da máquina que Smalltalk ou Lisp/LOGO, defendidas por Alan Kay e Seymour Papert respectivamente.

[9] Flash e Director foram originalmente comercializados pela empresa norte-americana Macromedia, adquirida pela Adobe Systems em 2005.

Ao longo da década de 2000, linguagens de programação como Processing[10] (Reas e Fry, 2007), DrawBot[11] e Context Free[12] foram lançadas tendo artistas e designers como público-alvo. Essas linguagens têm em comum a facilidade de *download* e instalação, guias de primeiros passos para iniciantes, diversas galerias de exemplos na Web e comunidades ativas compostas por usuários que se ajudam mutuamente[13].

Se por um lado a lista de opções para designers parece ter crescido tanto em quantidade quanto qualidade nas últimas três décadas, por outro o conhecimento sobre Computação necessário para a escrita dos próprios programas não se resolve apenas com o aumento na oferta de ambientes e linguagens de programação. Mesmo na mais simples das linguagens, há princípios da Ciência da Computação que precisam ser aprendidos e utilizados criativamente pelo designer na realização dos seus projetos. Conforme explica Maeda (2004, p. 113, tradução nossa):

> Usuários de ferramentas são muito mais comuns que construtores de ferramentas. Esse desequilíbrio tem sido tradicionalmente enraizado na vasta diferença entre os níveis de habilidade necessários para utilizar a ferramenta e para fabricá-la: para usar uma ferramenta no computador, você precisa fazer pouco mais de apontar e clicar; para criar uma ferramenta, você precisa compreender a arte misteriosa da programação de computadores.

Essa necessidade do letramento em Computação estava presente nas preocupações centrais de Papert e Kay nos anos 1970 e parece ainda mais relevante nos dias de hoje. Se parece razoável considerar que o número de pessoas nas mais diversas idades, escolaridades e histórias de vida interessadas em aprender a programar atualmente é maior que o dos primeiros anos da computação pessoal, também é importante lembrar que os computadores estão exponencialmente mais inseridos no cotidiano do que há 30 anos. Além das salas de aula e ambientes de trabalho, os computadores estão presentes nos serviços públicos, no sistema financeiro, nos momentos de diversão, lazer, compras e integram as mais diversas tecnologias de informação e comunicação contemporâneas.

Recuperando o problema anterior da alfabetização funcional ser insuficiente para o letramento efetivo em uma língua, faz-se necessário então discutir em profundidade quais

[10] Disponível em http://www.processing.org. Acesso em 18 de setembro de 2013.
[11] Disponível em http://www.drawbot.com. Acesso em 18 de setembro de 2013.
[12] Disponível em http://www.contextfreeart.org/. Acesso em 18 de setembro de 2013.
[13] Ver Sant'Anna et al (2012a) para uma comparação das principais linguagens disponíveis para iniciantes.

seriam os conhecimentos essenciais da Computação para que não-especialistas consigam expressar-se plenamente neste meio, tanto pela leitura quanto pela escrita.

Saber *ler* e *escrever* computacionalmente para poder se expressar em um mundo onde os computadores são onipresentes parece deslocar a questão da educação formal para a cidadania. Segundo Jeannette Wing (2006), as habilidades e competências tradicionalmente atribuídas a profissionais, estudantes ou pesquisadores da Ciência da Computação deveriam deixar de ser um saber do especialista para se tornarem conhecimentos disponíveis para qualquer um. Tal conjunto de habilidades, denominado pela autora como Pensamento Computacional (*Computational Thinking*, CT), não seria apenas de um saber técnico, mas uma necessidade de aprendizagem elementar contemporânea tal qual a leitura, escrita e aritmética. Para que seja possível compreender a proposta do CT, faz-se necessário revisar autores e trabalhos que dissertam sobre o conhecimento próprio da área da Computação e como este se relaciona com habilidades e competências de resolução problemas, em geral, e no Design, em particular.

1.2 – Computação, Pensamento Computacional e Design Computacional

Segundo Diverio e Menezes (2011) a Ciência da Computação é o conhecimento sistematizado da *Computação*, tendo se desenvolvido em diversos momentos e regiões ao longo da história da humanidade. A ênfase teórica desse conhecimento consiste em fundamentos e modelos computacionais, independendo de instrumentos ou máquinas de Computação e suas tecnologias – nos dias de hoje, representadas principalmente pelos computadores, *software* (programas) e *hardware* (a parte física dos dispositivos).

Já a ênfase prática da Ciência da Computação consiste na aplicação da teoria no projeto de sistemas computacionais, ou seja, na implementação de funções que podem computar. Os autores afirmam que algumas questões seriam recorrentes na Computação, tais como o que seria uma solução computável, quais seriam os limites do que pode ser computado e se existiriam problemas sem solução computacional.

> O objetivo do estudo da *computabilidade* é determinar a *solucionabilidade* de problemas, ou seja, investigar a existência ou não de algoritmos[14] que solucionem determinada classe de problemas. Mais ainda, investigar os limites da

[14] Um algoritmo é a descrição da solução de um problema, finita e não ambígua, consistindo de passos discretos, executáveis mecanicamente em um tempo finito (Diverio e Menezes, 2011). Futschek (2006) define um algoritmo em termos gerais como um método de solucionar um problema que consiste na definição de instruções exatas.

> computabilidade e, consequentemente, os limites do que pode efetivamente ser implementado em um computador (Diverio e Menezes, 2011, p.246, ênfases nossas).

Denning et al (1988) apresentaram os três principais paradigmas da disciplina de Computação em um relatório que foi aprovado e endossado pela *Association for Computing Machinery* (ACM)[15] como recomendação para os currículos básicos sobre o tema:

- *Teoria*, com raízes na Matemática, consiste em quatro passos seguidos no desenvolvimento de uma teoria válida e coerente: 1) caracterizar os objetos de estudo (definição); 2) hipotetizar sobre as possíveis relações entre eles (teorema); 3) determinar se as relações são verdadeiras (prova); 4) interpretar os resultados.
- *Abstração (modelagem ou experimentação)*, com raízes no método experimental científico, consiste em quatro etapas a serem seguidas na investigação de um fenômeno: 1) formulação de hipóteses; 2) construção de um modelo e realização de predições; 3) projeto de um experimento e coleta de dados; 4) análise dos resultados.
- *Projeto*[16], com raízes na Engenharia, compreende quatro etapas na construção de um sistema ou dispositivo a partir de um problema dado: 1) requisitos da situação; 2) especificações da situação; 3) projeto e implementação do sistema; 4) testes do sistema.

Considerando as raízes de cada um dos paradigmas, os autores explicam que matemáticos, cientistas e engenheiros realizam iterações em seus respectivos processos em busca de erros e inconsistências, validações dos modelos e verificação de hipóteses, satisfação dos requisitos do sistema. A Computação combinaria os três paradigmas de tal forma que seria difícil atribuir a primazia de um sobre os demais, apesar manterem suas especificidades e de representarem competências distintas:

> A teoria está preocupada com a habilidade de descrever e provar relações entre objetos. A abstração está relacionada à habilidade de usar essas relações para fazer predições que possam ser comparadas com o mundo. O projeto está interessado na

[15] A ACM é a principal organização mundial de profissionais da Computação. Fundada em 1947, realiza ações visando o avanço da Computação como ciência e como profissão. Disponível em http://www.acm.org/about/about?pageIndex=4. Acesso em 20 de setembro de 2013.

[16] O texto original emprega o termo *design*, que na língua inglesa indica projeto e planejamento, mas não desenho. Essa acepção é coerente com o sentido de projeto em português para área de Design, Engenharia e Arquitetura, denotando planejamento prévio de ações antes da execução propriamente dita.

> habilidade de implementar instâncias específicas dessas relações e utilizá-las para realizar ações úteis (Denning et al, 1998, p.11, tradução nossa).

Um segundo relatório mais recente endossado pela ACM (Shackelford et al, 2006) define a Computação como "qualquer atividade orientada a objetivos que exige, beneficia-se de, ou cria computadores" e cita como exemplos de atividades: processar, estruturar e gerenciar vários tipos de informação; realizar estudos científicos usando computadores; fazer sistemas computacionais se comportarem de forma inteligente; criar e usar formas de comunicação e mídias de entretenimento; encontrar e recuperar informações relevantes para qualquer propósito particular.

A partir das recomendações dos dois relatórios, pode-se sugerir que a formação acadêmica específica dos cientistas da Computação seria orientada ao desenvolvimento de determinadas habilidades e competências de bases matemática, científica e da engenharia, mas com aplicações em diversas áreas da atividade humana. É na direção da popularização dessas habilidades e competências que surgiu a proposta do Pensamento Computacional (CT), mencionado brevemente no fim da seção anterior. O principal obstáculo a ser transposto pelo CT consistiria no planejamento de oportunidades de aprendizado dos paradigmas e conteúdos temáticos da Computação, para em seguida colocá-los em prática na resolução de problemas nas demais diversas áreas.

Em um trabalho que atraiu a atenção da comunidade acadêmica, Jeannette Wing[17], professora do Departamento de Ciência da Computação na Universidade Carnegie Mellon (EUA), elencou as características que definiriam o CT como um tipo específico de pensamento. No texto, a autora exemplifica uma série de operações mentais com o intuito de esclarecer que pensar computacionalmente significaria (Wing, 2006):

- Reformular um problema aparentemente difícil em outro que se saiba como resolver, talvez por redução, incorporação, transformação ou simulação;
- Pensar recursivamente;
- Processamento paralelo;
- Interpretar código como dados e dados como código;
- Checagem de tipos como a generalização das dimensões de análise;

[17] Disponível em http://www.cs.cmu.edu/~wing/. Acesso em 15 de setembro de 2013.

- Reconhecer as virtudes e os perigos da nomeação, ou de dar a alguém ou a alguma coisa mais de um nome;
- Reconhecer tanto o custo quanto o poder de endereçamento indireto e das chamadas de procedimentos;
- Julgar um programa não apenas por sua correção e eficiência, mas pela estética, e o projeto de um sistema pela simplicidade e elegância;
- Usar a abstração e decomposição ao realizar uma tarefa grande e complexa ou ao projetar um sistema grande e complexo. É separação de preocupações;
- Escolher uma representação apropriada para um problema ou a modelagem de aspectos relevantes de um problema para torná-lo tratável;
- Utilizar invariantes para descrever o comportamento de um sistema sucintamente e declarativamente;
- Ter a confiança de que se pode, com segurança, usar, modificar e influenciar um grande sistema complexo sem entender cada um dos seus detalhes;
- Pensar em termos de prevenção, proteção, e recuperação em cenários de piores possibilidades por meio de redundância, contenção de danos e correção de erros;
- Usar raciocínio heurístico para descobrir uma solução.

Tais operações estariam baseadas na ideia de que a Ciência da Computação é o estudo da Computação – o que pode ser computado e como computá-lo – de maneira que o CT teria como características:

- Conceituação ao invés de programação – pensar como um cientista da Computação significa mais do que saber programar um computador; exige pensar em múltiplos níveis de abstração;
- Desenvolvimento de habilidades fundamentais, ao invés das mecânicas;
- Uma forma por meio da qual humanos, e não máquinas, pensam;
- Complementa e combina os pensamentos matemático, científico e da engenharia;
- Concentra-se em ideias e não artefatos, de forma que o interesse não é apenas o desenvolvimento de *software* e *hardware*, mas dos conceitos e abordagens

computacionais em uso para que as pessoas solucionem problemas, interajam, comuniquem-se e administrem suas vidas;

- Deve ser acessível a qualquer um em qualquer lugar, tornando-se realidade no dia em que estiver tão integrado à vida das pessoas a ponto de desaparecer como filosofia explícita.

Em síntese, Wing (2006, p.33) sugeriu que o CT "envolve resolução de problemas, projeto de sistemas e compreensão do comportamento humano, inspirando-se em conceitos fundamentais da Ciência da Computação". Bundy (2007) concorda com essa visão ao argumentar que os conceitos computacionais ofereceriam uma nova linguagem para descrever hipóteses e teorias, enquanto os computadores ofereceriam uma extensão para as faculdades cognitivas dos seres humanos.

Levando em consideração a importância dada pelos autores mencionados aos princípios e conceitos fundamentais da Computação para o desenvolvimento do CT, faz-se necessário abordar tais ideias em detalhe, bem como seus desdobramentos sobre a natureza dos problemas propriamente computacionais.

Princípios e conceitos da Computação

Denning (2003) argumenta que os princípios de uma área seriam um conjunto de histórias entrelaçadas sobre a estrutura e comportamento dos elementos daquela área, organizadas de forma a simplificar a explicação da origem, evolução e demais questões relativas ao funcionamento daqueles princípios. No caso da organização realizada por esse autor acerca dos "*Grandes Princípios da Computação*", a seleção foi realizada seguindo três critérios (Denning e Martell, 2007a):

I. *Universalidade:* os princípios relacionados emergem a partir de preocupações pervasivas, afetando a todos, sendo inevitáveis, duráveis ou permanentes;

II. *Recorrência*: os princípios são encontrados repetidamente em vários contextos, descobertos independentemente por diferentes grupos, são reprodutíveis e úteis para predições e projetos;

III. *Amplitude da influência*: os princípios formam e restringem todas as tecnologias e aplicações da Computação, moldando as práticas comuns e impactando ampla e profundamente a ciência, a indústria e a sociedade.

Tomando como base os critérios de seleção citados, Denning e Martell (2007b) criaram sete categorias de princípios conforme apresentado na Tabela 1.1. Essas categorias foram estruturadas como janelas para um único espaço de conhecimentos sobre a Computação, de modo que cada janela representaria uma visão singular sobre aquele espaço ao mesmo tempo em que uma mesma questão ou problema a ser resolvido poderia ser visto a partir de mais de uma janela (Fig. 1.9).

Tabela 1.1 – Categorias dos Grandes Princípios da Computação, adaptado de Denning e Martell (2007b)

Categorias	Questão Central
Computação	Quais processos, naturais e artificiais, são computacionais; o que eles podem ou não podem fazer; como lidam com as complexidades computacionais inerente e pervasiva.
Comunicação	Transmissão de dados com recepção confiável.
Coordenação	Como entidades autônomas trabalham em conjunto em direção a um resultado comum.
Armazenagem	Como computações armazenam e recuperam informação e como o arranjo dos dados nos sistemas de armazenamento afetam a performance.
Automação	Encontrar formas computacionalmente eficientes para desempenhar tarefas humanas.
Avaliação	Como sistemas que computam se comportam sob diversas cargas computacionais e quanta capacidade eles precisam para entregar os resultados no tempo adequado.
Projeto	Como projetar programas e sistemas que são seguros, confiáveis e utilizáveis.

Os benefícios resultantes da organização desse quadro teórico incluiriam revelar a estrutura profunda da Computação e o porquê dela permear tantas outras áreas; revelar princípios comuns entre tecnologias, possibilitando simplificações, novas descobertas e inovações; oferecer uma linguagem comum para a discussão da Computação com outras áreas; inspirar novas abordagens para o ensino e aprendizado da Computação; e inspirar o interesse dos jovens pela Computação (Denning, 2007).

Em outra direção, Wing (2008a) cita a abstração e a automação (os "As" do CT) como as essências da Computação. A autora divide a problemática da abstração em dois níveis. No primeiro, argumenta que diferentemente das abstrações da matemática ou da física, a abstração da Computação seria simbólica, extrapolando os limites do espaço e do tempo. Um algoritmo, por exemplo, seria a abstração de um procedimento passo a passo onde uma entrada seria transformada em uma saída desejada. Já as linguagens de programação seriam abstrações de sequências de caracteres que, quando devidamente interpretadas, realizariam uma computação. Por fim, Wing sugere que trabalhar com as restrições e

limitações do mundo real também seriam abstrações, uma vez que seria necessário considerar falhas e situações limite.

Figura 1.9 – As sete janelas das categorias dos Princípios da Computação. Adaptado de Denning (2007).

No segundo nível da análise, a autora sugere que o processo de abstração na Computação introduz o trabalho em camadas, sempre existindo pelo menos duas: uma de interesse e outra acima ou abaixo, correspondendo aos processos computacionais subjacentes – aqueles dos quais a camada de interesse depende – ou sobrejacentes – aqueles que resultarão das ações realizadas na camada de interesse[18].

Em relação à automação, Wing esclarece que as abstrações, como ferramentas mentais, podem ser automatizadas pelos computadores. A automação, como segundo princípio essencial da computação, consistiria na mecanização das abstrações, das camadas de abstração e da relação entre elas. Em outros termos, se o algoritmo é a abstração para a resolução de um problema, a construção e execução daquele algoritmo em um computador é a automação.

Uma terceira conceituação dos princípios do CT, interessante para esta tese em decorrência da orientação aos iniciantes, principalmente crianças, foi proposta por Brennan, Chung e Hawson (2011) como parte dos esforços para construir um currículo mínimo para o aprendizado da linguagem Scratch[19]. Ao invés de definir os princípios do CT nos termos de áreas de conhecimento (como as janelas de Denning) ou de essências (os "As" de Wing), o grupo sugeriu que haveria um conjunto de *conceitos*, *práticas* e *perspectivas* da Computação que poderiam ser apresentadas ao aprendiz:

[18] A Internet é um bom exemplo da abstração computacional em camadas. Considerando a rede como camada de interesse, os protocolos e estruturas de transmissão de dados são processos subjacentes em operação numa camada inferior. Ao mesmo tempo, a rede possui diversas camadas superiores onde aplicações são construídas, como navegadores, programas de e-mails, trocas de arquivos e mensagens instantâneas.

[19] A linguagem Scratch foi construída sob supervisão de Mitchel Resnick, ex-orientando de Seymour Papert no Instituto de Tecnologia de Massachusetts (MIT), compartilhando dos mesmos princípios e objetivos da linguagem LOGO. Disponível em http://scratch.mit.edu. Acesso em 15 de setembro de 2013.

Tabela 1.2 – Conceitos, práticas e perspectivas do Pensamento Computacional	
Conceitos	**Descrição**
Sequência	Identificar uma série de passos para uma tarefa
Laços	Executar a mesma sequência múltiplas vezes
Paralelismo	Fazer coisas acontecerem ao mesmo tempo
Condicionais	Tomar decisões baseadas em condições
Operadores	Suporte a operações matemáticas e expressões lógicas
Dados	Armazenar, recuperar e atualizar valores
Práticas	**Descrição**
Ser iterativo e incremental	Desenvolver uma parte, testá-la e depois desenvolver mais
Testar e depurar	Certificar-se que as coisas funcionam, encontrar e corrigir problemas
Reusar e remixar	Construir algo a partir de algo feito por outros ou por você
Abstrair e modularizar	Construir algo grande pela combinação de partes menores
Perspectivas	**Descrição**
Expressão	Compreender que a computação é meio de criação
Conexão	Reconhecer o poder de criar com e para outras pessoas
Questionamento	Sentir-se empoderado para fazer perguntas sobre o mundo

Pode-se observar que os princípios das abordagens de Denning e Wing estão diluídas nos conceitos, práticas e perspectivas da Tabela 2. Além da simplificação descritiva, a proposta do grupo que desenvolveu a linguagem Scratch possui um caráter mais funcional que epistêmico, centrando-se nos procedimentos e representações do aprendiz sobre os problemas a serem resolvidos e menos no aprendizado de conhecimentos especializados da Ciência da Computação. Essa distinção foi especialmente importante para a construção da abordagem pedagógica apresentada no capítulo 5.

Pensamento Computacional e Resolução de Problemas

Os conhecimentos e habilidades relacionados às categorias de Princípios da Computação (Denning), as competências de abstração e automação (Wing) ou os conceitos, práticas e perspectivas do CT (Brennan, Chung e Hawson) estão relacionados ao uso de estratégias computacionais para a resolução de problemas. Conforme defende Wing (2011), proporcionar oportunidades de desenvolvimento do CT a todos significa criar condições para que as pessoas: 1) compreendam quais aspectos de um determinado problema são tratáveis pela computação; 2) avaliem a correspondência entre ferramentas e técnicas computacionais e um problema; 3) compreendam as limitações e as capacidades das ferramentas e técnicas computacionais; 4) apliquem ou adaptem ferramentas ou técnicas computacionais a novos usos; 5) reconheçam oportunidades de usar a computação de

novas formas; 6) apliquem estratégias computacionais estilo dividir e conquistar[20] a qualquer domínio.

Visando a elaboração de estratégias de aprendizado para o CT, Wing (2008b) compilou uma lista de problemas típicos do cotidiano que poderiam se beneficiar das ferramentas e técnicas computacionais, além de um conjunto de temáticas da Ciência da Computação que estariam relacionadas àqueles problemas: recursão, abstração e representação de dados, composição e decomposição, classificação e busca, intratabilidade, indecidibilidade, código como dados e dados como código, correção, cache, enfileiramento, concorrência e computação distribuída.

Barr e Stephenson (2011) descrevem o CT como um processo de resolução de problemas que inclui algumas características, embora não seja limitado por elas:

- Formulação de problemas de uma maneira que nos permita utilizar um computador e outras ferramentas para resolvê-los;
- Organização lógica e análise de dados;
- Representação de dados por meio de abstrações como modelos e simulações;
- Automatização de soluções por meio de *pensamento algorítmico* (uma série de passos ordenados);
- Identificação, análise e implementação de soluções possíveis com o objetivo de atingir a combinação mais eficiente e eficaz de passos e recursos;
- Generalização e transferência desse processo de resolução de problemas para uma variedade ampla de problemas.

Dentre as características citadas pelos autores, o conceito de *pensamento algorítmico* merece um detalhamento mais aprofundado, considerando sua relação direta com a prática daqueles que pretendem resolver problemas utilizando a Computação. Para Futschek (2006), o pensamento algorítmico consistiria em habilidades de construção e compreensão de algoritmos capazes de solucionar um determinado problema, tendo em vista: a análise e especificação precisas do problema; a busca pelas ações básicas adequadas ao problema; a construção um algoritmo correto a partir do problema dado e das ações básicas; o pensamento acerca de todos os casos especiais e normais de um problema; e a melhoria na eficiência do próprio algoritmo.

[20] Essa estratégia, muito comum em algoritmos, corresponde à técnica de dividir um problema em subproblemas menores sucessivamente até que se atinja um estágio onde uma solução direta seja possível.

Recuperando a ideia do algoritmo como uma sequência finita de ações em direção à resolução de um problema, CT e pensamento algorítmico entrelaçam-se na resolução de problemas de natureza computacional: os princípios da Ciência da Computação são aplicados na construção de uma sequência finita de ações que podem ser executadas por um computador na resolução de um problema dado.

Essa interdependência entre os dois tipos de pensamento é sustentada pelas recomendações do *National Research Council* (1999) ao sugerir que o pensamento algorítmico seria a chave para o entendimento de muitos aspectos das tecnologias da informação e para a aplicação delas em situações relevantes para os indivíduos. No cotidiano das pessoas interagindo com as tecnologias, o CT opera por meio do pensamento algorítmico. É por meio da construção e entendimento de algoritmos que as pessoas se relacionam com os conceitos da Computação.

Apesar da definição de algoritmo como uma sequência de procedimentos criar condições para o entendimento do termo como uma *receita*, *técnica*, *método* ou *rotina* para a resolução de problemas, Knuth (1997) ressalta que a conotação da palavra seria ligeiramente diferente, com base em cinco características importantes: 1) *finitude*: o algoritmo sempre termina após um número finito de passos; 2) *distintividade*: cada passo de um algoritmo deve ser precisamente definido e as ações a serem realizadas devem ser especificadas de forma rigorosa e não ambígua para cada caso; 3) *entradas*: um algoritmo possui zero ou mais entradas e tais entradas são fornecidas antes do início do algoritmo ou dinamicamente durante a sua execução; 4) *saídas:* um algoritmo possui uma ou mais saídas e estas relacionam-se às entradas; 5) *eficácia*: espera-se que um algoritmo seja eficaz, no sentido de que suas operações devem ser suficientemente básicas de maneira que possam ser realizadas com exatidão e em um espaço finito de tempo por alguém usando papel e lápis. Em outras palavras, uma série infinita de etapas mal definidas ou um conjunto de procedimentos sem pontos de partida e de chegada claros não seriam algoritmos e provavelmente não teriam eficácia na resolução de um determinado problema.

Amorim (2005) explica que a essência do algoritmo está na comunicação entre dois agentes: um que transforma procedimentos, métodos e regras em um conjunto ordenado de instruções visando um resultado particular (o usuário), e outro que executa tais instruções sob algum nível de interpretação (o computador). Para o autor, os benefícios trazidos pelos algoritmos permitem a separação entre tarefas baseadas no conhecimento e tarefa operacionais, viabilizando a automação e divisão do trabalho. No extremo oposto, o

autor alerta que a dependência excessiva do pensamento algorítmico pode dificultar a criatividade, capacidade de julgamento e o senso comum, que seriam todos fatores importantes para a resolução de problemas significativos.

Do Pensamento Computacional ao Design Computacional

A dimensão criativa e capacidade de julgamento mencionados por Amorim (2005) são de suma importância para as aplicações dos conhecimentos da Computação nos processos de resolução de problemas do Design. No contexto da educação e atuação profissional dos designers ou outros profissionais de projeto[21], as habilidades e competências do CT são aplicadas em diferentes perspectivas e abordagens de projeto reunidas sob o rótulo de *Design Computacional* (DC), com desdobramentos nas ferramentas e estratégias empregadas pelo projetistas. A revisão a seguir discute duas contribuições da Computação para a atuação dos designers. Uma preocupa-se com as questões construtivas ou estéticas introduzidas pelos recursos computacionais, mais especificamente com os problemas de *síntese* da forma, enquanto a outra se debruça sobre as influências de um pensamento de natureza computacional nas estratégias de *análise* de problemas de Design.

Começando pela síntese, Mitchell (1990, p.27) explica que o Design (computacional) é "a computação da informação da forma que é necessária para guiar a fabricação ou construção de um artefato". Por *computação* Mitchell entende o conjunto de regras e operações lógicas de transformação e combinação de formas que compõem os artefatos utilizando o computador. Diferentemente do desenho auxiliado por computador (CAD), onde o projetista utiliza recursos em programas que simulam as ferramentas de desenho tradicionais, a forma do artefato seria *gerada* por um programa escrito pelo projetista.

Menges (2010) sugere que o DC tiraria vantagem do potencial dos computadores para juntar, organizar e processar informações. Projetar seria um processo de *geração da forma* por meio de procedimentos ou algoritmos com ênfase na definição e manipulação de variáveis e parâmetros. A relação dos designers com a geração da forma por computador foi precoce, iniciando-se nos anos 1960 com a arte algorítmica desenvolvida por pioneiros como Georg Ness, Frieder Nake, Vera Molnar, Michael Noll, Hiroshi Kawano e Manfred Mohr (Lieser, 2010).

[21] As questões desta seção também se aplicam a arquitetos, engenheiros, artistas visuais e outros que se identificam como designers, tendo ou não educação formal na área.

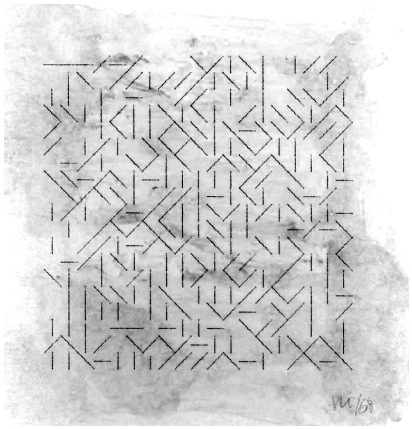


Figura 1.10 – Vera Molnar, Sem título, (1968). Obras como esta exploraram regularidades que emergiam a partir de padrões aleatórios gerados pelo computador. A composição resultante resultou da experimentação direta no código, sem a interferência de instrumentos de desenho ou pintura tradicionais no resultado final. Uma vez programada, a imagem era impressa em *plotters*.

Nas décadas que se seguiram, ao menos três aplicações dos algoritmos nas disciplinas de projeto se destacaram: imagens de síntese, estética generativa e Design Paramétrico. As *imagens de síntese* (Venturelli e Burgos, 1999) consistem em imagens integralmente geradas por procedimentos computacionais, ao invés de seguirem o percurso mais comum de captura de imagens reais por fotografia digital ou *scanners* e posterior tratamento em aplicações de desenho e ilustração digital como o Adobe Photoshop[22]. A *estética generativa* (Bense, 1968) e sua vertente contemporânea denominada *arte generativa* (Galanter, 2003, p.4), corresponderiam a

> [...] qualquer prática onde o artista utiliza sistemas como conjuntos de regras em linguagem natural, programas de computador, máquinas ou outras invenções procedurais colocadas em movimento com algum grau de autonomia contribuindo ou resultando em uma obra artística completa.

A autonomia da obra resultaria da aplicação de fundamentos da inteligência artificial e Computação Natural pelos designers, introduzindo diversas possibilidades à experiência estética. Autores como De Smedt, Lechat e Daelemans (2011) descrevem estratégias inspiradas na natureza para conferir autonomia às obras de arte generativa, tais como elementos que simulam o comportamento de enxames e algoritmos que evoluem em gerações e modificam-se a si mesmos. O *Game of Life* de John Cownay (Gardner, 1970) é um precursor desse tipo de sistema que consegue simular vida por meio de regras de produção simples.

O Design Paramétrico (Hernandez, 2006) ocorre em ambientes onde as investigações de variações de projeto são realizadas sem esforço, geralmente manipulando certas propriedades formais ou construtivas, denominadas *parâmetros*, enquanto outras se mantêm fixas – as *restrições*. Monedero (2000) aponta dois grandes grupos de interesse do

[22] Disponível em http://www.adobe.com. Acesso em 06 de outubro de 2013.

Design Paramétrico, sendo um mais antigo e relacionado à geração estática de variações na forma dos modelos a partir de execuções sucessivas de procedimentos de código. Essa alternativa mantém uma representação interna dos objetos e gera variações a partir dela cada vez que o código é executado. O segundo grupo se utiliza de métodos interativos que permitem a modificação das restrições e parâmetros em tempo real (Fig. 1.12), mesmo após a criação dos objetos, cuja representação interna no sistema seria então modificada ou estendida de forma interativa.

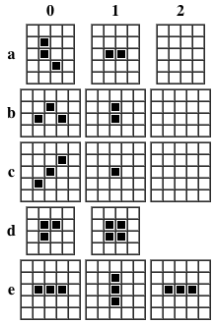


Figura 1.11 – O *Game of Life* possui regras para controlar o comportamento dos autômatos celulares, representados por pontos sobre uma matriz. Quando o programa é iniciado, os pontos são posicionados aleatoriamente na matriz e a cada geração verifica-se: 1) *sobrevivência*: cada ponto com dois ou mais vizinhos sobreviverá para a próxima geração; 2) *morte:* cada ponto com quatro ou mais vizinhos morre e será removido da matriz (*superpopulação*) e cada ponto com um ou nenhum vizinho morrerá por *isolamento*; 4) *nascimentos*: cada célula vazia adjacente a exatamente três vizinhos receberá um ponto na geração seguinte. Na Figura, os autômatos em *a*, *b* e *c* morrem na terceira geração. O autômato em *d* é estável e a configuração em *e* alterna indefinidamente entre os estados 1 e 2.

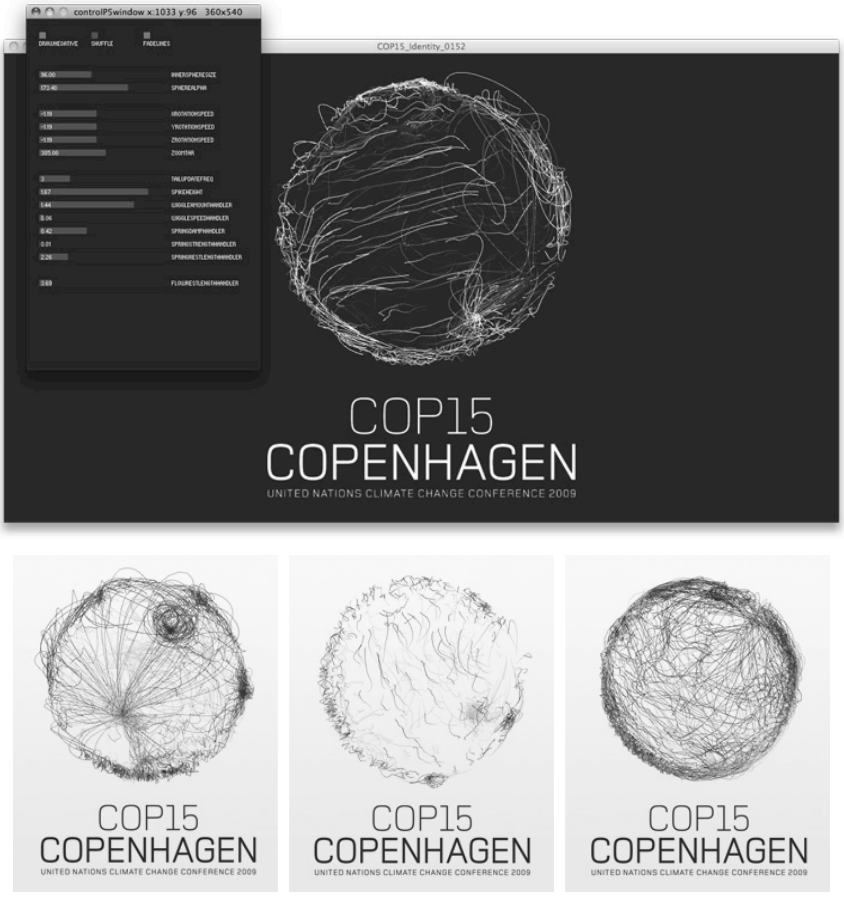


Figura 1.12 – A identidade visual da Conferência das Nações Unidas sobre mudanças climáticas em Copenhagen (COP15 – 2009) é um exemplo de Design Paramétrico. O comportamento visual do símbolo que representa o globo terrestre pode ser modificado em tempo real por meio do ajuste de parâmetros, assim como a velocidade rotação da esfera nos três eixos. Uma versão em vídeo das variações está disponível no endereço https://vimeo.com/8193600.

Quanto às contribuições do Pensamento Computacional para a análise de problemas de Design, Christopher Alexander (1964) desenvolveu um método de projeto influenciado pela *revolução cognitiva*. Pesquisadores da segunda metade do séc. XX envolvidos com as recém-inauguradas Ciências Cognitivas – que combinavam Psicologia, Computação, Cibernética, Matemática, Linguística, Medicina, entre outras – investigaram os processos cognitivos dos seres humanos a partir dos modelos fornecidos pela Teoria Matemática da Comunicação de Claude Shannon (1948), pelas máquinas universais de Alan Turing (1950) e pelos primeiros computadores.

O método de Alexander baseava-se na teoria ingênua dos conjuntos (Halmos, 1960) para descrever um processo cujo objetivo era "projetar formas claramente concebidas que são bem adaptadas a um contexto dado qualquer" (Alexander, 1964, p. 73). O autor argumenta que o processo consciente de Design seria normalmente mediado por representações do designer sobre o problema a ser resolvido, sempre dependente de algum grau de intuição. Sem desconsiderar que este processo seria imaginativo e intuitivo por definição, Alexander defendeu que haveria necessidade de criar abstrações tanto sobre as demandas do contexto de projeto quanto sobre as alternativas formais que se apresentariam como soluções, em decorrência do aumento na complexidade dos problemas.

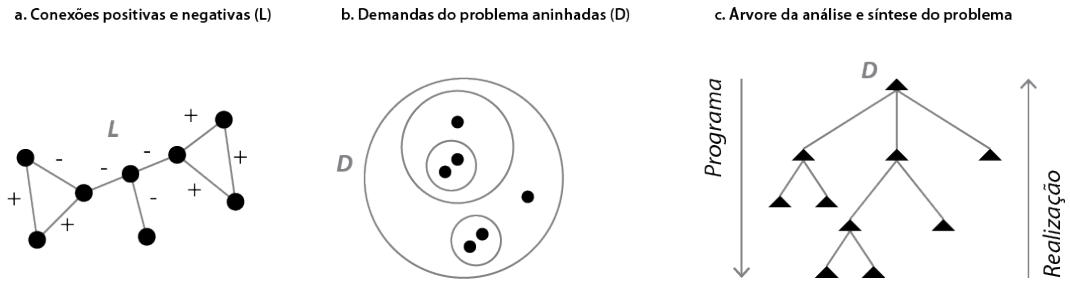


Figura 1.13 – Método de Christopher Alexander (adaptado de Alexander, 1964).

Alexander (p.81) expõe que seu método de análise se desenrola pela criação de um conjunto *D* de demandas do problema, que se apresentam como desajustes potenciais entre a forma e o contexto em questão que devem ser evitados. Os elementos de *D* podem não estar ligados ou relacionar-se de diversas formas, gerando outro conjunto de conexões *L* positivas ou negativas (Fig. 1.13 a). Os dois conjuntos *D* e *L* formariam então um grafo *G(D,L)* que corresponderia à estrutura do problema conforme representada pelo designer. Cada parte do problema a ser resolvida resultaria da decomposição hierárquica, descendente e aninhada de *D* segundo as avaliações em *L* (Fig. 1.13 b), resultando em estruturas em árvore (Fig. 1.13 c) denominadas *programas*. No sentido ascendente, o método de síntese opera por diagramas ao invés de conjuntos, permitindo que o designer

projete a formas que satisfaçam as demandas em *L* sucessivamente compostas e fundidas em direção à solução do problema como um todo – a *realização* do programa.

Cabe ressaltar que os contornos lógico-matemáticos do método apresentado vão além da apropriação da teoria ingênua dos conjuntos e da codificação binária para os grafos. Os processos de abstração e automação, apresentados por Wing (2008a)[23] como essências do Pensamento Computacional possuem respectivamente a mesma natureza do *programa* e *realização* descritos. Os aspectos analíticos e sintéticos do programa e sua realização operam pela construção de abstrações descendentes e automações ascendentes, onde os níveis de análise tem como entradas e saídas decomposições das demandas do contexto de projeto e composições das soluções formais produzidas pelo designer. No restante da sua obra, Alexander (1964, cap. 8, 9 e apêndices) apresenta o detalhamento algébrico do seu método, formalizando o tratamento matemático para a decomposição do conjunto de demandas do problema utilizando um programa desenvolvido por ele e Marvin Manhein para o computador modelo IBM 7090[24].

Embora o método de Christopher Alexander não seja o único de orientação computacional[25], trata-se de um exemplo pioneiro de processo de projeto que se apropria explicitamente de um raciocínio lógico-matemático para lidar com a complexidade dos problemas, realizando-o por meio de aplicações implementadas em um computador.

Pode-se sugerir, numa primeira leitura da discussão acima, que Alexander ofereceu uma contrapartida analítica para as experiências com as imagens de síntese que ocorrerem no mesmo período, contribuindo para o desenvolvimento de abordagens de projeto como o Design Paramétrico, que na prática integram análise e síntese no mesmo ambiente computacional de forma interativa e permitem o uso dos computadores como facilitadores de certos processos de design – que chamaremos a partir de agora de potencial *instrumental* da Computação no Design.

No entanto, uma segunda leitura é ainda mais relevante para os objetivos desta tese: as aplicações do Pensamento Computacional – análise baseada na teoria ingênua dos conjuntos, recursividade, heurísticas, abstrações, automações, linguagens de programação e o programa construído para auxiliar o processo – faziam parte da própria noção de projeto de Alexander e do seu mundo projetual. Em outros termos, a dimensão computacional daqueles problemas de Design não estava nos problemas ou demandas de

[23] Ver p. 31.
[24] Mainframe utilizado entre 1958 e 1969. Disponível em http://tinyurl.com/ibm7090. Acesso em 01 de outubro de 2013.
[25] Ver Jones (1992) para um compêndio de métodos de projeto.

projeto e sim na interpretação do designer-solucionador – por sua vez, referenciado daqui em diante como o potencial *representacional* da Computação no Design.

Essa distinção é importante por reconectar o conjunto de autores e teorias da Computação às intenções originais de Alan Kay e Seymour Papert apresentadas na primeira parte deste trabalho. Por um lado, o potencial instrumental da Computação no Design é inegável e encontra-se de certa forma consolidado, uma vez que há inúmeras opções de aplicações nas mais diversas áreas que ampliaram as possibilidades técnicas e criativas dos projetistas. Tais aplicações não deixam de mediar a relação dos designers com a Computação enquanto oferecem recursos de análise e síntese que, inevitavelmente, são fundados em princípios computacionais. Por outro lado, o potencial representacional da Computação para as habilidades de projeto, que pode inclusive dispensar o uso de um computador no processo, parece ainda carecer da mesma atenção e interesse que a vertente instrumental.

As perspectivas de Papert, Kay e Alexander se encontram no ponto onde o pensamento projetual em si seria mediado pelos Princípios da Computação, independentemente do uso instrumental dos computadores, no sentido proposto por Vygotsky (2007, p.55) e que reforça a distinção cognitiva entre os potenciais instrumental e representacional:

> A função do instrumento é servir como um condutor da influência humana sobre o objeto da atividade; ele é orientado *externamente*; deve levar necessariamente a mudanças nos objetos. Constitui um meio pelo qual a atividade humana externa é dirigida para o controle e domínio da natureza. O signo, por outro lado, não modifica em nada o objeto da ação psicológica. Constitui um meio da atividade interna dirigido para o controle do próprio indivíduo; o signo é orientado *internamente*. Essas atividades são tão diferentes uma da outra, que a natureza dos meios utilizados por elas não pode ser a mesma.

A abordagem para o Design Computacional que une os autores referidos e interessa aos objetivos desta tese é a que transborda os usos instrumentais dos computadores, explorando a função cognitiva dos Princípios da Computação como mediadores da atividade de projeto dos designers.

A próxima seção introduz o Construcionismo de Seymour Papert, uma abordagem pedagógica pioneira com muitos objetivos comuns a esta tese, e examina com mais detalhes o potencial representacional da Computação nos processos de resolução de problemas numa perspectiva de orientação construcionista.

1.3 – Construcionismo e Representações

A abordagem construcionista foi descrita por Seymour Papert e seus colaboradores em diversos trabalhos, embora três sejam de especial relevância para esta tese[26]. Cabe ressaltar que o interesse inicial do autor era investigar o potencial não-instrumental da Computação com crianças na educação básica, mas a proposta dos pesquisadores associados ou simpatizantes do construcionismo foi gradualmente ampliada para adolescentes (Rusk, Resnick e Cooke, 2009) e adultos (Maeda, 2001).

Em *Mindstorms – Children, Computers and Powerful Ideas* (Papert, 1980), o autor desenvolve duas ideias principais: 1) a de que seria possível construir computadores de forma que a comunicação com eles fosse um processo natural e 2) que aprender a se comunicar com o computador poderia transformar os meios pelos quais os outros aprendizados ocorreriam. Em relação ao problema da comunicação com o computador, Papert argumenta que a situação ideal deveria ser mais parecida com alguém vivendo em um certo país e aprendendo o idioma local naturalmente, embora a situação mais frequente seria semelhante ao aprendizado de uma segunda língua em salas de aula tradicionais. O computador poderia ser então preparado, por exemplo, para se comunicar utilizando linguagem matemática, oferecendo ao aprendiz a oportunidade de viver na "terra da matemática" e aprender sobre este assunto como alguém vivendo na França aprenderia sobre o idioma francês. Ao longo da obra, Papert apresenta o potencial da linguagem LOGO como preparação do computador para se comunicar com crianças pela matemática, física ou geometria e descreve resultados de 10 anos de pesquisa em escolas norte-americanas.

No segundo trabalho, Harel e Papert (1991) formalizam o Construcionismo, aprofundando questões que não foram explicitamente descritas em *Mindstorms* e apresentando dados de pesquisas que endossariam a abordagem:

- Haveria duas facetas do Construcionismo: a *séria* tem suas raízes na construção de estruturas do conhecimento conforme proposto por Piaget, enquanto a *divertida* demanda que qualquer coisa deveria ser entendida por sua construção, de forma recursiva e autorreferente;
- A atividade construcionista, ao integrar Matemática, Arte e Design, aumentaria a eficiência da instrução de um professor no mesmo assunto;

[26] A obra *The Connected Family: Bridging the Digital Generation Gap* (Papert, 1996) foi deixada de fora desta revisão por abordar questões específicas acerca do papel dos pais na educação de crianças nascidas na era digital.

- O Construcionismo é uma teoria sobre a aprendizagem, não estando restrita aos computadores. Os autores oferecem exemplos de ambientes de aprendizagem de orientação construcionista utilizando nós de corda (Strohecker, 1996), robótica (Resnick e Ocko, 1991) ou mesmo papel e lápis;
- O Construcionismo valoriza outros métodos além dos analíticos e abstratos – a abordagem *hard*, ou do planejador – da resolução de problemas, enfatizando a importância de estratégias concretas, incrementais, que dependem da experimentação do percurso sem planejamento prévio – a abordagem *soft*, do *pintor-programador* ou da *bricolagem*. Essa perspectiva, denominada *pluralismo epistemológico* (Turkle e Papert, 1991), é a principal divergência entre o Construcionismo e o construtivismo piagetiano, visto que defende que o pensamento concreto não seria um estágio em progressão para a emergência do pensamento formal. Segundo a visão construcionista, certos perfis de aprendizes poderiam ser bem-sucedidos na vida adulta, resolvendo problemas complexos, sem necessariamente abandonar o pensamento concreto;

Por fim, em *A Máquina das Crianças*, Papert (1994) explicita a distinção entre as estratégias pedagógicas pautadas pelo aumento da qualidade e quantidade de ensino e aquelas orientadas pelos interesses e visões de mundo do aprendiz, favorecendo o *máximo* de aprendizagem com *o mínimo* de ensino. O objetivo do autor é repensar o papel da escola e dos professores para uma geração de aprendizes que cresceram aprendendo mais com videogames do que nas salas de aula. Recuperando algumas ideias do livro anterior, Papert (p.128) explica que o Construcionismo teria como principal característica a atribuição de

> [...] especial importância ao papel das construções no mundo como um apoio para o que ocorreu na cabeça, tornando-se, desse modo, menos uma doutrina puramente mentalista. Também leva mais a sério a ideia de construir na cabeça reconhecendo mais de um tipo de construção [...] e formulando perguntas a respeito dos métodos e materiais usados.

A breve revisão dos textos de Papert e seus colaboradores suscita pontos relevantes sobre a perspectiva construcionista, a saber:

a) Mais foco na aprendizagem e menos na instrução;

b) A importância de se preparar o computador ou qualquer outro meio com as linguagens apropriadas para a comunicação com o aprendiz, considerando as peculiaridades das áreas de conhecimento envolvidas;

c) A necessidade de se planejar situações de aprendizagem que contemplem e valorizem diferentes perfis de aprendizes (*hard* e *soft*);

d) A autonomia do sujeito frente à construção do próprio conhecimento sobre os problemas que busca resolver.

Tentativas de realização prática dos pontos mencionados podem ser observadas em experiências pedagógicas há quase 50 anos[27]. Não obstante, a experiência mais relevante para esta tese foi desenvolvida para além das salas de aula pela equipe do Xerox PARC, que construiu a interface gráfica que simulava um escritório americano típico nos anos 1970: mesas de trabalho (o *desktop*), armários de arquivos com documentos organizados em pastas suspensas, calculadoras, planilhas para serem preenchidas, uma lixeira para descartar o que não é mais necessário e assim por diante. Esse modelo de interação que se mantém dominante até os dias de hoje foi batizado de *metáfora do escritório* (Moggridge, 2007) pode ser entendido como uma tentativa de preparar o computador para se comunicar facilmente com o usuário. Assim como o aprendizado do idioma francês seria mais natural na França, a metáfora do escritório oferecia a seus usuários a oportunidade de aprender naturalmente a linguagem do trabalho no mundo dos escritórios dos anos 1970, adotando ambientes, objetos, rotinas e tarefas típicas dos trabalhadores da época como mediadores da interação humano-computador.

A influência das ideias de Papert na concepção do computador pessoal do Xerox PARC foi explicitamente mencionada por Alan Kay (1989). Além da obra *Os meios de comunicação como extensões do homem* de Marshall McLuhan e o computador FLEX[28], Kay diz ter ficado impressionado pelo uso que Papert fazia das ideias de Piaget para auxiliar crianças a aprenderem a programar computadores com LOGO. Contudo, foi por meio do trabalho de Jerome Bruner que Alan Kay encontrou o caminho mais fértil para descrever seus objetivos com a computação pessoal (p.128).

Bruner (1964) elaborou uma teoria da aprendizagem descritiva e normativa, baseada em três formas de representação do conhecimento em domínios de resolução de problemas e

[27] Ver Barrella (1991), Rocha (1995), Harvey (1997) e Feruzzi (2001) para apresentações de experiências práticas com a linguagem LOGO e a abordagem Construcionista em diferentes contextos.

[28] Kay (1989, p.123) menciona ter construído o computador FLEX em parceria com Ed Cheadle como parte da sua tese de doutorado. Esse computador continha diversas ideias aproveitadas anos mais tarde no Xerox PARC, incluindo interface gráfica baseada em ícones.

que revisam questões dos estágios sensório-motor, operatório concreto e operatório formal propostos por Piaget e Inhelder (2003). A representação *ativa* consiste em um conjunto de ações motoras apropriadas que representam eventos passados; a *icônica* sintetiza eventos pela organização seletiva de perceptos e imagens, pela estrutura espacial, temporal e qualitativa do campo perceptivo e suas imagens transformadas; e a *simbólica* corresponde a um conjunto de proposições lógicas ou simbólicas derivado de um sistema simbólico arbitrário, regido por normas ou leis para formar ou transformar proposições (Bruner, 1976).

Em sua teoria, Bruner (1964, 1976) discute as aplicações das formas de representação em diferentes domínios do conhecimento buscando a economia para o processamento da informação. Segundo o autor, cada tipo de informação teria uma forma mais econômica e mais potente para ser estruturada e representada. A sequência por meio da qual as informações são apresentadas em cada forma também influiria na facilidade de compreensão pelo aprendiz. Essa teoria foi utilizada por Kay (1987, 1989) para cunhar o *slogan* "*Doing with Images makes Symbols*", que definiria seu método para preparar os computadores para se comunicarem com o usuário comum.

Kay defendia que a interação com o computador deveria partir do concreto – *fazer com imagens* – em direção à abstração da construção de símbolos. Nesse método os usuários poderiam manipular imagens diretamente na tela do computador utilizando o *mouse* como extensão das mãos (aponte e clique na opção desejada, clique na janela que deseja mover, redimensionar ou trazer para frente) que selecionam imagens que remetem a funções (janelas, menus e ícones dos programas, documentos ou pastas). A linguagem de programação Smalltalk foi planejada para criar programas que também se comunicariam por meio de imagens manipuláveis diretamente pelo usuário.

Tabela 1.3 – Concepção de Alan Kay a partir de Bruner – adaptado de Kay (1989)

Dimensão	**Representação**	**Computação Pessoal**	**Função**
Fazer *Doing*	Ativa	*Mouse*	Saber onde está, manipulação
com Imagens *with Images*	Icônica	Ícones, janelas	Reconhecer, comparar, configurar, concretude
constrói Símbolos *makes Symbols*	Simbólica	Smalltalk	Combinar longas cadeias de raciocínio abstrato

Recuperando os pontos essenciais do Construcionismo listados anteriormente, pode-se sugerir que o método adotado no Xerox PARC realizou cada um deles, pelo menos no contexto dos escritórios norte-americanos dos anos 1970:

a) A adoção da metáfora do escritório minimizou a demanda por instrução para que os usuários compreendessem o modo de operação do computador, uma vez que o contexto e as tarefas eram potencialmente familiares;

b) A linguagem definida para a interação humano-computador, baseada na manipulação direta dos ícones que representam graficamente ambientes, tarefas e objetos do escritório na interface parece ter sido suficiente para que os usuários construíssem suas interpretações sobre o funcionamento do sistema;

c) Usuários com perfil *soft* poderiam utilizar os computadores apenas pela manipulação e exploração direta dos ícones, enquanto usuários *hard* poderiam personalizar e estender o sistema planejando programas com Smalltalk. As tarefas típicas de um escritório encontravam-se acessíveis aos dois perfis;

d) Ainda que os recursos do sistema fossem os mesmos para todos os usuários, a programação e a manipulação direta proporcionavam oportunidades de construção de conhecimento personalizadas e abertas a diferentes interesses e necessidades de um ambiente de trabalho.

No tocante ao contexto desta tese, o método desenvolvido por Alan Kay apresentou-se como uma alternativa relevante para criar oportunidades de aprendizagem dos Princípios da Computação, especialmente pela relação direta com o Construcionismo. Contudo, para que o mesmo método fosse adotado, restaria ainda construir o contexto ou metáfora que definiriam a linguagem a ser utilizada nas interações dos aprendizes com o conteúdo nas três formas de representação. Segundo o próprio Bruner (1976, p.75):

> [...] uma teoria da aprendizagem procura considerar o fato de um currículo refletir não só a natureza do conhecimento em si mesmo como também a do conhecedor e do processo de aquisição do conhecimento. É um caso em que é obrigatoriamente mal delineada a fronteira entre sujeito, objeto e método.

A definição do contexto para a aprendizagem dos princípios do CT por estudantes de Design do ensino superior nesta tese teve início por uma revisão da literatura sobre resolução de problemas e uma discussão acerca da relação entre o contexto, o sujeito e os objetos envolvidos no processo.

1.4 – Representações e resolução de problemas

A concepção da cognição como *processamento de informações* e as preocupações quanto à *economia da informação* descritas por Bruner e utilizadas por Kay eram típicas daquele momento histórico no qual pesquisadores pensavam a mente humana como um processador serial de símbolos, inspirado em um computador. Foi neste contexto que surgiram as teorias cognitivistas que influenciaram Christopher Alexander[29] na elaboração do seu método de projeto de natureza computacional.

Mayer (1990 apud Eysenck & Keane, 2007) define a solução de problemas como o "processamento cognitivo direcionado para a transformação de uma determinada situação em uma situação de objetivo quando nenhum método óbvio de solução está disponível para o solucionador do problema". De forma similar, Matlin (2004) explica que a resolução de problemas é utilizada quando se pretende atingir um determinado objetivo, mas a solução não se apresenta imediatamente na memória. Já Sternberg (2010) descreve o ciclo de resolução de problemas como uma sequência de passos que inclui a identificação do problema, a definição do problema, a formulação de estratégia, a organização das informações, a alocação de recursos, o monitoramento e a avaliação.

Shapiro (2011) destaca que as abordagens cognitivistas de resolução de problemas teriam como marcos inaugurais com o *General Problem Solver* criado por Herbert Simon e Allen Newell, os testes de recuperação de memórias de Saul Sternberg, as teorias computacionais da percepção de David Marr e Tomaso Poggio, e a visão solipsista da mente. Newell e Simon (1958) propuseram a Teoria do Processamento da Informação para a Resolução de Problemas (TPI), um dos modelos fundadores da Inteligência Artificial (IA). O objetivo dessa nova área era programar computadores para exibirem comportamentos considerados inteligentes[30] e, a partir da comparação de relatórios das operações realizadas pelas máquinas com protocolos verbais de seres humanos resolvendo os mesmos problemas, encontrar caminhos para compreender os processos cognitivos do homem. Tais comportamentos eram verificados principalmente na resolução de problemas estilo quebra-cabeças como xadrez, Torre de Hanói ou de lógica proposicional (Langley & Rogers, 2005), e uma série de teorias sobre as estratégias adotadas por seres humanos foi elaborada e aplicada nos mais diversos contextos.

[29] Ver p. 39.

[30] Este é o argumento central do Teste de Turing (1950): se uma máquina for capaz de exibir comportamento inteligente indistinguível do comportamento de seres humanos na mesma situação, tal máquina deveria ser considerada "inteligente".

A visão solipsista da mente, realizada teórica e experimentalmente na TPI, tem implicações importantes nas formas de representação do conhecimento na resolução de problemas. Essa perspectiva, resumidamente (Lourenço, 2002; Newell e Simon, 1976; Simon, 1978), sugere que a mente receberia entradas (*inputs*) pelos órgãos do sentido a partir de um mundo exterior, dado e anterior a quem o percebe. O processamento consistiria na manipulação de símbolos por regras computacionais e as representações seriam ocorrências em um sistema formal (Varela, Thompson e Rosch, 2002). As saídas (*outputs*) desse processamento seriam o comportamento do indivíduo e sua ação (movimento, fala, manipulação de objetos) no mundo. O processo de resolução de problemas dos cognitivistas consistia, essencialmente, na construção de regras de produção (algoritmos) que, a partir da representação dos estados do problema, desencadeariam operações de transformação que reduziram a diferença entre o estado inicial e o objetivo final (Sternberg, 2010). Esse processo envolveria mecanismos perceptivos, mnemônicos e atencionais, que foram investigados e quantificados por pesquisadores de orientação cognitivista[31].

As dificuldades encontradas pelos indivíduos solucionadores eram explicadas nos termos de representações incorretas do problema, dos seus estados, uso de operadores inapropriados para a situação (Simon, 1970), ou por fixação funcional (Flavell, Cooper e Loiselle, 1958). Quanto à tipologia, os problemas foram definidos como *bem-formados*, quando os estados inicial e final são claros e os operadores conhecidos; *malformados* (Chi & Glaser, 1985) quando o estado inicial, meta, operadores ou os três são mal definidos; ou *capciosos* (Rittel e Weber, 1973), quando os problemas são particulares a ponto de não possuírem formulação definitiva, dependendo principalmente da representação elaborada pelo indivíduo para solucioná-lo e da dinâmica do processo que não necessariamente busca uma única resposta correta. Os conhecimentos adquiridos ou estratégias desenvolvidas nos percursos de resolução poderiam ser generalizados para problemas *isomórficos* (Sternberg, 2010), onde a situação inicial, estados, operadores ou objetivos a serem alcançados seriam semelhantes.

Parece razoável entender que a *representação* elaborada pelo sujeito para o problema, para seus estados, objetivos ou metas e os operadores mais apropriados para cada situação era fundamental para os cognitivistas. Não obstante, esse cenário considerava a cognição do sujeito em separado do mundo no qual atuava, do seu próprio corpo e das

[31] Miller (1956) é um trabalho clássico que ilustra bem as questões da época. O autor investigou os processos cognitivos como processadores de informação com capacidade limitada e elaborou um modelo sobre o funcionamento da percepção e memória que ficou conhecido como "o mágico número sete ± 2", referente ao número médio de itens (*chunks*) de mesma natureza que poderiam ser percebidos e processados em média por um ser humano.

suas ações. Doise e Mugny (1997) explicam que as concepções dominantes da psicologia cognitiva seriam abstrações, ao considerarem o indivíduo isoladamente do contexto social no qual as atividades cognitivas ocorreriam. Abric (2001, p.200) completa essa visão:

> [...] o comportamento de um grupo em situações de resolução de problemas não está determinado pelo tipo da tarefa que efetua, sim pela representação que faz da tarefa. Um grupo idêntico utilizando duas representações distintas de uma mesma tarefa adota comportamentos diferentes, independentes da realidade objetiva.

As particularidades dessas situações, as variadas histórias dos sujeitos e a própria multiplicidade de problemas apontariam para a necessidade de teorias cognitivas diferentes das computacionais, que considerem um indivíduo que conhece, constrói e representa seu próprio mundo enquanto atua nele. Se o conhecimento do solucionador determinaria, de alguma forma, a representação do problema, e essa representação guiaria o reconhecimento das soluções apropriadas (Chi e Glaser, 1985), compreender a origem, desenvolvimento e aplicações práticas desse conhecimento por parte dos indivíduos seria de suma importância para pesquisas como a apresentada nesta tese.

1.5 – Contextualização dos processos representacionais

Pesquisadores de diferentes filiações teóricas desenvolveram trabalhos buscando contextualizar a gênese, estrutura e funções cognitivas das representações, alguns deles antes mesmo da revolução cognitivista ou do aparecimento das primeiras críticas a este movimento. A abordagem genética, como nos lembra Duveen (1994), é necessária para que se possa compreender o desenvolvimento da construção de um fenômeno psicológico, ou seja, os processos pelos quais aquele fenômeno foi produzido. No caso da contextualização da construção e desenvolvimento das representações do solucionador frente a um problema, a revisão a seguir considera tanto os processos cognitivos que se desenrolam pela interação com outros indivíduos quanto com o meio e objetos físicos ou simbólicos. Já a complementariedade entre estrutura e função daqueles processos, ou a ligação entre o *sujeito epistemológico* e o *sujeito psicológico*, oferece oportunidades para investigar a constituição do conhecimento e a finalidade ou valores atribuídos a ele pelos indivíduos em contextos de resolução de problemas (Inhelder e Caprona, 1997).

A perspectiva sócio-histórica

Segundo Wertsch (1985), ainda anos 1930 Vygotsky e seus colaboradores na antiga União Soviética alegavam que o comportamento só poderia ser adequadamente investigado se

considerado como produto da filogênese, da história sociocultural e da ontogênese, que seriam os três percursos básicos do desenvolvimento. A importância dada por Vygotsky ao papel do contexto no desenvolvimento cognitivo pode ser observada quando este autor explica que as funções psicológicas superiores seriam mediadas por operações com signos. Tais operações começariam por um processo interpessoal que seria gradualmente reconstruído internamente até tornar-se intrapessoal, passando de uma atividade externa para uma interna (Vygotsky, 2007). Outro indício pode ser encontrado na teoria da zona de desenvolvimento proximal (ZDP) elaborada pelo pesquisador soviético, que propõe que o nível de desenvolvimento real dos indivíduos poderia ser potencializado pela ajuda de uma pessoa mais experiente na realização de uma tarefa. Além do conceito de ZDP reforçar a importância dada por Vygotsky à interação social como motor do desenvolvimento, parece evidente a importância da intervenção de outras pessoas nas situações onde o aprendizado se adianta ao desenvolvimento (Oliveira, 2010).

O estudo clássico realizado por Luria (2010) no Uzbequistão entre 1931 e 1932 adotou a perspectiva sócio-histórica de Vygotsky para investigar os efeitos da escolarização formal nas habilidades de resolução de problemas em grupos sociais até então sem acesso à educação institucionalizada. Para o autor, as mudanças promovidas pela revolução soviética resultaram em oportunidades singulares de análise dos processos de desenvolvimento cognitivo naquelas comunidades numa perspectiva histórica, uma vez que foi possível observar como indivíduos cujas estratégias de raciocínio haviam se desenrolado principalmente pela experiência direta seriam transformadas pelo acesso às formas dos pensamentos verbal, lógico e discursivo tipicamente introduzidos pela escola.

Uma parte do mesmo estudo (p.160) buscou compreender os mecanismos pelos quais os indivíduos se desvencilhariam da experiência prática direta e se apropriariam dos conhecimentos adquiridos pela educação formal como estratégias de solução, ao mesmo tempo em que se propôs a pesquisar os conflitos que surgiriam nas situações onde o conhecimento prático não se enquadraria ou contradiria as condições lógico-formais necessárias para solucionar os problemas. Luria enuncia que os resultados evidenciariam o papel fundamental das transformações sócio-históricas nas atividades cognitivas, naquele caso representadas por mudanças importantes na estrutura social, no planejamento e coletivização do trabalho e no acesso à escolarização básica. Há uma crítica forte nas conclusões do autor sobre as abordagens da Psicologia que entendem as estruturas fundamentais de percepção, representação, dedução, raciocínio, imaginação, consciência e da própria identidade como fixas e indiferentes às transformações sócio-históricas do contexto onde se desenvolvem (p.218).

No centro do método genético de Vygotsky (2008) estava a investigação da função da palavra no desenvolvimento cognitivo, especialmente na passagem do pensamento para a fala e nas relações estabelecidas pelo individuo para coordenar a construção de sentido, e propondo distinções entre os aspectos semânticos e externo da fala. A palavra seria utilizada "como um meio para a formação de conceitos" (p.73), num processo não linear que ocorreria em duas direções: de um lado, conceitos são formados por sínteses e agregações da experiência; do outro, análises e abstrações descoladas da experiência isolam e examinam elementos para formar conceitos. A síntese dos conceitos espontâneos se desenvolve de forma ascendente em direção à generalização, enquanto os conceitos abstratos desenvolvidos na interação com as variadas instituições sociais seguem a trajetória descendente rumo à concretude e aplicação no cotidiano. As duas linhas genéticas se desenvolveriam independentemente, até se encontrarem em um determinado momento, promovendo mudanças qualitativas importantes (Oliveira, 1992, p.30):

> O indivíduo humano, dotado de um aparato biológico que estabelece limites e possibilidades para seu funcionamento psicológico, interage simultaneamente como o mundo real em que vive e com as formas de organização desse real dadas pela cultura. Essas formas culturalmente dadas serão, ao longo do processo de desenvolvimento, internalizadas pelo indivíduo e se constituirão no material simbólico que fará a mediação entre o sujeito e o objeto de conhecimento.

Não obstante, a perspectiva introduzida por Vygotsky não discute aprofundadamente os mecanismos pelos quais a internalização se desenrolaria, especialmente no que tange à passagem da atividade simbólica do nível interpessoal para o intrapessoal. Wertsch (1985) aponta as dificuldades de considerar o significado da palavra como unidade de análise fundamental do método genético de Vygotsky, sugerindo que a palavra teria a função de mediação semiótica no desenvolvimento cognitivo, ao invés de denotar uma unidade elementar das próprias funções cognitivas. Também ficaram de fora do conjunto de considerações do autor os desdobramentos das relações de gênero, identitárias e de grupo sobre o aprendizado explicado pela ZDP (Castorina, 2010). Mesmo no tocante à visão original da ZDP como uma relação assimétrica de poder entre um indivíduo menos (aprendiz) e outro mais experiente (tutor), haveria outros dois mecanismos a serem incluídos na análise para caracterizar apropriadamente a transição de um conhecimento construído no nível intersubjetivo para o subjetivo (Wertsch, 1985, p.167):

I. *A perspectiva referencial* aborda as diferenças entre tutor e aprendiz quanto aos atos de referenciar os signos envolvidos nas atividades bem como compreender as

referências feitas pelo outro. Perspectivas referenciais distintas baseiam-se em diferentes quantidades e naturezas de informação introduzidas na interação: *dêitica*, onde o referencial é pressuposto e já existiria cognitivamente para os interlocutores, resultando em nomeações simples (este, aquele) ou apontamentos que não denotam necessariamente o significado do objeto referenciado; *expressão de referência comum*, correspondente à introdução do aprendiz às formas de nomeação e categorização cotidiana dos referentes, geralmente resultando em redundância ou pouca informação adicional àquilo que o próprio aprendiz poderia empregar para referir-se ao objeto; e a *expressão informativa do contexto*, que caracteriza o referente de maneira não óbvia para alguém que não entende a definição da situação, desafiando semioticamente o aprendiz a ressignificar o contexto e a compreendê-lo como o faria um membro mais experiente daquele grupo social. Em termos gerais, a perspectiva referencial situa tutor e aprendiz em níveis comuns onde a interação é possível, caminhando da semiose interpessoal fundada em informação menos diferenciada em direção a apropriações e usos criativos de formas de referenciação intrapsicológicas mais elaboradas.

II. O nível de *abreviação* corresponde aos graus das formas explícitas de representação linguística, podendo ser reduzidas (abreviadas) ou expandidas (não-abreviadas), afetando a regulação do outro durante o processo. A análise, no entanto, não se dá a partir da resposta que a elocução abreviada ou não-abreviada eliciaria, mas da expressão utilizada em si. Formas abreviadas podem convidar o aprendiz a identificar e conduzir passos intermediários implícitos ao que foi dito, desafiando-o semioticamente, enquanto formas não-abreviadas reposicionam o direcionamento das ações sobre o tutor.

Além da ampliação do detalhamento no sentido interpessoal para o intrapessoal, outros avanços na teoria original de Vygotsky foram feitos para melhor caracterizar o contexto no qual ocorreriam as interações sociais e o aprendizado. Engeström (2009) relata que a teoria da atividade de Alexis Leontyev deslocou a unidade de análise da ação do indivíduo para a atividade coletiva em um sistema que também envolveria uma comunidade, divisão do trabalho e regras próprias dos diversos contextos históricos, além da mediação entre os indivíduos e artefatos na construção de objetos. Outra revisão da teoria da atividade, resultante da sua internacionalização, passou a discutir as diferenças entre diversas tradições e perspectivas, bem como a incluir no mínimo dois sistemas da atividade da geração anterior em interação na análise. Essa geração é marcada por princípios que: 1) situam a unidade de análise no sistema como um todo em sua rede de relações com outros

sistemas; 2) incluem múltiplas vozes na análise, assumindo que um sistema da atividade é uma comunidade de pontos de vista, tradições e interesses múltiplos; 3) assumem a dimensão histórica da atividade e do sistema no qual ocorre; 4) definem as contradições e tensões estruturais próprias da história daquele sistema como fontes de conflitos mas também de inovação e mudança nas atividades; 5) proclamam a possibilidade de ocorrerem transformações expansivas no sistema pela reconceituação dos motivos e objetos da atividade em direção novas formas radicalmente distintas das do sistema anterior, como numa zona de desenvolvimento proximal coletiva. Atualizações semelhantes da abordagem de Vygotsky podem ser encontradas nas comunidades de prática de Lave (2009) e na teoria social da aprendizagem de Wenger (2009).

Pode-se entender que a definição do contexto no qual são elaboradas as representações dos problemas e tarefas a serem resolvidas pelo aprendiz, à ótica da psicologia sócio-histórica, inevitavelmente precisa incluir mais do que a análise do significado social dos signos envolvidos. Um exemplo dessa necessidade pode ser observado nos estudos de Carraher, Carraher e Schliemann (1982, 1985) sobre a resolução de problemas matemáticos, onde os resultados sugerem que o sentido dos símbolos no contexto parecem contribuir positivamente para o desempenho do solucionador na resolução das mesmas tarefas quando comparadas às suas resoluções descontextualizadas.

Parece importante, a partir do exposto, conceituar a própria atividade do solucionador como um processo simultaneamente histórico, que evolui no tempo e espaço, e social, mediado pelas relações que estabelece com o tutor, ferramentas de tutoria e com a situação, a partir de questões identitárias, grupais, de gênero, de poder, ligadas à divisão do trabalho, aos papéis sociais do envolvidos e da própria evolução do sistema da atividade que dá suporte a toda essa rede de relações.

A epistemologia genética

A interação social também foi explicitada por Piaget (1964) como condição necessária, apesar de insuficiente, para o desenvolvimento cognitivo. Para o autor, a maturação biológica, a experiência individual, a transmissão cultural e interação social, além da equilibração das estruturas cognitivas seriam, juntas, condições necessárias para que ocorra o desenvolvimento do indivíduo. Ainda assim, críticos acusam os estudos piagetianos de diminuírem ou minimizarem a participação do contexto nos processos de desenvolvimento (Doise e Mugny, 1997; Leman, 1998), argumento que pode ser questionado a partir da leitura das obras *O Juízo Moral da Criança* e *Estudos Sociológicos*.

Piaget (1973) elaborou um modelo de interação social inspirado na sua descrição do desenvolvimento de estruturas cognitivas lógicas e nas funções de equilibração e majoração. A "lógica" social construída pelos indivíduos engendraria a acumulação de sistemas morais, jurídicos, crenças e ideologias políticas ao mesmo tempo em que buscaria a supressão dos conflitos por meio da reflexão individual acerca de tais noções coletivas. Para o autor, a lógica das interações sociais seria a da coordenação e conservação gradual de operações reversíveis de um indivíduo que parte da visão do mundo centrada em si em direção ao reconhecimento da posição do outro (p.182):

> [...] as ações dos indivíduos uns sobre os outros, as quais constituem toda a sociedade, só criam uma lógica com a condição expressa de adquirirem elas também uma forma de equilíbrio, análoga à estrutura da qual podemos definir as leis no fim do desenvolvimento das ações individuais.

Garcia (1996, p.214) diz que a epistemologia genética piagetiana teria como pontos essenciais "conceber o conhecimento como processo e não como um estado" e "considerar os mecanismos dos processos de conhecimento na história do saber socializado assim como na história do indivíduo em desenvolvimento". Sobre esses temas, Duveen (1994) expõe que Piaget distingue dois modos de aquisição do conhecimento social, guiados por um tipo especial de conflito produtivo não limitado por influências hegemônicas e aberto à invenção e à construção: 1) aquisição por transmissão social, marcada pela assimetria de poder e onde o conhecimento seria reproduzido em função da influência e prestígio da fonte – a *coação*, de operação semelhante à ZDP de Vygotsky; 2) aquisição através da reconstrução, em relações sociais de simetria entre pares – a *cooperação*. La Taille (1992) explica que Piaget pensa o social e suas influências sobre os indivíduos na perspectiva da ética, tendo em vista que a construção do conhecimento deveria ser orientada por princípios de igualdade e respeito mútuo em uma sociedade democrática.

Para Piaget (1964) conhecer os objetos significa agir sobre eles, transformá-los, modificá-los e compreender esse processo de transformação. A ideia de *operação* diria respeito a uma ação interiorizada que modificaria o objeto, reversível, nunca isolada e sempre inter-relacionada com outras operações em outros níveis, formando uma estrutura de conhecimento. O problema central do desenvolvimento, segundo o autor, seria compreender "a formação, elaboração, organização e funcionamento dessas estruturas" (p.20), que seriam variáveis, na relação com as funções invariantes de organização e adaptação. Esta última, considerada pelo autor como um processo de auto-regulação dos indivíduos na sua relação com o meio, sustenta-se no equilíbrio entre as funções de

acomodação e assimilação (Piaget, 1987): esta corresponde à incorporação de qualquer dado da experiência às estruturas cognitivas existentes, enquanto a anterior consiste nas adaptações das estruturas cognitivas aos novos dados do meio.

Se no arcabouço da epistemologia genética os processos de construção das estruturas do conhecimento são de suma importância, os procedimentos figuram como noção solidária à primeira nas condutas dos indivíduos (Inhelder e Piaget, 1979). É por meio da ação, ou seja, dos procedimentos, que a experiência é adaptada (assimilada e acomodada) às estruturas de conhecimento. As avaliações e seleções funcionais desses conhecimentos estruturais já adquiridos no desenrolar das ações relacionam-se com a noção de esquema, que na obra de Piaget referem-se às unidades de comportamento, organizadoras da conduta cognitiva, remetendo ao que é generalizável e atuando como um instrumento de assimilação (Montangero e Maurice-Naville, 1994). Cellérier (1997) explica que interessa ao construtivismo psicológico compreender o papel dos esquemas como constituintes das realizações das funções, e de meta-esquemas que coordenam outros esquemas de meios e finalidades na resolução de problemas. Para esse autor (p.346):

> [...] o valor de um esquema está ligado à produtividade diferencial que ele traz para o sistema cognitivo, isto é, não é mais do que relativa ao estado deste último, na medida em que depende do subconjunto dos outros esquemas, com os quais interage para a mesma tarefa, em colaboração, mas também em concorrência.

Microgênese em Piaget e Vygotsky

Enquanto a abordagem estrutural – também chamada *macrogênese* no âmbito da obra de Piaget – propõe-se a identificar e compreender as estruturas essenciais do pensamento, a abordagem da *microgênese* investiga as aplicações e atribuições de sentido àquelas estruturas em contextos específicos de resolução de problemas por indivíduos em particular (Inhelder e Caprona, 1997). As investigações de Vygotsky (2008) sobre a relação entre pensamento e palavra também se inscrevem na abordagem microgenética, uma vez que o que buscavam mostrar "não é a maneira como os significados se desenvolvem ao longo de grandes períodos de tempo, mas o modo como funcionam no processo vivo do pensamento verbal" (p.156). As duas vertentes da abordagem microgenética, por caminhos distintos, buscavam compreender as construções realizadas pelos indivíduos em escalas espaço-temporais mais curtas, onde as estratégias poderiam ser observadas nos termos da sua pertinência funcional para aquele problema específico.

> [...] é uma compreensão da tarefa fundada, simultaneamente, na representação da situação final e do *como-fazer* para aí chegar. Estes dois aspectos relevam do próprio sujeito, que os constrói progressivamente (p.28).

Siegler e Crowley (1991) elencam três propriedades que definiriam a abordagem microgenética: a) as observações consideram o período integral, do início da mudança ao momento onde ela atinge uma estabilidade relativa; b) a densidade da observação é altamente relativa à taxa de mudança do fenômeno; e c) o comportamento observado está sujeito à análise de tentativa a tentativa, visando inferir o processo que deu origem tanto a aspectos quantitativos quanto qualitativos da mudança.

Saada-Robert (1997, p.159) explica que a hipótese da abordagem microgenética considera que o solucionador ativa um saber sincrético no início do processo de resolução problema, composto ao mesmo tempo por "generalidades difusas, em relação à situação atual, e de particularismos justapostos, em relação aos conhecimentos anteriores, sobre os quais assenta". Esse saber, segundo a autora, transforma-se ao longo do processo por meio de uma passagem dupla do difuso ao preciso e do esparso ao unitário. *Mudanças de significação*, referentes aos esquemas utilizados e aos objetos envolvidos e suas relações, e as *transformações de controle*, referentes à organização das ações e significações tendo em vista a finalidade, são mecanismos que atuam para 1) ativar esquemas em função da sua pertinência frente à situação (a *rotina*); 2) reforçar ou afastá-los de acordo com sua significação relativa ao objetivo (a *primitiva*); e finalmente 3) ser composto em uma unidade significante que possa efetivamente transformar a situação (o *procedimento*).

Em outro estudo do mesmo grupo de pesquisas, Blanchet (1997) apresenta um método de pesquisa microgenética da construção de significados pela observação experimental das condutas dos indivíduos a partir de unidades significativas da ação:

a) As unidades *procedimentais* referem-se ao controle da atividade prática direta do sujeito sobre o objeto, compostas por índices perceptivos e ações realizadas pelo indivíduo na tomada de decisão nos diversos pontos da solução. Tais unidades não supõem a construção de um plano mais abstrato ou geral em direção à solução final, mantendo-se num nível mais local da exploração da seleção e avaliação de esquemas de ação.

b) As unidades *representativas* correspondem à organização de um plano mais abstrato em função de um sistema de representação diferente da atividade prática, mas que permite a reconstrução de um modelo da situação pelo

> indivíduo. As unidades representativas podem assumir duas funções distintas no controle da atividade: a *função causal* relaciona-se às transformações do objeto percebidas pelo indivíduo e consideradas possíveis, legítimas ou em conformidade com um dado sistema de regras, enquanto a *função teleonômica* consiste na definição do encadeamento das etapas a serem seguidas em função daquelas transformações percebidas como possíveis, subordinadas a uma finalidade – chegar à solução do problema.

Vale ressaltar que a construção do conhecimento a partir das formas de representação ativa-icônica-simbólica propostas por Jerome Bruner[32] e utilizadas na concepção do computador pessoal diferem da perspectiva da microgênese no sentido de que nesta última não há hierarquia de uma modalidade em relação à outra (Saada-Robert, 1997). As particularidades das representações dos solucionadores e das situações resultariam em diversas estratégias, acentuando a importância da valorização do contexto de ação e das formas pelas quais os indivíduos representam e significam o problema, principalmente quanto aos meios e fins possíveis.

A concepção de atividade de Leontyev (2009), ao desenvolver a noção de mediação de Vygotsky, ampliou os níveis de análise originais de forma a considerar as contribuições macroestruturais nas condutas mais individualizadas. Para o autor, a atividade humana tem motivações que influenciam a orientação dos objetivos da ação, que por sua vez se realizam em operações que ocorrem em condições específicas. Esses três níveis (atividade-motivação, ação-objetivos e operação-condições) descrevem as condutas do solucionador de forma distinta da mediação semiótica, mas que ainda assim se desenrolam intimamente relacionadas ao contexto de solução (p.101):

> [...] Sem dúvidas, para a consciência do indivíduo, o objetivo pode parecer uma abstração daquela situação, mas a ação dele não pode ser abstraída dela. Por essa razão, apesar do seu aspecto intencional (o que precisa ser realizado), a ação também tem seu aspecto operacional (como, por quais meios pode ser realizado), o que é determinado não pelo objetivo em si, mas pelas condições objetivas da sua realização.

Pode-se estabelecer um paralelo entre o aspecto operacional da ação de Leontyev, tendo em vista as intenções do solucionador, e as funções das unidades significativas descritas por Blanchet, principalmente quanto às relações entre causalidade, finalidade e seleção

[32] Ver p.45, Tabela 1.3.

dos esquemas de ação, ou aos processos de transformação da rotina ao procedimento de Saada-Robert. De forma geral, conforme nos relatam Eichler e Fagundes (2001), a pesquisa microgenética permite a observação contínua de dois comportamentos distintos: um relativo à construção de hipóteses sobre o problema sobre as quais o solucionador adota estratégias e procedimentos de resolução; e outro correspondente à interpretação das características apresentadas pelo meio sobre o qual o solucionar tenta resolver o problema e constrói novos sistemas de procedimentos. Tal observação adota o estudo de caso como método visando realizar uma análise aprofundada e detalhada das condutas, buscando-se a coerência interna do desenrolar da resolução (projeto, meios e solução) como uma totalidade indivisível analisada temporalmente, através de indícios observados quanto à sua pertinência em um dado momento ou em diferentes momentos da análise (Guterres, Eichler e Del Pino, 2007).

Em relação a contextos socioculturais como o escolar, Kelman e Branco (2004, p.95) explicam que a microgênese teria múltiplas funções ao possibilitar o

> [...] estudo de características do desenvolvimento humano que vão se constituindo na dinâmica das interações verbais e não-verbais e na observação das negociações que ocorrem no fluxo interativo entre professor-aluno e aluno-aluno, no face-a-face.

Colocando de outra forma, a análise microgenética oferece oportunidades para aprofundar a compreensão dos mecanismos de operação das relações de ensino-aprendizagem no âmbito da ZDP, especialmente quanto às qualidades do contexto e às estratégias adotadas pelas partes envolvidas no processo (Branco e Salomão, 2001).

Representações sociais e práticas sociais

O processo de significação e representação do problema, das ações possíveis percebidas pelo solucionador no contexto e a própria finalidade das suas ações podem ser discutidos a partir da Teoria das Representações Sociais (TRS), com ênfase à concepção das práticas sociais na abordagem estrutural da teoria. O estudo das Representações Sociais (RS), iniciado por Serge Moscovici nos anos 1960, interessa-se pela investigação do conhecimento social como objeto de pesquisa fundamental da Psicologia Social:

> [...] engloba em si todos os processos psicológicos suscetíveis de serem estudados, tais como a memória, a percepção, o processamento de informação ou a dissonância etc., processos que atuam em conjunto com os aspectos da vida social, tais como os

> valores, as normas, os símbolos e as tradições para gerar conhecimento em um contexto social (Palmonari e Cerrato, 2011, p.313).

O conceito de RS é definido por Jodelet (1985, p.474, tradução nossa) como "uma forma de conhecimento específico e um saber do senso comum cujos conteúdos manifestam a operação de processos generativos e funcionais socialmente caracterizados". Para a autora, as representações designam uma forma de pensamento prático, orientado à comunicação, à compreensão e ao domínio do entorno social. Vala e Monteiro (2004) explicam que uma representação é social a partir de três critérios: 1) quantitativo, na medida em que é compartilhada por um conjunto de indivíduos; 2) genético, no sentido de que é coletivamente produzida a partir das interações e fenômenos de comunicação em um dado grupo social, refletindo sua situação, projetos, problemas e estratégias nas relações com outros grupos; 3) funcionalidade, considerando que oferecem programas para a ação e comunicação no que diz respeito às questões que emergem no relacionamento com os objetos de interesse de um grupo.

De acordo com Moscovici (2003) uma das hipóteses que explicam as razões pelas quais os grupos criariam representações fundamenta-se na ideia do controle – as representações sociais filtram as informações que provêm do meio ambiente e dessa forma controlam o comportamento individual. Para o autor, a finalidade de todas as representações "é tornar familiar algo não-familiar, ou a própria não-familiaridade" (p.54). Wagner et al (1999) afirmam que os pesquisadores das RS observam conversas e ações relacionadas a fenômenos ou objetos sociais. Nessas conversas, as pessoas atribuiriam características e significados a objetos com o intuito de torná-los parte do mundo social delas. Os autores enfatizam que, para um objeto figurar no mundo de um grupo, ele precisa ser socialmente representado. Já para Spink (2003), o estudo das RS introduz uma nova perspectiva que amplia o conhecimento, enquanto objeto de investigação, para além das fronteiras da ciência e passa a contemplar o conhecimento do homem comum – o senso comum.

No que tange às funções das RS, Moscovici (2003) descreve duas: a) convencionalizar os objetos, pessoas ou eventos, proporcionando-lhes uma forma definitiva, categorizando-os e colocando-os como um modelo de um determinado tipo, compartilhado por um grupo; b) prescrever formas de pensar e perceber a realidade através da tradição e de estruturas imemoriais. O autor ainda qualifica as RS como um fenômeno específico de compreender e de comunicar o que já se sabe, um modo de criar tanto a realidade quanto o senso comum. Numa argumentação semelhante, Abric (2001) cita quatro funções para as RS: 1) função de saber, similar ao processo de convencionalização citado por Moscovici, mas também

contemplando a comunicação social e os processos de intercâmbio, transmissão e difusão do saber do senso comum; 2) função identitária, que situa os grupos e os indivíduos no campo social, fornecendo subsídios para os processos de comparação social, relações de pertencimento e no exercício do controle social pela coletividade sobre os indivíduos de um grupo; 3) função de orientação, similar à função prescritiva de Moscovici, que consiste na intervenção das representações na definição da finalidade de uma dada situação, determinando, de antemão, os tipos de relações relevantes para o indivíduo ou, na resolução de uma tarefa, quais estratégias cognitivas serão adotadas; 4) funções justificadoras, que permitem a explicação das ações e comportamentos de um grupo em relação aos outros, podendo justificar e perpetuar a diferenciação social.

No processo de geração das RS, os mecanismos de ancoragem e objetivação são responsáveis pela transformação do não-familiar em familiar (Moscovici, 2003). A ancoragem classifica e fornece um nome a algo estranho, afim de enquadrá-lo num sistema de categorias de conhecimentos pré-existente e familiar aos indivíduos. O processo de ancoragem articula as três funções básicas da representação: função cognitiva de integração da novidade, função de interpretação da realidade e função de orientação das condutas e relações sociais (Jodelet, 1985). Já a objetivação torna concreto aquilo que é abstrato, transformando um conceito em imagem de uma coisa (Trindade, Santos e Almeida, 2011). A objetivação relaciona-se à organização dos elementos constituintes da RS e dos processos por meio dos quais tais elementos adquirem materialidade, ocorrendo num percurso de três momentos: construção seletiva, esquematização e naturalização (Vala e Monteiro, 2004). Especificamente sobre o processo de ancoragem, Doise (1992) descreve três tipos de análise possíveis: a primeira, *psicológica*, investiga a ancoragem a partir das experiências individuais; a segunda, *sociológica*, estuda como as relações de pertencimento e filiações dos indivíduos a grupos sociais influenciam as representações e crenças compartilhadas por seus membros; já a terceira, *psicossociológica*, analisa os posicionamentos ideológicos dos indivíduos sobre as diferentes relações e categorias do campo social.

As funções descritas para as RS são fundamentais para a discussão do conceito no contexto de resolução de problemas, especialmente no que diz respeito à aproximação teórica com as funções de adaptação cognitiva. Conforme sugere Jodelet (1985), a representação social elaborada por um grupo define os objetivos e procedimentos de solução específicos para seus integrantes, não necessariamente levando em conta a realidade da estrutura funcional da tarefa. Nos termos das funções dos esquemas citadas anteriormente – assimilar dados da experiência às estruturas de conhecimento existentes

– a perspectiva psicossocial das RS permitiria compreender como o conjunto de Princípios da Computação seria representado pelos estudantes de Design e como tais representações sociais influenciariam suas práticas no processo de resolução de problemas de projeto. Nessa direção, vale recuperar o esclarecimento de Palmonari & Cerrato (2011) sobre a visão de Moscovici de que as RS não seriam a representação de um objeto, mas o próprio objeto. Para os estudantes de Design, presume-se que suas representações acerca dos Princípios da Computação seriam a própria Computação e todas as estratégias de solução, assim como as práticas de resolução de problemas, estariam subordinados aos elementos daquelas RS elaboradas e compartilhadas por aquele grupo.

Segundo Sá (1996), a teoria do núcleo central das RS, elaborada por Jean-Claude Abric nos anos 1970, debruça-se principalmente sobre os processos de transformação das representações originadas em mudanças nas práticas sociais. Essa teoria surgiu a partir de uma hipótese de que toda representação seria organizada em torno de um núcleo central, que por sua vez seria constituído por um ou mais elementos que dariam significado à representação. Esse núcleo central desempenha duas funções na estruturação das representações (p.70): a) *função geradora*, de criação ou transformação do significado dos outros elementos constitutivos da representação; b) *função organizadora*, de determinar a natureza dos laços que unem os elementos da representação e lhe conferir estabilidade.

Abric (1993) expõe que as representações teriam como primeira característica a estabilidade e o movimento, a rigidez e a flexibilidade; e como segunda seriam tanto consensuais quanto marcadas por diferenças individuais. Essas características aparentemente contraditórias evidenciam a existência de sistema interno duplo na abordagem estrutural, com papéis complementares nos processos representacionais:

- *O sistema central*, composto pelo núcleo central da representação, determinado por questões históricas, sociológicas e ideológicas, marcado pela memória coletiva do grupo que a elabora e pelo sistema de normas a que se refere. O sistema central é estável, coerente e resistente a mudanças, mantendo-se relativamente independente do contexto imediato onde a representação se apresenta;

- *O sistema periférico* é um complemento essencial do sistema central do qual depende, cumprindo um papel funcional de regular e adaptar os elementos centrais da representação às situações concretas experimentadas pelos indivíduos. Por isso, é flexível, heterogêneo no que diz respeito ao seu conteúdo, marcado por

modulações individuais e se propõe a construir uma interface entre a realidade e o sistema central, ao mesmo tempo que o protege.

No que tange a relação específica entre representações e práticas, Abric (2001) defende que ambas geram-se mutuamente, apresentando uma série de evidencias encontradas em pesquisas sobre representações sociais que contestam visões de orientação materialista onde as representações seriam determinadas pelas práticas sociais. Para o autor, as representações constituídas seriam tributárias das práticas sociais que o grupo em questão desenvolveu ou com as quais foi confrontado.

A importância da compreensão da mútua determinação de representações e práticas sociais deve-se principalmente ao caráter prescritivo das cognições que formam as RS, ou seja, a implicação destas para as ações realizadas pelos indivíduos ou grupos que as compartilham. A noção prescritiva das RS diz respeito às ações possíveis, em termos condicionais ou incondicionais e absolutos, e como tais prescrições se relacionariam com o sistema duplo das representações.

Sá (1996) explica que o sistema central seria construído por prescrições absolutas e incondicionais, enquanto a o sistema periférico consistiria em prescrições condicionais. Se por um lado essa distinção reforça a função do sistema periférico como interface entre o cotidiano e o sistema central, por outro contribui para o entendimento sobre o percurso das transformações nas representações a partir das práticas. O esquema proposto por Flament (2001) explica que mudanças estruturais na incondicionalidade do núcleo central desenrolam-se sequencialmente a partir de transformações iniciadas nas circunstâncias externas, seguidas de mudanças nas práticas sociais, para só então modificar os prescritores condicionais periféricos e por fim promover mudanças nos prescritores absolutos do núcleo.

O desenrolar das transformações das RS pelas práticas, segundo Abric (1993, 2001), teria relação com a percepção dos indivíduos sobre a reversibilidade ou irreversibilidade da situação. Mudanças reversíveis nas práticas afetariam apenas os elementos periféricos, resultando em transformações aparentes ou superficiais na representação. Já as mudanças percebidas como irreversíveis reduziriam a autonomia do indivíduo em manter suas representações frente às contradições introduzidas pela nova situação, dando início a um processo de transformação que pode ser:

a) *Progressivo*, quando as novas práticas não estão em contradição total com o núcleo central da representação. A transformação se dá sem ruptura ou

fragmentação do núcleo central, com integração progressiva dos esquemas ativados pelas novas práticas rumo à constituição de um novo núcleo;

b) *Resistente*, nas situações onde as representações das novas práticas estão em contradição com o núcleo central, mas os mecanismos de defesa da representação conseguem operar: interpretação e justificação das novas obrigações, racionalizações e menções a informações ou normas externas à representação, inclusive com o surgimento dos *esquemas estranhos*;

c) *Brutal*, quando as novas práticas questionam diretamente o núcleo central da representação, sem possibilidade de operação dos mecanismos de defesa ou integração gradual. A transformação brutal é marcada pelo caráter irreversível das novas práticas, que desencadeiam mudanças diretas e completas no núcleo central e em toda a representação.

Investigações sobre as transformações representações da tecnologia em geral por meio da análise da mudança nas práticas correlacionadas, não apenas dos computadores e da Computação, vêm sendo realizadas desde que os variados dispositivos e tecnologias se tornaram mais presentes no cotidiano das pessoas. O estudo de Singéry (2001) verificou que as diferentes representações elaboradas por funcionários sobre o processo de informatização de uma empresa exerceriam influência sobre práticas deles para lidar com a introdução da tecnologia nas suas rotinas de trabalho, avaliando positivamente ou negativamente as mudanças. Pellegrino (2003) realizou um estudo comparativo sobre os processos de implementação e uso de *intranets*[33] na Itália e Reino Unido, apontando alguns fatores que afetariam esses processos: história e cultura organizacional, imagens partilhadas sobre as *intranets*, negociações entre atores e grupos sociais, repertório e gênero das rotinas daquelas comunidades.

Araque et al (2013) pesquisaram os efeitos de um programa de inclusão digital nos hábitos de famílias de baixa renda nos subúrbios da Califórnia quanto ao uso em geral dos computadores, procura por emprego e auxílio nos estudos. Os autores puderam verificar que famílias participantes do programa utilizaram mais o computador, preencheram formulários de candidatura a vagas de emprego e enviaram mais currículos do que famílias não-participantes. Pasqualotti, Barone e Doll (2012) pesquisaram como idosos frequentadores de um centro de convivência para a terceira idade se apropriaram e significaram as tecnologias de informação e comunicação. O estudo indicou, entre outros

[33] Redes corporativas que conectam computadores no mesmo espaço de trabalho ou remotamente. Tais redes oferecem acesso a serviços e informações de interesse dos trabalhadores.

resultados, um engajamento em atividades que contribuam para a manutenção da participação do idoso no grupo, exercício da cidadania e comunicação com a família. Contarello e Sarrica (2005) exploraram as RS da Internet elaboradas por estudantes universitários italianos e como elas se relacionam com o bem-estar social deles. Hakkarainen (2012) analisou a recusa de idosos finlandeses a utilizarem computadores e acessarem a Internet, identificando que essas tecnologias são representadas por aquele grupo como "coisas e ferramentas inúteis e perigosas que ameaçam a liberdade, estilo de vida, saúde e segurança das pessoas, como criam diferenças entre usuários e não usuários" (p.1212). Os participantes reconheceram a utilidade dos computadores e Internet para outras pessoas, mas não para eles, adotando uma perspectiva defensiva numa sociedade informatizada e que exerce pressões sociais para que todos se tornem usuários.

No contexto educacional, Chaib (2002) discutiu um levantamento sobre as representações sociais da informática elaboradas por professores suecos, apontando três categorias: a) uma visão pessimista, rejeitando a entrada dos dispositivos na sala de aula; b) uma visão otimista, relacionada ao compromisso docente de dominar a tecnologia; e c) a visão realista, que concebe a entrada dos computadores na sala de aula como fenômeno inevitável para o ensino moderno. Também em relação ao contexto educacional, Carbonaro et al (2010) propõem uma estratégia para atrair estudantes do sexo femininos para disciplinas da Computação em cursos superiores por meio do desenvolvimento de jogos digitais. O estudo sugere que a estratégia, por não ser orientada por gênero, teria mais possibilidades de despertar o interesse e aumentar a participação das estudantes em quando comparada às abordagens com conteúdos tradicionalmente específicos para estudantes do sexo masculino.

Em comum, esses estudos reforçam a mútua determinação entre representações e práticas sociais, com ênfase para as estratégias de resistência (idosos finlandeses, funcionários) e adaptação progressiva dos grupos (idosos brasileiros, estudantes, professores, famílias de baixa renda) frente às mudanças desencadeadas por circunstâncias externas (pressões sociais pró-informatização, modernização dos espaços de trabalho) ou por transformações nas próprias práticas (introdução dos computadores nas salas de aula e espaços de convivência para a terceira idade).

Integrando Representações Sociais, Mediação e Epistemologia Genética

As três abordagens para a contextualização dos processos de resolução de problemas apresentadas oferecem contribuições relevantes para o tema desta tese, cada uma em sua especificidade, gerando uma possível triangulação teórico-metodológica:

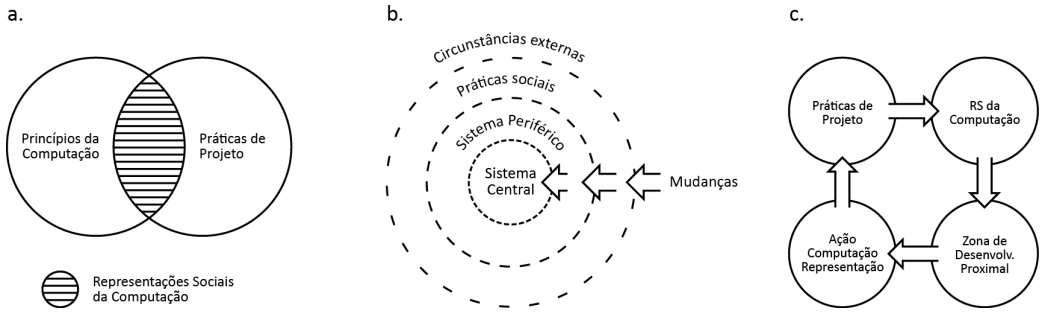


Figura 1.14 – Esquema geral da tese

I. O conceito de *prática social*, tomado na perspectiva estrutural das RS, pode oferecer instrumentos relevantes para pesquisar como os estudantes de Design relacionam-se com a Computação (Figura 1.14-a) e, dessa forma, mapear o nível real do qual partiriam em direção à zona proximal. Sincronicamente, transformações nas práticas sociais de um grupo – os usos criativos dos princípios das Computação pelos estudantes de Design nos seus projetos – também são fontes de mudanças em potencial de significado no objeto de representação – a própria Computação enquanto área de conhecimento. Essas mudanças poderiam se desenrolar por transformações graduais nas práticas de projeto (Figura 1.14-b), principalmente pela introdução de novas ferramentas de aprendizado.

II. A ZDP de Vygotsky (2007), atualizada pela caracterização de Wertsch (1985), se apresenta como um modelo importante para a concepção de uma relação de mediação entre um tutor e ferramentas de tutoria e um aprendiz no que tange ao aprendizado dos Princípios da Computação. As dificuldades resultantes das diferenças de perspectiva referencial entre os polos assimétricos da ZDP sobre os princípios listados por Denning e Martell (2007b), Wing (2008a) ou Brennan, Chung e Hawson (2011)[34] poderiam ser minimizadas pelo uso de referências dêiticas, de uso comum ou informativas daquele contexto. As interações, caso planejadas levando-se em conta os níveis de abreviação apropriados para o enunciado de cada problema e para a ativação dos conhecimentos envolvidos, poderiam ser ao mesmo tempo desafiadoras semioticamente e adequadas à realidade do nível de desenvolvimento do aprendiz (Figura 1.14-c). Também parece importante conceber o Design como um sistema da atividade complexo, onde o estudante seria parte de uma comunidade que relaciona-se segundo regras específicas e onde a divisão do trabalho prevê distinções sociais (identitárias,

[34] Ver pp.31-33.

inclusive) e econômicas entre aqueles que não dominam e os que dominam e empregam os princípios, práticas e perspectivas do Pensamento Computacional nas estratégias de projeto.

III. A análise microgenética (Blanchet, 1997) pode auxiliar na investigação das transformações dos significados dos conhecimentos sobre a Computação por parte dos estudantes exatamente no desenrolar das escolhas de procedimentos e avaliação da pertinência de esquemas no interior das interações da Zona de Desenvolvimento Proximal (Figura 1.14-c). As unidades significativas da ação, em suas formas procedimental e representativa, sendo esta última considerada também quanto às funções de causalidade e finalidade, podem completar a interpretação, por parte do pesquisador, das condutas dos estudantes em direção ao desenvolvimento do Pensamento Computacional no processo de resolução de problemas. A principal contribuição da abordagem microgenética consiste no suporte teórico e metodológico para a compreensão da apropriação criativa de uma forma específica de conhecimento socializado, o da Computação, na invenção dos procedimentos de projeto individualizados dos estudantes.

Finalmente, esta triangulação se propõe a sustentar o percurso de elaboração de uma abordagem para o desenvolvimento do Pensamento Computacional, centrada na *ação* criativa dos estudantes de Design sobre conhecimentos da *Computação*, visando contribuir para a transformação das suas *representações* e das suas práticas de projeto.

Esta tese teve como objetivo geral investigar os elementos das representações sociais elaboradas por estudantes de Design do ensino superior sobre a Computação e os computadores, com o intuito de elaborar uma proposta pedagógica de orientação construcionista para estudantes do mesmo perfil. Como objetivos específicos, pretendeu:

a) Caracterizar o início e desenvolvimento do relacionamento dos estudantes com os computadores nas suas vidas;

b) Mapear as principais práticas realizadas pelos estudantes com o auxílio ou participação dos computadores e verificar como os estudantes se apropriam da Computação e dos computadores na resolução de problemas profissionais, acadêmicos ou cotidianos;

c) Investigar os elementos das representações sociais sobre a Computação e seus princípios, elaboradas pelos estudantes;

d) Elaborar, a partir dos dados a pesquisa, uma abordagem para a aprendizagem dos Princípios da Computação para estudantes de Design alinhada aos princípios construcionistas elaborados por Seymour Papert e à concepção do computador meta-meio de Alan Kay;

e) Experimentar junto aos estudantes, segundo a perspectiva da microgênese, a abordagem pedagógica desenvolvida.

As justificativas para a realização da teste estão fundamentadas principalmente nos desdobramentos da popularização e democratização crescente do acesso às tecnologias de informação e comunicação. O Brasil já é o quarto no ranking dos países que mais vendem computadores no mundo, ficando atrás apenas dos EUA, Japão e China. Segundo o estudo da consultoria IDC (2012), foram vendidos 15,5 milhões de computadores no Brasil apenas em 2012. Se considerarmos a série histórica da Pesquisa Nacional por Amostragem de Domicílios (IBGE, 2011), podemos observar um crescimento entre 2000 e 2011 de 8,5% para 27,3% no número de residências brasileiras que informaram possuir ao menos

um computador conectado à Internet[35]. Assim, nos últimos dez anos, pelo menos um terço dos brasileiros passou a ter a companhia, dentro de casa, desse equipamento que nas décadas anteriores esteve restrito ao universo dos cientistas da computação e demais especialistas da informática. O acesso às tecnologias de informação e comunicação por estudantes e profissionais de outras áreas precisa ser acompanhado de oportunidades de aprendizado dos princípios e conceitos da computação, permitindo que esses usuários realmente sejam criativos e aumentem suas capacidades e habilidades utilizando o computador ao invés de serem limitados por ele.

Uma publicação da Sociedade Brasileira de Computação (SBC, 2011), define a criação de oportunidades para o desenvolvimento do Pensamento Computacional como uma meta fundamental não apenas para estudantes do ensino superior das áreas tecnológicas, mas desde a educação básica. No texto, o Prof. José Carlos Maldonado, presidente da SBC afirma (p.9):

> "[...] Não há dúvidas de que faria uma grande diferença o ensino dos princípios e fundamentos básicos da Computação e da solução de problemas, assim como o desenvolvimento de um olhar crítico sobre o uso e a qualidade dos sistemas de Computação, permeados e entrelaçados com aspectos de direito e cidadania".

No que tange especificamente à formação do Designer, existe uma demanda real de mercado brasileiro por profissionais que consigam dialogar com as tecnologias de informação e comunicação. O desenvolvimento das competências e habilidades ligadas ao Pensamento Computacional por estudantes de Design pode significar mais do que chances de inserção futura no mercado de trabalho: a Computação abre novas possibilidades para o processo criativo, podendo transformar não apenas os produtos da atividade do designer, mas o próprio processo e relações de trabalho.

Do ponto de vista da relevância acadêmica, a pesquisa visa ampliar o conhecimento acerca da incorporação da Computação no cotidiano das pessoas. A interlocução entre o conceito de representação social e o construcionismo, como a apresentada nesta tese, pode oferecer subsídios para a compreensão da influência de crenças e conhecimentos partilhados por determinados grupos sociais enquanto desenvolvem seus projetos.

Finalmente, este trabalho se propôs a explorar possibilidades de complementação do referencial teórico adotado ao articular a Teoria das Representações Sociais de Moscovici,

[35] Cálculo realizado a partir da síntese dos dados da Pesquisa Nacional por Amostragem de Domicílios de 2011. Disponível em http://ibge.gov.br/home/estatistica/populacao/trabalhoerendimento/pnad2011/. Acesso em 03 de setembro de 2013.

a Epistemologia Genética de Piaget e a Psicologia Sócio-Histórica de Vygotsky. Mais do que identificar as similaridades resultantes da perspectiva genética comum às três teorias, a investigação realizada nesta tese demandou a integração daquelas bases epistemológicas para se debruçar sobre o problema de pesquisa, reafirmando a complementaridade entre a gênese, a estrutura e a função dos conhecimentos construídos e compartilhados por um determinado grupo social.

A abordagem metodológica deste trabalho foi dividida em três etapas, sendo a primeira referente à investigação dos elementos das representações sociais elaboradas pelos estudantes de Design do ensino superior acerca da Computação e dos computadores. Esta etapa contou com a participação de estudantes de Design da Universidade Federal do Espírito Santo – UFES, da Universidade de Brasília – UnB (DF), das Faculdades Integradas Barros de Melo – AESO e da Universidade Federal de Pernambuco – UFPE (PE), além de professores das respectivas instituições que se dispuseram a realizar a coleta de dados. O delineamento metodológico desta etapa, seus resultados e a intepretação dos dados são apresentados no capítulo 4.

A segunda etapa consistiu na construção de uma abordagem para a aprendizagem dos conceitos, práticas e perspectivas do Pensamento Computacional inspirada na interpretação dos dados da primeira etapa e nas teorias de orientação construcionista discutidas ao longo desta tese. A concepção e funcionamento da abordagem são detalhadas no capítulo 5.

A terceira etapa, relatada no capítulo 6, corresponde à experimentação da abordagem pedagógica em estudos de casos prototípicos de dois estudantes de Design da UFES, sendo um programador e outro não.

4.1 – Procedimentos e instrumentos

No período de março a maio de 2013, o modelo de questionário do Anexo II foi enviado em versão digital formato PDF, cópias impressas e versão *online* no Google Forms[36] para os professores da UnB, AESO e UFPE. Foi recomendado aos professores que explicassem os objetivos da pesquisa aos alunos e que solicitassem a devolução dos questionários imediatamente após o preenchimento. No caso dos alunos que não puderam responder o questionário no momento definido pelo professor, o link com a versão *online* foi divulgado. A coleta realizada na UFES utilizou os mesmos procedimentos, principalmente com questionários impressos e online. O questionário foi dividido em cinco partes:

1) *Categorização do participante* – sexo, idade, instituição de ensino, ano de ingresso no curso, se sabe programar em alguma linguagem e as atividades mais realizadas com o auxílio do computador, em ordem de importância. Este conjunto de questões teve como objetivo coletar informações demográficas, separar programadores de não-programadores e mapear as principais atividades que cada grupo realiza com a participação dos computadores.

2) *Origens da sua relação com a programação*, apenas para os que disseram que sabiam programar: em qual linguagem de seriam fluentes; programas e projetos desenvolvidos naquela linguagem; como aprenderam a programar e com qual idade; pessoas próximas do participante que também saberiam programar; o que teria despertado o interesse pela programação, solicitando exemplos de situações, eventos ou objetivos que poderiam estar relacionados. A última questão perguntou ao participante se saber programar fez diferença na vida profissional, acadêmica ou pessoal, com uma justificativa da resposta. Esse grupo de questões buscou compreender as circunstâncias nas quais a programação passou a fazer parte dos interesses do participante e em qual contexto – influência de amigos ou parentes, interesses específicos.

[36] Disponível em http://forms.google.com

3) *Evocações sobre os Princípios da Computação* – três palavras em ordem de importância que vinham à cabeça do participante quando pensavam em termos indutores selecionados a partir dos Grandes Princípios da Computação (Denning, 2004; *computação, comunicação, coordenação, armazenagem, projeto, avaliação, abstração* e *automação*), das essências da Computação (Wing, 2008; *abstração* e *automação*) e do conceito de *algoritmo* (Knuth, 1997). O levantamento das evocações teve o objetivo de identificar elementos centrais e periféricos das representações sociais dos participantes sobre os termos;

4) *Relações de cada princípio com a Computação:* foi solicitado ao participante que explicasse como entende a relação de cada termo evocado no item anterior com a Computação, com o intuito de aprofundar a investigação dos elementos das representações.

5) *Design versus Computação* e *Design versus Computadores*: duas questões abertas para que os participante relatem como a Computação e os computadores os auxiliam na resolução de problemas acadêmicos, do cotidiano ou pessoais. Esse último grupo teve como objetivo marcar mais claramente como os participantes pensam a Computação e os computadores no Design.

Participantes

Um total de 86 participantes responderam os questionários na primeira etapa, sendo 30 estudantes da UFES, 41 da UnB, 14 da AESO e um da UFPE (Tabela 4.1).

Tabela 4.1 – Participantes por IES

Instituição	***f***	**%**
Universidade Federal do Espírito Santo	30	34,9
Universidade de Brasília	41	47,7
Faculdades Integradas Barros de Melo	14	16,3
Universidade Federal de Pernambuco	1	1,2
Total	**86**	**100**

Procedimentos de análise

Os dados foram organizados em três grupos, sendo o primeiro correspondente ao perfil do participante (partes 1 e 2 do Anexo II), utilizado como suporte para a análise e interpretação das respostas dos outros grupos. O segundo incluiu as evocações sobre os Princípios da Computação e como os participantes entendiam cada termo evocado em relação à Computação enquanto área (partes 3 e 4). O terceiro grupo de dados foi

composto pelas duas últimas partes do questionário, referentes àquilo que os participantes relataram realizar com o auxílio dos computadores e da Computação nas atividades acadêmicas, pessoais ou profissionais.

O conjunto de dados do segundo grupo foi processado por uma ferramenta desenvolvida pelo autor (Sant'Anna, 2012), similar ao pacote Evoc (Vèrges, 2002). Tais recursos são ferramentas de apoio à análise prototípica, também conhecida por análise de evocações ou das quatro casas (Wachelke e Wolter, 2011). Em síntese, esse processo constrói um quadro com quatro zonas que organizam termos evocados pelos participantes a partir das frequências e ordens de evocação.

+ + **Núcleo Central** Baixa ordem de evocação Alta frequência	**+ -** **Primeira Periferia** Alta ordem de evocação Alta frequência
- + **Zona de contraste** Baixa ordem de evocação Baixa frequência	**- -** **Segunda Periferia** Alta ordem de evocação Baixa frequência

Figura 4.1 – Quadro de ordens versus frequências de evocação

A zona superior esquerda agrupa os termos com ordem de evocação mais baixa e frequência de menções mais alta, onde há probabilidade de se encontrar palavras que representariam elementos do núcleo central das RS, cabendo as ressalvas apontadas por Sá (1996) e Wachelke (2009; 2011) quanto à necessidade de aplicação de outras técnicas para verificação de tal centralidade. A zona superior direita, também chamada de primeira periferia, contém respostas com alta frequência e alta ordem de evocação, indicando elementos secundários da representação. A segunda periferia, situada na zona inferior direita, agrupa termos com baixa frequência e alta ordem de evocação, sendo pouco relevantes para a estrutura da representação no grupo social investigado. Por fim, a zona de contraste, situada no canto inferior esquerdo do quadro, apresenta termos com baixa ordem de evocação e baixa frequência, podendo tanto ser complementos da primeira periferia, como indicar a existência de um subgrupo entre os participantes que valoriza certos elementos da representação de forma diferente da maioria, inclusive com um possível núcleo central distinto. Antes de serem processados pela análise prototípica, os foram lematizados[37] com o intuito de reduzir a variabilidade das flexões de gênero e

[37] Lematização é o processo que agrupa termos pelo radical comum, desde que não haja alterações semânticas.

número. Formas como *computador* e *computadores*, *matemática* e *matemático* foram padronizadas para seguir a evocação mais frequente do termo.

A segunda parte das questões do segundo grupo e o terceiro grupo de questões foram processados pela mesma ferramenta desenvolvida pelo autor, utilizando recursos de apoio à análise de conteúdo (Bardin, 2009) – criação de categorias temáticas e algoritmos de agrupamento e processamento de textos – e distância euclidiana entre os termos (Segaran, 2008). Nesse momento, a análise seguiu o seguinte procedimento:

I. Todas as respostas foram lidas, uma a uma, identificando seus principais temas. Uma mesma resposta pôde ser categorizada em mais de um tema.

II. Os temas foram então agrupados em grupos temáticos maiores, mantendo a coerência interna das categorias.

III. Termos relevantes identificados nos grupos temáticos tiveram suas distâncias euclidianas calculadas em relação às demais palavras de cada resposta, indicando quais palavras coocorrem com maior proximidade. As distâncias unidimensionais foram calculadas para termos com frequência superior a cinco, excluindo-se as palavras de parada, em relação aos demais termos que coocorreriam pelo menos duas vezes com distância euclidiana inferior ou igual a dez.

Em síntese, os resultados das análises foram organizados com o intuito de identificar diferenças nas representações entre estudantes programadores e não-programadores, sempre apresentando primeiramente os dados referentes a todos os participantes (N=86), para em seguida evidenciar particularidades dos estudantes programadores (N=23) e não-programadores (N=63), com a exceção dos casos onde nenhuma diferença nas evocações ou na análise do conteúdo foi encontrada.

Limitações do estudo

Conforme sugere Wachelke (2011), o tipo de material coletado em evocações não permite uma varredura completa do campo representacional, ao mesmo tempo em que elementos consensualmente não compartilhados também refletiriam uma dimensão coletiva que delimitaria o campo da representação.

Também cabe destacar que as preocupações de Wertsch (1985) sobre as diferenças de perspectiva diferencial entre o pesquisador e os participantes do estudo também se aplicam à coleta das evocações. Embora os termos indutores pertençam a um universo semântico potencialmente compartilhado pelos envolvidos, a capacidade dos

participantes referirem-se aos objetos da pesquisa é inevitavelmente variada, principalmente considerando o teor técnico de alguns Princípios da Computação. As justificativas e explicações das evocações coletadas pelo pesquisador auxiliaram a compreender as diferentes perspectivas referenciais, mas certamente não as esgotaram.

Sobre a análise prototípica, é fundamental enfatizar que o número reduzido de participantes que formou os grupos de não-programadores (N=63) e especialmente o de programadores (N=23) prejudicou o processamento das evocações pela ferramenta de análise em termos estatísticos. Nesse sentido, é sensato mencionar que as tabelas de frequências versus ordem de evocação para os termos indutores construídas com dados segmentados por grupo tiveram papel essencialmente ilustrativo, não se configurando como estruturas efetivas do objeto de representação em questão, nem tampouco identificando formalmente seus sistemas centrais e periféricos. Tais tabelas serviram exclusivamente para complementar a análise prototípica das evocações da totalidade dos participantes e auxiliar na compreensão dos temas encontrados na análise de conteúdo. Investigações futuras das representações elaboradas sobre os Princípios da Computação deverão ampliar e igualar o número de participantes em cada perfil, para que uma comparação adequada entre as diferentes estruturas das representações seja realizada.

Uma última limitação a ser considerada diz respeito às especificidades das oportunidades de inserção no mercado de trabalho nas diferentes regiões onde os estudantes que participaram da pesquisa residem (sudeste, centro-oeste e nordeste), bem como as variações no conteúdo programático das disciplinas da área de Computação ofertadas nas grades curriculares dos cursos de Design. Por um lado, maior ou menor oferta de estágios e empregos para estudantes com conhecimentos em programação nas regiões pesquisadas podem incentivar ou inibir o interesse em desenvolver tal habilidade. Por outro, diferentes perfis das instituições de ensino ou interesses de pesquisa do seu corpo docente podem intensificar ou restringir oportunidades de contato dos estudantes com projetos que demandem a experimentação de linguagens de programação e Princípios da Computação. Considerando tais diferenças, assumiu-se o Art. 4º das Diretrizes Curriculares Nacionais para cursos superiores em Design do Ministério da Educação (2004) como parâmetro para nortear a relação desejada para os estudantes com os conhecimentos da área da Computação enquanto técnica, processo criativo, linguagem e tecnologia fundamental do setor produtivo.

Teste de centralidade

As evocações com hipótese de integrarem o núcleo central para os quatro termos indutores comuns e essenciais às diferentes perspectivas do Pensamento Computacional – *computação*, *algoritmo*, *abstração* e *automação* – tiveram sua verificação de centralidade checada a partir de um método misto que combinou uma variação do Índice de Centralidade de RS a partir de Evocações – INCEV (Walchelke, 2009), da confirmação da centralidade pela análise gráfica do questionário de caracterização (Polli e Wachelke, 2013) e técnica do questionamento (*mise en cause*).

O questionário de verificação de centralidade (Anexo III) foi dividido em duas partes:

I. *Caracterização do participante*: idade, ano de ingresso no ensino superior, se sabia ou não programar e, caso soubesse, desde que idade programa e em qual linguagem seria mais fluente. Esse primeiro grupo visa situar o participante do teste em algum dos grupos identificados pela análise prototípica.

II. *Verificação de centralidade para as evocações de cada termo indutor* pela combinação de duas estratégias: a) um quadro onde os participantes indicariam se aquela evocação seria muito importante ou pouco importante para a sua compreensão do termo indutor; b) técnica de questionamento em que o participante deveria responder e justificar se conseguiria pensar no termo indutor sem lembrar de cada evocação relacionada.

O questionário do teste de centralidade foi enviado a 20 alunos do curso de Design da Universidade Federal do Espírito Santo. Os dados foram analisados visando confirmar ou refutar a hipótese indicada pela análise prototípica e de conteúdo.

4.2 – Perfil dos participantes

As idades informadas pelos estudantes estão no intervalo entre 17 e 35 anos, com média de 23,1 anos, moda e mediana em 23 e desvio padrão de 3,8 anos. Dos 86 participantes, 48 são do sexo feminino e 38 do masculino; 41 relataram estar nos dois primeiros anos do curso (ingresso a partir de 2011), dentre os quais oito eram calouros (ingresso em 2013) e 45 nos anos seguintes, sendo o estudante mais antigo da turma de 1999 da UFES. Considerando as grades curriculares vigentes nos cursos da UFES, UnB, AESO e UFPE, pode-se supor que o grupo dos participantes com ingresso anterior a 2011 já teve algum contato com disciplinas com conteúdos específicos envolvendo a Computação ou com usos explícitos dos computadores na prática de projeto (Tabela 4.2).

Tabela 4.2 – Disciplinas com conteúdos específicos da Computação ou usos explícitos dos computadores por IES/período[38]		
Disciplina	**IES**	**Período**
Computação Gráfica I	UFES	3º
	AESO	1º
Computação Gráfica II	UFES	4º
	AESO	2º
Computação Gráfica Aplicada	UnB	3º

No tocante às idades nas quais começaram a utilizar os computadores, os participantes organizam-se em três grupos: usuários cuja relação teve início antes dos onze anos (52), entre 11 e 17 anos (32) e após os 17 anos (dois).

Tabela 4.3 – Atividades com os computadores*	
1ª mais citada	***f***
Programas gráficos	17
Trabalhar	16
Trabalhos acadêmicos	13
Internet, redes sociais e outros serviços online	10
E-mail	9
Pesquisa	6
2ª mais citada	
Internet, redes sociais e outros serviços online	13
Pesquisa	10
Trabalhar	9
Programas gráficos	9
Trabalhos acadêmicos	9
Jogos	6
3ª mais citada	
Internet, redes sociais e outros serviços online	14
E-mail	7
Blogs e notícias	6
Comunicação	6
Pesquisa	5
Programas gráficos	5
Outros programas	5
Filmes e séries	5
Trabalhos acadêmicos	5

* Dados referentes às atividades citadas por no mínimo cinco participantes.

Quanto às atividades realizadas mais frequentemente com os computadores (Tabela 4.3), aparecem em primeiro lugar: a utilização dos programas gráficos do pacote Adobe para desenho, ilustração e atividades correlatas; atividades profissionais reunidas sob o rótulo *trabalhar*, contemplando termos como *freelance* e estágio; referencias a trabalhos

[38] Informações obtidas nas grades curriculares disponibilizadas pelos professores das IES que colaboraram na coleta de dados.

acadêmicos; acesso ou navegação na Internet; redes sociais, incluindo menções diretas do termo e ao Facebook. Em segundo lugar, aparecem os usos da Internet, redes sociais, pesquisas, trabalhar e uso dos programas gráficos. Em terceiro lugar, foram citados a Internet, redes sociais, e-mail, acesso a notícias e blogs, comunicação, pesquisa, uso de programas gráficos, processadores de textos e planilhas, acessos a filmes e séries.

Embora citações ao uso de e-mail e redes sociais pudessem ser agrupadas sob o rótulo de comunicação ou Internet, optou-se por manter as ocorrências em grupos distintos pois alguns participantes mencionaram explicitamente esses termos em diferentes ordens de importância, sinalizando alguma diferenciação relevante. Da mesma forma, trabalhos acadêmicos poderiam envolver a realização de pesquisas, mas a distinção também pode ser observada nas respostas de alguns participantes. Oito participantes não informaram uma terceira atividade realizada com os computadores.

Ainda sobre as atividades, houve apenas uma menção à programação, apesar de 23 participantes terem afirmado saberem programar em alguma linguagem. As Tabelas 4.4, 4.5 e 4.6 apresentam respectivamente os dados sobre: linguagens de programação mais utilizadas pelos estudantes que se definiram como programadores; formas por meio das quais os estudante programadores aprenderam a programar; pessoas próximas aos estudantes programadores que também saberiam programar.

Tabela 4.4 – Linguagens mais utilizadas pelos estudantes programadores (N=23)

Linguagem	*f*
HTML	4
HTML e CSS	2
HTML, CSS e Javascript	2
C#	2
C	2
Javascript	2
Scratch	2
CSS, HTML, PHP, Java	1
Visual Basic	1
Java	1
Pascal	1
Python	1
PHP	1
Processing	1

Tabela 4.5 – Como o estudante aprendeu a programar (N = 23, respostas múltiplas)

Método	*f*
Sozinho	3
Cursos (na IES ou técnicos)	9
Mercado de trabalho	2
Amigos	1
Livros	3
Tutoriais, cursos e vídeos online	11

Tabela 4.6 – Proximidade do estudante programador com outros programadores (N = 23, respostas múltiplas)

Relação	*f*
Colegas de faculdade	19
Colegas de trabalho	12
Parentes próximos	8
Professores e colegas de colégio	2

O participante que disse ter aprendido a programar mais jovem tinha 12 anos, enquanto o que aprendeu mais tardiamente disse iniciado os estudos em programação agora, aos 35

anos. Nos demais a média foi de 19,4 anos de idade, com moda e mediana em 19 e desvio padrão de 5,21 anos. Já no que se refere aos principais programas desenvolvidos pelos que se definiram como programadores, 13 disseram ser programas ligados à sua atuação no mercado de trabalho, cinco mencionaram jogos e cinco programas ligados a estudos da própria linguagem ou acadêmicos. O despertar da motivação para programar relatado nesse grupo de participantes foi atribuído a seis[39] principais causas: desenvolvimento de projetos pessoais (14), atividades acadêmicas (7), inserção no mercado de trabalho (6), curiosidade (6), independência de outros profissionais para realizar trabalhos (4) e criação de interatividade entre usuário e objeto (1). A seguir são exemplificados alguns relatos dos participantes sobre a importância da programação nas suas atividades:

> Independência. uma frase que venho usando é que a faculdade de Design dá visão, mas não dá braços e pernas. Dominar a programação é poder criar, plenamente e sem barreiras práticas, boa parte das coisas que visualizo como designer mas que sozinho certamente não iria concluir. Não irei mais deixar de fazer algo porque o *software* não permite, ao invés disso entenderei, pela computação, cada passo pra se materializar uma ideia, e se não puder fazer isso sozinho ainda assim, pelo menos terei maior noção desses processos para passar adiante. [...] Situações: ideias que não saem do papel em grande parte por falta de experiência prática com programação. (*M, 26 – UFES, independência de outros profissionais para realizar trabalhos*)

> A principio curiosidade, depois o meio acadêmico que escolhi me levou ao desejo de aprender mais e meu trabalho com um coletivo tinha bastante procura dos músicos locais da cidade pelo serviço. (*M, 29 – AESO, motivado por curiosidade, atividades acadêmicas e inserção no mercado de trabalho*)

> Eu gosto de blogs e queria deixar o meu com uma aparência mais atraente e que tenha a ver com minha personalidade. (*F, 19 – UnB, motivada pelo desenvolvimento de projetos pessoais*)

Por fim, quando questionados se saber programar teria feito alguma diferença na vida pessoal, acadêmica ou profissional, os participantes que se definiram como programadores forneceram respostas cujos conteúdos manifestam cinco temas[40], sendo quatro deles coincidentes com as motivações para o despertar do interesse pela programação: saber programar fez diferença na inserção no mercado de trabalho (12); na

[39] As respostas analisadas abordam mais de um tema simultaneamente.
[40] As respostas analisadas abordam mais de um tema simultaneamente.

percepção de independência do participante (4); no desenvolvimento do pensamento lógico (4); na realização de projetos pessoais (4); fez diferença em se tornar um designer melhor (3); nas atividades da faculdade (3); e em outras situações tais como poder ajudar os amigos (1) ou diferenciar o trabalho dos demais estudantes (1). Dois participantes disseram que saber programar não fez diferença.

> Sim, na vida profissional, saber programar me diferencia dos demais designers que por não entenderem a lógica de programação não sabem se comunicar com programadores ou tem mais dificuldade para pensar soluções. (*F, 26 – UnB, diferenciação do trabalho dos demais estudantes, desenvolvimento do pensamento lógico e inserção no mercado de trabalho*).

> Simplesmente toda a diferença. O ponto mais forte é o de autonomia. saber programar me deu uma maior flexibilidade tanto no momento de pensar algum projeto em relação as suas possibilidades reais por exemplo quanto a parte de executá-lo sem necessariamente ter outros profissionais envolvidos. Esse potencial de fazer com as suas próprias mãos talvez seja a maior diferença tanto na vida acadêmica, profissional e pessoal. (*M, 25 – UFES, percepção de independência do participante, realização de projetos pessoais, atividades acadêmicas e inserção no mercado de trabalho*)

> Ser um designer que sabe programar faz diferença ao trabalhar em uma equipe de programadores, mesmo que programar não faça parte de suas atribuições em um projeto. Vários projetos pessoais meus envolvem desenvolvimento de jogos, e naturalmente programação. (*M, 32 – UnB, fez diferença ser um designer melhor, inserção no mercado de trabalho e realização de projetos pessoais*)

> Profissional sim, porque facilita e aumenta meu entendimento nas criações de projetos relacionados. (*M, 29 – AESO, inserção no mercado de trabalho*)

4.3 – Primeira síntese: atividades, paradigmas e motivações

A revisão de literatura apresentada na introdução deste trabalho indicou alguns usos históricos dos computadores por designers, a saber: programas de *desktop publishing* (Meggs, 1997), autoria multimídia (Canter, 1986), composição musical (IRCAM, 2009), programação de imagens de síntese (Maeda, 2001; Reas e Fry, 2007; Venturelli e Burgos, 1999), Arte Generativa (Bense, 1968; Galanter, 2003), Design Paramétrico (Hernandez, 2006) e a própria prática de projeto (Alexander, 1964). A primeira parte dos dados

coletados sugere maior relação dos estudantes de Design com programas gráficos que integram o conjunto de ferramentas do *desktop publishing*, tanto entre aqueles que relataram saber programar quanto entre os demais.

Programas específicos de autoria multimídia como Adobe Flash e Director não foram mencionados por nenhum dos 86 participantes, enquanto um único estudante da UnB informou utilizar programas de produção musical. É importante destacar novamente que, dentre os 23 participantes que informaram saber programar, apenas um estudante da UFES informou a programação como a segunda atividade que realiza mais frequentemente. A análise preliminar do conjunto de atividades relatadas parece indicar a prevalência do *potencial instrumental* da Computação entre todos os participantes da pesquisa, programadores e não-programadores. Os dados endossam os paradigmas do computador (Moggridge, 2007) como meio de expressão (desenhar, projetar, comunicar-se), mas também como veículo (navegar, ler notícias, ouvir música, ver filmes e séries) e ferramenta (realização de trabalhos acadêmicos e profissionais).

Por outro lado, nas respostas dos estudantes que se definiram como programadores, elementos que caracterizariam o *potencial representacional* foram explicitamente mencionados: autonomia e independência, desenvolvimento do pensamento lógico, melhor capacidade de comunicação com profissionais da área de informática e de pensar ou criar soluções de projeto. Vale lembrar que a linguagem mais citada pelos estudantes, o HTML, não é efetivamente uma linguagem de *programação*, mas uma linguagem para a *estruturação* de documentos de hipertexto. Isso quer dizer que o HTML não oferece ao estudante a possibilidade de explorar plenamente os conceitos do Pensamento Computacional sugeridos por Brennan, Chung e Hawson (2011), Wing (2008a) ou Denning e Martell (2007b). Não obstante, alguns marcadores HTML direcionados à formatação de texto[41] são amplamente utilizados em blogs, redes sociais e outros espaços da Internet citados pelos participantes da pesquisa. Pode-se supor que o HTML seja uma linguagem próxima daqueles estudantes por estar inserida em diversas aplicações que fazem parte do cotidiano deles.

Por fim, mesmo nos casos nos quais o HTML não figurou como a linguagem na qual se tem fluência, a motivação para o despertar do interesse pela programação parece conectar coerentemente uma linguagem e o objetivo a ser atingido com os recursos dela pelo

[41] A especificação oficial da linguagem de marcadores HTML é mantida pelo W3C Consortium e está disponível no endereço http://www.w3.org/community/webed/wiki/HTML. Acesso em 01 de novembro de 2013.

estudante: Scratch, Python e C para jogos; PHP, HTML, Javascript e CSS para Web Design e desenvolvimento de sistemas.

4.4 – Análise Prototípica

A segunda parte do questionário permitiu avançar na compreensão dos 86 estudantes acerca dos Princípios da Computação, complementando a primeira análise. Os termos indutores escolhidos para eliciar as evocações foram reunidos da revisão dos trabalhos de Brennan, Chung e Hawson (2011), Wing (2008a) e Denning e Martell (2007b): *computação, comunicação, coordenação, armazenagem, projeto, avaliação, automação, abstração e algoritmo*.

A Tabela 4.6 sintetiza as informações essenciais acerca do tratamento dos dados[42]: total de evocações por termo indutor, zona de corte para a frequência alta, frequência mínima para inclusão na análise e ponto de corte da ordem média de evocação (OME). Procurou-se reduzir as incertezas durante a interpretação das evocações dos participantes sobre cada termo indutor com base na análise de conteúdo das questões da parte quatro do questionário (Anexo II).

Tabela 4.6 – Tratamento dos dados da análise prototípica (N = 86)

Termo indutor	Total de evocações	Frequência alta		Frequência mínima		Ponto de corte da OME
		%	*f*	%	*f*	
Computação	253	5%	12,6	1,5%	3,8	1,5
Comunicação	254	4%	10,16	2%	5,1	1,8
Coordenação	251	4%	10	1,5%	3,7	2,0
Armazenagem	253	3,95%	10	1,5%	3,8	2,0
Projeto	250	4%	10	1,5%	3,75	1,9
Avaliação	250	4%	10	1,2%	3	1,9
Abstração	246	2,8%	6,8	1,6%	3,9	2,0
Automação	242	3%	7,3	1,5%	3,6	1,8
Algoritmo	247	4%	9,9	1,2%	3	1,7

Computação

O termo *computação* (Tabela 4.7) induziu 27 evocações do termo *computador* contra 26 e 18 evocações dos termos *programação* e *tecnologia* respectivamente, que aparecem mais próximos da ordem média dois e com frequências igualmente altas. Este desenho sugere a hipótese do posicionamento dos computadores no núcleo central, enquanto tecnologia e programação estariam na primeira periferia da representação. A zona de contraste foi constituída apenas o termo *cálculo*, mencionado três vezes com ordem média de evocação

[42] Seguimos as orientações de Wachelke e Wolter (2011) para a definição dos parâmetros da análise prototípica.

1,25. A segunda periferia inclui termos como código, sistemas, interatividade, algoritmo e informática.

Tabela 4.7 – Frequência x Ordem de Evocação – Computação (N=86)

f %	*f* >= 5 / OME < 1.5	OME
10,67	Computador	1,37

f %	*f* >= 5 / OME >= 1.5	OME
10,28	Programação	1,81
7,11	Tecnologia	1,94

f %	*f* < 5 / OME < 1.5	OME
1,58	Cálculo	1,25

f %	*f* < 5 / OME >= 1.5	OME
2,77	Código	2
2,37	Sistemas	1,83
2,37	Interatividade	2,33
1,98	Internet	1,8
1,98	Algoritmo	2
1,98	Informática	2,2
1,98	Ferramenta	2,6

Considerando apenas as evocações para o mesmo termo indutor relatadas pelos estudantes que se disseram programadores (Tabela 4.7a), a palavra *computador* permanece com hipótese de centralidade, porém o termo *código* foi citado por três participantes. A primeira periferia passou a incluir a palavra *ferramenta*, citada por três participantes. A palavra cálculo moveu-se da zona de contraste para a segunda periferia.

Tabela 4.7a – Frequência x Ordem de Evocação – Computação (programadores, N=23)

f %	*f* >= 4 / OME < 1.5	OME
10,67	Computador	1,37
4,55	Código	1,33

f %	*f* >= 4 / OME >= 1.5	OME
4,55	Tecnologia	1,67
4,55	Programação	2
4,55	Ferramenta	2,33

f %	*f* < 4 / OME < 1.5	OME
1,52	Adobe, conexão, futuro, criação, pessoal, sistemas, gráficos, bits, tarefas, internet, processamento de informações	1

f %	*f* < 4 / OME >= 1.5	OME
3,03	Cálculo	1,5
3,03	Projeto	1,5
3,03	Linguagem	2
3,03	Desenvolvimento	2,5

Nem entre programadores ou não-programadores apareceram evocações que conectem o termo indutor às questões da computabilidade e da solucionabilidade de problemas (Diverio e Menezes, 2011), aos paradigmas principais da disciplina (Denning et al, 1988) ou às características do Pensamento Computacional (Wing, 2006).

Tabela 4.7b – Frequência x Ordem de Evocação – Computação (não-programadores, N=63)

f %	*f* >= 5 / OME < 2	OME
12,3	Programação	1,78
11,76	Computador	1,36

f %	*f* >= 5 / OME >= 2	OME
8,02	Tecnologia	2

f %	*f* < 5 / OME < 2	OME
2,14	Algoritmo	1,75
1,6	Informação	1,33

f %	*f* < 5 / OME >= 2	OME
2,67	Sistemas	2
2,67	Informática	2,2
2,67	Interatividade	2,2
2,14	Ciência	2
2,14	Internet	2
2,14	Código	2,5
2,14	Conhecimento	2,5

A presença dos elementos programação, tecnologia e ferramenta (este último apenas no caso dos programadores) na periferia pode indicar possibilidades de transformações futuras da representação do objeto concreto para suas aplicações mais abrangentes – sistemas, interatividade, informática, Internet e algoritmos – o que seria fundamental para o desenvolvimento do *potencial representacional* da Computação a partir do *potencial instrumental.* Cabe ressaltar que, entre os programadores, a frequência de evocação da palavra código foi a mesma da de tecnologia, programação e ferramenta, com diferença de 0,30 na OME entre os termos.

Retomando a primeira síntese sobre os dados (p.76), as evocações analisadas parecem reforçar a ideia do computador como objeto que dá concretude aos processos representacionais da Computação, tanto entre os participantes programadores quanto não-programadores. A análise de conteúdo dos temas das respostas sobre a relação entre a Computação e o computador (Tabelas 4.8 e 4.9) ajuda a esclarecer esses processos:

Tabela 4.8 – Análise de conteúdo Computador x Computação (N=86)*

Tema	*f*	%
Computador como processamento	14	9,4
Computador como *produto* da Computação	13	8,7
Computador como *produtor* da Computação	17	11,4
Computador como máquina, instrumento ou ferramenta (física) que computa	39	26,2
Computador como meio para a Computação	28	18,8
Computador (*Hardware*) *versus* Computação (*Software*)	21	14,1
Resolução de problemas, trabalhos e tarefas	14	9,4
Outros	3	2,0

*As respostas contêm temas de uma ou mais categorias.

Tabela 4.9 – Análise de conteúdo Computador x Computação: exemplos de respostas por tema	
Temas	**Exemplos**
Computador como processamento Computador como máquina, instrumento ou ferramenta (física) que computa Computador como *produto* da Computação Computador como meio para a Computação	Ferramenta auxiliar de memória, comunicação e cálculo. Computadores são máquinas que processam e armazenam nossos dados, e a computação estuda os meios pra se fazer isso. O computador foi o equipamento que permitiu que o ato de processar (ou computar) algo fosse delegado do ser humano para a máquina.
Computador como *produto* da Computação	Produto da computação e, também, máquina de fazer computação. Computador é uma das consequências do estudo da computação. A computação possibilita o computador de existir.
Computador como *produtor* da Computação	Sem computador a Computação não existe, uai. Precisa para existir computação. Está diretamente ligado, pois, na minha mente, tudo é feito a partir do computador.
Computador como máquina, instrumento ou ferramenta (física) que computa	Elemento físico primordial para o exercício/conceito.
Computador como meio para a Computação Computador (*Hardware*) *versus* Computação (*Software*)	A computação pode ser feita através de computadores, e para mim está bastante relacionada a softwares e internet.
Computador como processamento Computador (*Hardware*) *versus* Computação (*Software*) Resolução de problemas, trabalhos e tarefas	Entendo que computador é o conjunto de hardware e softwares necessários para receber, processar e devolver informações.

Para os participantes, o computador é um meio físico no qual a Computação *acontece*, e esse acontecimento se dá por processamento, armazenagem, organização e execução de dados. O computador é um aparelho, dispositivo ou ferramenta (um *hardware*) por meio do qual o participante interage com a Computação (o *software*). Este entendimento possui dois extremos, sendo o primeiro o da dependência total entre o meio físico e os processos computacionais e o segundo que considera o computador como um resultado dos conhecimentos da Computação, sendo apenas *mais um* meio para a sua prática. Entre esses dois extremos há uma gradação de concepções que caminham do paradigma do computador como meio para as aplicações dos dispositivos nos processos de resolução de problemas e realização de tarefas e trabalhos.

Comunicação

As evocações sobre o termo *comunicação* (Tabela 4.10) são formadas pelas palavras *informação* e *linguagem* na zona de alta frequência e baixa ordem de evocação, seguidas de *mensagem* na primeira periferia. As palavras com hipótese de pertencer ao núcleo central,

periferias e zona de contraste tiveram suas frequências muito próximas, de forma que o corte foi realizado com base na comparação entre as distâncias euclidianas entre os agrupamentos observados na Figura 4.2.

Com base no gráfico, pode-se questionar a escolha da posição do termo *linguagem* como parte do núcleo central, uma vez que a distância entre esta palavra e *mensagem* no plano formado pelos eixos de frequência e ordem média de evocação é de 0,5, com moda e mediana coincidentes para os dois termos.

Tabela 4.10 – Frequência x Ordem de Evocação – Comunicação (N=86)

f %	*f* >= 4 / OME < 1.8	OME
5,51	Informação	1,79
4,72	Linguagem	1,75

f %	*f* >= 4 / OME >= 1.8	OME
4,33	Mensagem	2,09

f %	*f* < 4 / OME < 1.8	OME
3,54	Fala	1,67
3,15	Pessoas	1,5

f %	*f* < 4 / OME >= 1.8	OME
2,76	Imagem	2,43
2,36	Ideias	1,83
2,36	Troca	2,17
2,36	Meio	2,17
2,36	Expressão	2,17

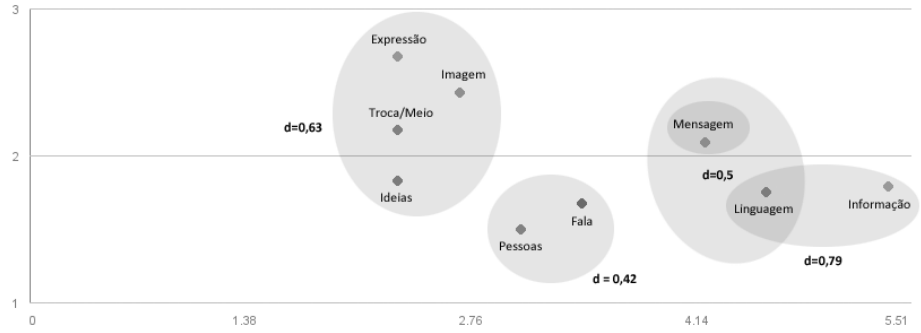


Figura 4.2 – Plano de frequências e ordens médias de evocação para o termo Comunicação

As evocações sobre a *comunicação* inscrevem-se em um conjunto de entendimentos distinto daquele utilizado no âmbito da Computação. Parece razoável entender que essas evocações, consideradas em paralelo ao temas identificados pela análise de conteúdo (Tabela 4.11), relacionam-se a representações hegemônicas[43] acerca da comunicação enquanto processo que permeia diversos domínios do conhecimento, e não apenas aos desdobramentos da Teoria Matemática da Comunicação de Claude Shannon (1948) na transmissão e recepção de confiável de dados.

[43] Vala (1997, p.9) explica que as representações sociais hegemônicas seriam ancoradas "sobretudo em crenças e valores largamente difundidos, indiscutíveis, coercivos e que se referem à natureza do homem e à natureza da ordem social".

As evocações dos programadores (Tabela 4.10a) também posicionaram *informação* com hipótese de centralidade, mas o termo *linguagem* foi deslocado para a segunda periferia em relação às evocações da totalidade dos participantes. A primeira periferia foi composta apenas pela palavra *imagem*, enquanto *mensagem* apareceu na zona de contraste. Entre os não-programadores (Tabela 4.10b), apenas *linguagem* apresenta hipótese de centralidade, com *informação* e *mensagem* na primeira periferia. Tal estrutura pode indicar que programadores e não-programadores representam a comunicação de forma distinta.

Tabela 4.10a – Frequência x Ordem de Evocação – Comunicação (programadores, N=23)

f %	*f* >= 4 / OME < 1.8	OME
6,15	Informação	1,79
4,62	Propaganda	1,75
4,62	Pessoas	1,67
4,62	Ideias	1,67

f %	*f* >= 4 / OME >= 1.8	OME
4,62	Imagem	2,67

f %	*f* < 4 / OME < 1.8	OME
3,08	Mensagem	1,5
3,08	Meio	1,5

f %	*f* < 4 / OME >= 1.8	OME
3,08	Linguagem	2
3,08	Fala	2
3,08	Rede	2
3,08	Expressão	2

Tabela 4.10b – Frequência x Ordem de Evocação – Comunicação (não-programadores, N=63)

f %	*f* >= 4 / OME < 1.8	OME
5,29	Linguagem	1,7

f %	*f* >= 4 / OME >= 1.8	OME
5,29	Informação	1,8
4,76	Mensagem	2,22

f %	*f* < 4 / OME < 1.8	OME
3,7	Fala	1,57
2,65	Pessoas	1,4
2,65	Interação	1,6
2,12	Palavras	1,25

f %	*f* < 4 / OME >= 1.8	OME
2,65	Relação	2,2
2,65	Diálogo	2,2
2,65	Troca	2,2
2,12	Imagem	2,25
2,12	Conversa	2,25
2,12	Meio	2,25
2,12	Expressão	3

Para os participantes programadores e não-programadores, a comunicação e a Computação se relacionam na construção e oferta de dispositivos de comunicação (celulares, e-mails, Internet e outras plataformas), no encurtamento das distâncias, aumento da eficiência ou facilidade de comunicação por esses dispositivos. Os elementos da metáfora do tubo, fundada na Teoria da Comunicação de Shannon (1948) , aparecem nas respostas e conectam Computação e Comunicação em temas como transmissão correta da informação; trânsito ou transporte de dados, números e informações; utilização

da matemática para comunicar; redes e sistemas de comunicação de dados; automatização e massificação da comunicação; transmissão de mensagens do emissor ao receptor.

Tabela 4.11 – Análise de conteúdo Comunicação x Computação (N=86)*

Tema	*f*	%
Computação como linguagem	9	8,7
Computação como mídia	11	10,7
Computação dispositivo de comunicação	42	40,8
Interação (Comunicação) humano-computador	12	11,7
Teoria da Comunicação / Metáfora do tubo	22	21,4
Visualização de dados	1	1,0
Outros	6	5,8

*As respostas contêm temas de uma ou mais categorias.

A Computação também aparece como *mídia* e como *linguagem*, remetendo aos meios e formas de expressão de natureza computacional: expressão de ideias por imagens, textos, jogos, ou sistemas; expansão da comunicação em variados níveis de linguagem e por diversas formas; dimensão simbólica dos números como base da Computação; meio para troca de informações. Um outro tema das respostas centra-se sobre a comunicação entre o computador e seus usuários, de forma que a relação entre a Computação e Comunicação poderia ser entendida como um diálogo, seja na interação entre usuários e produtos; na avaliação dos produtos de um processo; como requisito para uma comunicação eficiente entre um indivíduo ou grupo e uma tarefa desejada; na comunicação entre diferentes produtos (máquina-máquina).

Entre os participantes programadores, os temas da Computação como mídia, como dispositivo de comunicação ou ligados à metáfora do tubo são mais frequentes na análise de conteúdo. Nenhuma resposta desse grupo incluiu a categoria da Computação como linguagem, reforçando o posicionamento do termo na segunda periferia da análise prototípica (Tabela 4.10a). As respostas dos não-programadores, em contrapartida, apresentam maior frequência para o tema da Computação como dispositivo de comunicação, tendo sido mencionado por 31 dos 63 participantes desse grupo. As demais categorias foram proporcionalmente menos citadas pelos não-programadores, sendo a Computação como mídia a menos frequente delas.

Coordenação

Já o termo *coordenação* (Tabela 4.12) induziu a evocação das palavras *organização, gestão* e *ordem* com hipótese de centralidade. A primeira periferia foi ocupada pela palavra *liderança*, enquanto a zona de contraste foi composta por *controle*, e *motora*. A segunda

periferia foi formada pelas evocações *orientação* e *hierarquia*. Numa situação semelhante à do termo indutor comunicação, coordenação eliciou com frequência palavras relacionadas a temas do universo da administração (organização, gestão, liderança), possivelmente ancoradas em representações igualmente hegemônicas, com a exceção de *motora* na zona de contraste. Essa possibilidade pode ser sustentada pela análise de conteúdo (Tabela 4.13), onde a categoria temática *gestão e questões organizacionais* reuniu o maior número de respostas sobre a relação entre Computação e Coordenação.

Tabela 4.12 – Frequência x Ordem de Evocação – Coordenação (N=86)

f %	*f* >= 4 / OME < 2	OME
17,93	Organização	1,76
6,77	Gestão	1,88
5,58	Ordem	1,36

f %	*f* >= 4 / OME >= 2	OME
8,37	Liderança	2

f %	*f* < 4 / OME < 2	OME
3,59	Controle	1,78
2,39	Motora	1,17

f %	*f* < 4 / OME >= 2	OME
1,59	Orientação	2
1,59	Hierarquia	2,5

Tabela 4.13 – Análise de conteúdo Coordenação x Computação (N=86)*

Tema	*f*	%
Coordenação de agentes com objetivos comuns	8	9,1
Coordenação motora	4	4,5
Gestão e questões organizacionais	27	30,7
Não souberam responder	16	18,2
Organização do computador	1	1,1
Organizar ou ordenar dados	13	14,8
Organizar ou ordenar operações e funções	9	10,2
Programação, códigos ou lógicas	3	3,4
Requisito ou propriedade essencial da Computação	7	8,0

*As respostas contêm temas de uma ou mais categorias.

Programadores e não-programadores apresentaram diferenças pouco relevantes na estrutura da representação para o termo *coordenação* (Tabelas 4.12a e 4.12b): programadores posicionam *organização* na primeira periferia, enquanto não-programadores mantêm a hipótese de centralidade da palavra; o núcleo central dos não-programadores inclui *controle* e a zona de contraste deste grupo contém *gerenciamento* e *planejamento* além de *motora*. Em linhas gerais, o termo *coordenação* manteve-se associado a questões gerenciais quando os grupos são considerados separadamente, situação sustentada também pela análise de conteúdo.

Tabela 4.12a – Frequência x Ordem de Evocação – Coordenação (programadores, N=23)

f %	*f* >= 4 / OME < 2	OME
12,31	Gestão	1,88
4,62	Ordem	1,33

f %	*f* >= 4 / OME >= 2	OME
15,38	Organização	2,1
12,31	Liderança	2

f %	*f* < 4 / OME < 2	OME
3,08	Mão	1
3,08	Motora	1,5

f %	*f* < 4 / OME >= 2	OME
3,08	Capacidade	3

Tabela 4.12b – Frequência x Ordem de Evocação – Coordenação (não-programadores, N=63)

f %	*f* >= 4 / OME < 2	OME
18,82	Organização	1,66
5,91	Ordem	1,36
4,84	Gestão	1,89
4,3	Controle	1,88

f %	*f* >= 4 / OME >= 2	OME
6,99	Liderança	2

f %	*f* < 4 / OME < 2	OME
2,15	Motora	1
1,61	Gerenciamento	1,33
1,61	Planejamento	1,67

f %	*f* < 4 / OME >= 2	OME
2,15	Hierarquia	2,5
1,61	Chefiar	2
1,61	Orientação	2,33
1,61	Responsabilidade	2,67

A abordagem da Computação para os processos de coordenação, orientadas à atividade de agentes autônomos tendo em vista um objetivo comum aparece como categoria temática da análise de conteúdo em respostas que mencionaram o paralelismo ou coocorrência de processos com objetivos ou fins comuns; coordenação entre programas, sujeitos ou redes; e a autonomia da máquina para gerir seus recursos de memória e processamento seguindo as instruções dos programas utilizados pelos usuários.

Outros temas bastante recorrentes foram: 1) a Computação como organização e ordenação de dados e informações para serem processados, armazenados e apoiar tomadas de decisão; 2) a organização da operação da máquina, em nos termos das tarefas e comandos a serem realizados; 3) a coordenação como requisito ou pré-requisito da Computação para que seja possível solucionar problemas; 4) a coordenação das regras, lógicas e códigos utilizados nas linguagens de programação.

Programadores mencionaram mais frequentemente em suas respostas (21,74%) o tema da coordenação como *atividade de agentes autônomos com objetivos comuns* que os não-programadores (4,76%), e os dois grupos abordaram a temática da *gestão e questões gerenciais* de forma semelhante (39,13% e 28,57% respectivamente). Apenas o grupo dos não-programadores forneceu respostas que definem o termo coordenação com um

requisito ou propriedade essencial da Computação, ao mesmo tempo em que nenhum programador citou *coordenação motora* nas respostas.

Armazenagem

Algumas palavras evocadas para o termo *armazenagem* (Tabela 4.14) remetem a práticas de armazenamento em geral, não apenas no contexto dos computadores: *guardar*, *espaço*, *estocar* com hipótese de centralidade; *segurança*, *depósito*, e *reserva* na segunda periferia. Os termos próprios da computação aparecem: 1) no limite da frequência mínima para inclusão no núcleo central (*dados*); 2) citados por 13 participantes na primeira periferia (memória); 3) na zona de contraste, fazendo menção a dispositivos de armazenamento (*pendrive*) e ao *computador*; 4) e na segunda periferia, por termos relacionados a aplicações da armazenagem na Computação – *banco de dados*, *segurança*, *nuvem* – e a dispositivos de armazenamento – *harddisk*, ou disco rígido.

Tabela 4.14 – Frequência x Ordem de Evocação – Armazenagem (N=86)

f %	*f* >= 3,95 / OME < 2	OME
10,28	Guardar	1,31
6,72	Espaço	1,65
4,35	Estocar	1,73
3,95	Dados	1,4

f %	*f* >= 3,95 / OME >= 2	OME
5,14	Memória	2

f %	*f* < 3,95 / OME < 2	OME
1,98	Pen Drive	1,8
1,58	Computador	1,75

f %	*f* < 3,95 / OME >= 2	OME
2,77	Banco de Dados	2,14
2,37	Segurança	2,17
2,37	Depósito	2,17
2,37	Hard Disk	2,67
1,98	Nuvem	2,2
1,58	Reserva	2,25

A análise prototípica para o termo *armazenagem* realizada para os programadores e não-programadores em separado apresentou diferenças importantes (Tabelas 4.14a e 4.14b). As evocações dos programadores indicaram hipótese de centralidade para *memória*, ao contrário do posicionamento na primeira periferia tanto para a totalidade dos participantes quanto para os não-programadores. A palavra *espaço*, central tanto para os não-programadores quanto para a totalidade dos participantes apareceu na primeira periferia dos programadores. A zona de contraste deste grupo foi composta por *banco de dados* e *dados*, sugerindo que alguns estudantes que sabem programar pensam a armazenagem também no contexto da sua aplicação na Computação, e não apenas nos dispositivos de armazenamento (*pen drives* e o próprio computador).

Tabela 4.14a – Frequência x Ordem de Evocação – Armazenagem (programadores, N=23)

f %	*f* >= 3,95 / OME < 2	OME
10,77	Guardar	1,41
6,15	Memória	1,5

f %	*f* >= 3,95 / OME >= 2	OME
6,15	Espaço	2,25

f %	*f* < 3,95 / OME < 2	OME
1,98	Banco de dados	1,8
1,58	Dados	1,75

f %	*f* < 3,95 / OME >= 2	OME
2,77	Manter	2,14
2,37	*Hard Disk*	2,17
2,37	Salvar	2,17

Tabela 4.14b – Frequência x Ordem de Evocação – Armazenagem (não-programadores, N=63)

f %	*f* >= 3,95 / OME < 2	OME
10,11	Guardar	1,37
6,91	Espaço	1,46
5,32	Estocar	1,7
4,26	Dados	1,38

f %	*f* >= 3,95 / OME >= 2	OME
4,79	Organização	2,11
4,79	Memória	2,22

f %	*f* < 3,95 / OME < 2	OME
2,66	Pen Drive	1,8
2,13	Computador	1,75

f %	*f* < 3,95 / OME >= 2	OME
2,66	Depósito	2
2,66	Segurança	2,2
2,66	Nuvem	2,2
2,66	Banco de dados	2,6
2,13	*Hard Disk*	2,75

A análise de conteúdo (Tabela 4.15) sobre a relação entre armazenagem e Computação enfatiza a importância do armazenamento de dados como principal tema para os todos participantes, inclusive com ampliação das práticas, contextos e dispositivos envolvidos: guardar grandes quantidades de informações acima da capacidade humana; guardar a informação mesmo quando a energia é desligada; uso de fitas, cartões de memória, banco de dados e outros dispositivos físicos para guardar informações; permanência e retenção de dados e informação; destinação dos arquivos e documentos; armazenamento da minha vida digital.

Tabela 4.15 – Análise de conteúdo Armazenagem x Computação (N=86)*

Tema	*f*	%
Armazenamento como processo	3	3,4
Armazenamento de dados	66	75,9
Armazenamento para uso futuro	7	8,0
Armazenamento versus sustentabilidade	2	2,3
Não souberam responder	5	5,7
Outros	4	4,6

*As respostas contêm temas de uma ou mais categorias.

Sete participantes mencionaram que a Computação se utiliza do armazenamento de forma temporária ou permanente para preservar dados computados e utilizá-los no futuro, como em sistemas de bancos de dados. Dois participantes disseram que o armazenamento na Computação estaria ligado à sustentabilidade, auxiliando na redução do consumo de papel.

Programadores e não-programadores, em suas respostas, abordaram essencialmente o mesmo tema do *armazenamento de dados*, com a categoria *armazenamento versus sustentabilidade* figurando apenas entre os não-programadores.

Avaliação

Quanto ao termo indutor *avaliação*, as palavras evocadas com hipótese de centralidade (Tabela 4.16) remetem a questões de avaliação de desempenho – prova, teste e nota – enquanto *análise* aparece isolada na primeira periferia e *qualidade* na zona de contraste. A segunda periferia reúne palavras que se aproximam da avaliação de desempenho – conhecimento, conceito, verificação, *feedback*, aprendizagem, desempenho, resultado – e no sentido de análise – julgamento, crítica, observação e processo.

Tabela 4.16 – Frequência x Ordem de Evocação – Avaliação (N=86)

***f*%**	***f*>= 4 / OME < 1,9**	**OME**
11,2	Prova	1,32
10,0	Teste	1,76
6,8%	Nota	1,76

***f*%**	***f*>= 4 / OME >= 1,9**	**OME**
4,8	Análise	1,92

***f*%**	***f*< 4/ OME < 1,9**	**OME**
3,2	Qualidade	1,5

***f*%**	***f*< 4 / OME >= 1,9**	**OME**
3,2	Julgamento	2
2,4	Conhecimento	2,17
2%	Conceito	2

Programadores e não-programadores situaram a palavra *prova* no núcleo central, com variações para a posição de *teste* e *nota* (Tabelas 4.16a e 4.16b). A palavra *análise* manteve-se na primeira periferia da estrutura do grupo dos não-programadores, seguindo o posicionamento dos dados processados para a totalidade dos participantes.

Tabela 4.16a – Frequência x Ordem de Evocação – Avaliação (programadores, N=23)

***f*%**	***f*>= 4 / OME < 1,6**	**OME**
14,29	Prova	1,11

***f*%**	***f*>= 4 / OME >= 1,6**	**OME**
9,52	Teste	1,67

***f*%**	***f*< 4/ OME < 1,6**	**OME**
3,17	Conhecimento	1,5

***f*%**	***f*< 4 / OME >= 1,6**	**OME**
3,17	Verificação	2
3,17	Conceito	2
3,17	Seriedade	3

Tabela 4.16b – Frequência x Ordem de Evocação – Avaliação (não-programadores, N=63)

f %	*f* >= 4 / OME < 1,8	OME
10,16	Prova	1,42
10,16	Teste	1,79
8,56	Nota	1,69

f %	*f* >= 4 / OME >= 1,8	OME
5,88	Análise	1,82

f %	*f* < 4/ OME < 1,8	OME
3,74	Qualidade	1,57

f %	*f* < 4 / OME >= 1,8	OME
3,74	Julgamento	2,14
2,14	Conhecimento	2,5

A análise de conteúdo (Tabela 4.17) indica que, no contexto da Computação, os participantes consideram o conceito de avaliação principalmente relacionado ao monitoramento do desempenho do *hardware* ou *software*: supervisão, verificação e validação das operações, performance e do funcionamento do computador; definição de parâmetros e métricas de avaliação; processamento e cruzamento de dados para que se avalie o sistema; teste da qualidade ou eficácia do sistema ou de produtos gerados; avaliação de erros ou problemas visando a melhoria do sistema.

Tabela 4.17 – Análise de conteúdo Avaliação x Computação (N=86)*

Tema	*f*	%
Análise ou verificação de dados	11	12,6
Avaliação de programas, algoritmos ou afins	3	3,4
Avaliação formal, provas e testes acadêmicos	9	9,2
Computador como ferramenta de suporte ao processo de avaliação	12	13,8
Monitoramento do desempenho da máquina	33	37,9
Não respondeu	11	12,6
Outros	9	10,3

*As respostas contêm temas de uma ou mais categorias.

Próxima à essa categoria está a que contempla ideias da avaliação como análise ou verificação computacional de dados: leitura, verificação, compilação, interpretação ou comparação de dados; análise da informação, relatórios ou de valores armazenados. O computador também aparece como ferramenta de apoio ao processo de avaliação acadêmico: correção automática ou facilitada de provas, escolhas de materiais, melhoria das ferramentas de trabalho e avaliação das soluções de problemas por um computador.

A avaliação da qualidade de algoritmos, programas e linguagens de computação apareceu com frequência menor, assim como temas como a preparação para o início do trabalho com o computador, a seleção e conceituação das ferramentas. Os temas abordados pelos participantes tocam em questões importantes dos princípios de Denning e Martell (2007b) para a avaliação, em especial quanto ao monitoramento do desempenho da máquina, das suas operações, análise e verificação de dados.

A categoria temática com maior ocorrência em ambos os grupos foi a de *monitoramento do desempenho da máquina*, com 43,48% entre os programadores e 36,51% entre não-programadores. Somente os participantes deste último grupo relacionaram o tema da avaliação na Computação às *avaliações formais*, *provas e testes acadêmicos*. A utilização da Computação para *análise ou verificação de dados* foi mais citada entre programadores (21,74%) que não-programadores (9,52%), ao contrário do uso da Computação como ferramenta de suporte ao processo de avaliação, ligeiramente mais frequente nas respostas dos não-programadores (15,87% contra 8,7%).

Projeto

Por sua vez, a análise das evocações sobre o termo indutor *projeto* resultou na Tabela 4.18, refletindo o universo particular da atividade de projeto com o qual os estudantes estão familiarizados, não necessariamente restrita à presença da Computação no processo.

Tabela 4.18 – Frequência x Ordem de Evocação – Projeto (N=86)

f %	*f* >= 4 / OME < 1,9	OME
8,0	Planejamento	1,7
6,8	Design	1,29
4,4	Ideia	1,27

f %	*f* >= 4 / OME >= 1,9	OME
5,6	Metodologia	2
4,4	Desenvolvimento	1,91

f %	*f* < 4/ OME < 1,9	OME
2,8	Trabalho	1,86
2,0	Criação	1,2
2,0	Realização	1,6
1,6	Objetivo	1,25
1,6	Etapas	1,75

f %	*f* < 4 / OME >= 1,9	OME
2,8	Produto	2
2,8	Organização	2,14
2,8	Pensamento	2,14
2	Pesquisa	2
2	Resultado	2,8
1,6	Processo	2
1,6	Solução	2,5

Faz-se necessário destacar que as evocações de programadores e não-programadores diferem em pontos importantes, considerando-se os mesmos padrões de frequência e ordem média de evocação. No primeiro grupo o núcleo central seria potencialmente composto por *criação* e *Design,* enquanto a primeira periferia seria formada apenas por *desenvolvimento* (Tabela 4.18a).

As evocações dos participantes não-programadores resultariam em outra configuração do quadro: *planejamento*, *Design*, *ideia* e *desenvolvimento* no núcleo central, seguidos de *metodologia* primeira periferia (Tabela 4.18b).

Tabela 4.18a – Frequência x Ordem de Evocação – Projeto (programadores, N=23)

f %	*f* >= 4 / OME < 1,9	OME
6,06	Criação	1
6,06	Design	1,5

f %	*f* >= 4 / OME >= 1,9	OME
4,55	Desenvolvimento	2

f %	*f* < 4/ OME < 1,9	OME
3,03	Realização	1
3,03	Atividade	1
3,03	Usuário	1,5

f %	*f* < 4 / OME >= 1,9	OME
3,03	Pensamento	2
3,03	Organização	2
3,03	Produto	2
3,03	Planejamento	2,5
3,03	Pesquisa	3
3,03	Fazer	3

Tabela 4.18b– Frequência x Ordem de Evocação – Projeto (não-programadores, N=63)

f %	*f* >= 4 / OME < 1,9	OME
9,78	Planejamento	1,61
7,07	Design	1,23
5,43	Ideia	1,3
4,35	Desenvolvimento	1,88

f %	*f* >= 4 / OME >= 1,9	OME
7,07	Metodologia	1,92

f %	*f* < 4/ OME < 1,9	OME
3,8	Trabalho	1
1,63	Pesquisa	1
1,63	Objetivo	1,5
1,63	Etapas	1,67

f %	*f* < 4 / OME >= 1,9	OME
3,03	Produto	2
3,03	Pensamento	2,2
3,03	Organização	2,2
3,03	Resultado	2,8
3,03	Processo	2
3,03	Solução	2,5

As diferenças entre as três configurações (todos, programadores e não-programadores) incentivam o questionamento da posição da palavra *desenvolvimento* na estrutura da representação. Para o grupo dos não-programadores, a palavra estaria mais próxima de *trabalho* (*d*=0,55), termo de maior frequência e menor OME da zona de contraste. Entre os programadores, desenvolvimento estaria mais distante de *Design*, termo com maior OME do núcleo central (*d*=6,24), e mais próxima de usuário (*d*=1,6), palavra de maior OME da zona de contraste. Tal posição pode indicar que a relação entre *desenvolvimento* e *projeto* seria compartilhada entre a totalidade dos participantes de forma distinta da sugerida pelas zonas da Tabela 4.18.

Avançando na análise de conteúdo das respostas sobre a relação entre o termo projeto e a Computação (Tabela 4.19) junto à discussão sobre as contribuições da Computação para a atividade de projeto realizada na introdução desta tese, pode-se entender que os participantes pensam ou têm algum conhecimento sobre o potencial da Computação e dos computadores tanto nas contribuições para o processo de síntese – imagens de síntese,

Design Generativo e Paramétrico; quanto de análise – identificação e avaliação de variáveis de projeto com auxílio de ferramentas computacionais.

A categoria *projeto como desenvolvimento* aproxima-se da perspectiva analítica de Alexander (1964), incluindo temas como: construção e produção de algo onde a Computação é o próprio meio de projeto; pensar soluções cujas variáveis podem ser computadas; planejar o uso conjunto de programas em sequência ou etapas tendo um objetivo final; programação seria um tipo de projeto. Já a categoria *projeto como Design assistido por computador* se inscreve nos usos tradicionais dos computadores pelos designers descritos por Meggs (1997), oferecendo ferramentas e programas específicos para projetar. A temática do *projeto como planejamento e método* situa o computador, mais do que a Computação, como ferramenta de gerenciamento das etapas de projeto: planejamento, organização, designação, previsão e execução de ações, metas e prazos, visando minimizar erros, desperdícios ou atrasos. Esta categoria relaciona-se aos elementos com hipótese de centralidade, fazendo referência às práticas de projeto que não necessariamente estão restritas ao uso da Computação mas que, pela análise de conteúdo, parecem se beneficiar dela.

Tabela 4.19– Análise de conteúdo Projeto x Computação

Tema	***f***	**%**
Projeto como desenvolvimento	20	22,0
Projeto como Design assistido por computador	16	17,6
Projeto como planejamento e método	29	31,9
Projeto como resolução de problemas	4	4,4
Projeto do próprio computador	2	2,2
Projeto no sentido organizacional e de gestão	3	3,3
Outros	6	6,6
Não respondeu	11	12,1

*As respostas contêm temas de uma ou mais categorias.

No que diz respeito às menções explícitas da Computação como estratégia de design, a categoria *projeto como resolução de problemas* apresenta as únicas menções, a saber: utilização da Computação para analisar dados e resolver problemas; propor a resolução de problemas utilizando processos computacionais; desenvolvimento de uma solução computacional para um problema; usar a Computação para apresentar uma resposta ao problema. As demais categorias agruparam temas diversos e menos relacionados às evocações: projeto do próprio computador, realização de pesquisas no computador para projetar; aplicações para a organização do trabalho do profissional de Design. Onze participantes não disseram não saber responder a relação entre Computação e *projeto*.

A análise de conteúdo das respostas de programadores e não-programadores em separado indica mais semelhanças que diferenças no entendimento da relação entre Computação e *projeto*. As categorias *projeto como desenvolvimento* e *projeto como planejamento e método* são as mais frequentes nas respostas dos dois grupos (programadores: 30,43% e 26,09%; não-programadores: 20,63% e 20,63%). A diferença mais acentuada foi encontrada nas menções ao tema *projeto como Design assistido por computador*, observada nas respostas de 15 participantes não-programadores contra apenas um programador.

Abstração

A palavra *abstração* induziu a evocação de palavras mais próximas do universo das Artes Visuais e do Design do que da Computação (Tabela 4.20). Considerando que o estudante de Design tem contato, de forma geral, com o conceito de abstração ao estudar movimentos artísticos, meios e técnicas de representação gráfica, parece plausível a proximidade na distribuição de frequências e ordem média de evocação das palavras com hipótese de centralidade e da primeira periferia. A abstração, para a totalidade dos participantes, parece referir-se mais ao processo criativo (conceito, imaginação, criatividade, pensamento), de natureza artística, do que ao tipo de abstração simbólica ou de encapsulamentos em diferentes níveis de complexidade que seriam essenciais na Computação (Wing, 2008a).

Tabela 4.20 – Frequência x Ordem de Evocação – Abstração (N=86)

f %	*f* >= 2,8 / OME < 2	OME	*f* %	*f* >= 2,8 / OME >= 2	OME
4,07	Arte	1,8	3,25	Criatividade	2,13
4,07	Conceito	1,9	2,85	Pensamento	2,14
2,85	Imaginação	1,86			

f %	*f* < 2,8/ OME < 2	OME	*f* %	*f* < 2,8 / OME >= 2	OME
2,03	Liberdade	1,8	2,03	Ideia	2,2
2,03	Simplificação	1,8	1,63	Distração	2,75
1,63	Subjetivo	1,5			

Em relação às diferenças entre programadores e não-programadores, as evocações induzidas pelo termo *abstração* geraram uma situação particular: os participantes apresentaram diferenças individuais muito fortes em relação aos elementos da representação, evocando termos com OME e frequência muito baixas. Esse fato impossibilitou a constituição de uma estrutura coerente a partir dos dados que pudesse ser interpretada em separado.

Por outro lado, a análise de conteúdo sobre a relação entre *abstração* e Computação (Tabela 4.21) contribui para sustentar a interpretação das evocações na temática do processo criativo (Tabela 4.22), embora 17 participantes disseram não saber responder.

Tabela 4.21 – Análise de conteúdo Abstração x Computação (N=86)*

Tema	*f*	%
Abstração como encapsulamento da complexidade	17	19,3
Abstração como dificuldade de entendimento	6	6,8
Abstração como subjetividade e criação	27	30,7
Abstração como interpretação, tradução ou mapeamento	8	9,1
Abstração não se relaciona com Computação	5	5,7
Abstrato em oposição ao que é concreto	1	1,1
Não respondeu	17	19,3
Outros	7	8,0

*As respostas contêm temas de uma ou mais categorias.

Tabela 4.22 – Análise de conteúdo Abstração x Computação: exemplos de respostas por tema

Temas	Programadores	Não-programadores
Abstração como subjetividade e criação	Formulação de um conceito. Liberdade criativa, sugestão, a parte livre onde as ideias surgem. o limiar da criação. Interpretação criativa de determinado fenômeno.	Construção de projeto no campo da imaginação que dá forma ao projeto. Não levar em consideração algum cálculo. "Desracionalização" e fuga da lógica, desvio para outra coisa e desprendimento da racionalização ou ferramenta específica.
Abstração como encapsulamento da complexidade	[...] enxergar uma determinada situação (o funcionamento de um programa, por exemplo) por uma composição de 'engrenagens', que enquanto não são foco de estudo são apenas 'engrenagens', compondo um sistema maior e podendo até ser repetidas ao longo do sistema, mas que quando necessário podem ser investigadas sob um olhar mais detalhado, seguindo a ideia que mesmo essas engrenagens podem (e são) compostas por engrenagens menores, progressivamente. Camadas de codificação que tem como objetivo esconder processos computacionais mais complexos.	A interface pode ser vista como uma abstração dos códigos de um computador. Você abstrai que existem programações complexas e só usa a interface. Conceitos envolvidos, mas não tão explícitos. [...] É você esquecer que uma fotografia é composta por pixels e só se focar no que a imagem mostra etc.
Abstração como interpretação, tradução ou mapeamento	Entendo como o ato de sair do específico e tornar geral. É pegar um exemplo e formar um conceito a partir dele. na computação, é entender como a solução se aplica a cenários diferentes, mesmo sendo produzida para um cenário específico.	Abstração seria pegar tudo o que foi projetado e representar de outra maneira. o movimento do mouse, por exemplo, são coordenadas x e y na tela. Retirar conceitos dos resultados práticos.

Ainda segundo a análise de conteúdo, alguns estudantes teriam uma concepção mais próxima da descrita por Wing (2008a). Do total dos estudantes, 13 não-programadores e quatro programadores mencionaram questões relevantes para tal concepção. Entre os dois grupos, as diferenças nas respostas aparecem principalmente quanto à temática da *abstração como encapsulamento da complexidade* (Tabela 4.22): os programadores abordaram diretamente a abstração como processo que esconde a complexidade, enquanto os não-programadores citaram também mudanças de pontos de vista, interpretações, traduções ou mapeamentos de operações de um nível de análise para outro, e a abstração entre o código e o resultado que produz. As demais categorias correspondem aos seguintes temas: há pouca ou nenhuma relação entre Computação e abstração; abstração refere-se à dificuldade de entendimento da própria Computação e seus processos; abstração entendida em oposição à concretude.

Automação

O termo *automação* (Tabela 4.23) induziu as palavras *automático* e *máquina* com hipótese de centralidade, com *robótica*, *tecnologia* e *praticidade* na primeira periferia. A zona de contraste foi formada pelas evocações *carro* e *computador*, enquanto a segunda periferia por *programação*, *rapidez*, *simplicidade*, *autonomia* e *mecanismos*.

Tabela 4.23 – Frequência x Ordem de Evocação – Automação (N=86)

***f*%**	***f*>= 3 / OME < 1,8**	**OME**
5,37	Automático	1,46
4,96	Máquina	1,75

***f*%**	***f*>= 3 / OME >= 1,8**	**OME**
9,5	Robótica	1,87
4,55	Tecnologia	1,82
3,72	Praticidade	1,89

***f*%**	***f*< 3/ OME < 1,8**	**OME**
1,65	Carro	1,25
1,65	Computador	1,25

***f*%**	***f*< 3 / OME >= 1,8**	**OME**
2,48	Programação	2,17
2,48	Rapidez	2,33
1,65	Simplicidade	2
1,65	Autonomia	2
1,65	Mecanismos	2,25

As evocações dos não-programadores sobre o termo *automação* geraram uma distribuição dos termos nas zonas da Tabela 4.23a semelhante à da totalidade dos participantes. Em contrapartida, as evocações dos programadores consideradas em separado apresentaram a mesma dificuldade de interpretação do termo indutor *abstração*. As diferenças individuais, marcadas principalmente pela evocação de exemplos de situações, objetos ou aplicações da automação, dificultaram a composição de uma estrutura coerente.

Tabela 4.23a – Frequência x Ordem de Evocação – Automação (não-programadores, N=63)

f %	*f* >= 3 / OME < 1,8	OME
7,22	Automático	1,46
6,67	Máquina	1,75

f %	*f* >= 3 / OME >= 1,8	OME
11,67	Robótica	1,95
4,44	Tecnologia	2

f %	*f* < 3/ OME < 1,8	OME
2,22	Computador	1,25
2,22	Carro	1,25
1,67	Facilidade	1,67

f %	*f* < 3 / OME >= 1,8	OME
2,78	Programação	2,2
2,22	Praticidade	2
2,22	Mecanismos	2,25
1,67	Repetição	2
1,67	Desemprego	2,33
1,67	Autonomia	2,33
1,67	Rapidez	2,67

A análise de conteúdo (Tabela 4.24) sugere alguns caminhos em potencial para a compreensão dos processos de ancoragem e objetivação dos elementos das representações sobre a *automação*: trata-se de uma propriedade ou processo que confere autonomia às máquinas (computadores ou outras); otimizando, facilitando, simplificando ou agilizando a realização de tarefas originalmente realizadas pelo trabalho manual e repetitivo do homem; que requer algum grau de programação de rotinas, sequências ou repetições de ações, tendo como consequência a possibilidade de se dispensar a supervisão do homem.

Tabela 4.24 – Análise de conteúdo Automação x Computação (N=86)*

Tema	*f*	%
Automação como processo	22	20,0
Automação como propriedade	14	12,7
Automação como repetição ou sequência	2	1,8
Automação facilita a vida	6	5,5
Funcionamento autônomo do computador	34	30,9
Otimização e redução de tempo	11	10,0
Robótica	3	2,7
Substituição ou mecanização do trabalho	15	13,6
Outros	3	2,7

*As respostas contêm temas de uma ou mais categorias.

Programadores e não-programadores mencionaram os temas da Tabela 4.24 com frequências semelhantes em suas respostas, com duas exceções: apenas programadores citaram a relação da Computação com a automação como *repetição ou sequência* e nenhum programador citou a robótica para a referida relação.

A hipótese de centralidade da palavra *máquina* e a presença de *robótica* com alta frequência na primeira periferia, portanto, podem indicar a necessidade de que a

automação, para os 86 participantes da pesquisa, seja uma propriedade daquilo que é mecânico ou fisicamente implementado em algum sistema. Carros e computadores, que figuraram na zona de contraste, são exemplos contemporâneos de máquinas autônomas recorrentemente discutidas na mídia[44].

Embora o conceito de *automação* de Wing (2008a) trate da mecanização de processos (abstrações), não há exigência quanto à sua realização exclusiva por máquinas. Knuth (1997) lembra que uma das características importantes dos algoritmos é a possibilidade de ser realizado por um ser humano com papel e lápis.

Algoritmo

Por fim, as evocações sobre o termo *algoritmo* (Tabela 4.25) encerram a análise prototípica indicando hipótese de centralidade para as palavras *matemática* e *números*, com *programação* e *código* na primeira periferia, *cálculo* e *regras* na zona de contraste. Esse arranjo das distribuições de frequências e ordens médias de evocação apontam para dois temas importantes que dariam suporte à ancoragem e objetivação dos elementos das representações dos participantes da pesquisa sobre a noção de algoritmo. Primeiramente, algoritmos pertenceriam ao universo da matemática (especificamente relacionados a cálculo, para alguns), materializados por números. Em segundo lugar os algoritmos estariam frequentemente, embora não essencialmente, ligados à programação, com sua realização pelos códigos. Nessa direção, parece relevante notar que as evocações dos participantes tenham se agrupado em áreas específicas de conhecimento (matemática e programação) e em suas respectivas ferramentas (números e código).

Tabela 4.25 – Frequência x Ordem de Evocação – Algoritmo (N=86)

***f* %**	***f* >= 4 / OME < 1,7**	**OME**
15,79	Matemática	1,67
10,53	Números	1,46

***f* %**	***f* >= 4 / OME >= 1,7**	**OME**
8,91	Programação	1,95
5,26	Código	1,92

***f* %**	***f* < 4/ OME < 1,7**	**OME**
3,24	Cálculo	1,63
1,62	Regras	1,25

***f* %**	***f* < 4 / OME >= 1,7**	**OME**
3,24	Lógica	2,38
2,43	Complexo	2,33
2,02	Instrução	2
2,02	Computação	2,4
1,21	Passo a passo	2
1,21	Computador	2,67
1,21	Tarefa	2,67

[44] O automóvel autônomo desenvolvido pelo Google é um exemplo dessa discussão que tem permeado a mídia. Disponível em http://tinyurl.com/carro-google. Acesso em 21 de novembro de 2013.

Ao considerar apenas as evocações dos estudantes que se disseram programadores, *matemática* mantem-se com hipótese de centralidade, mas *código* desloca-se da primeira periferia para o núcleo central com frequência e OME semelhantes ao termo anterior A primeira periferia apresenta configuração distinta, composta por *instrução* e *tarefa*. Os termos *cálculo* e *regras* continuam na zona de contraste, que também passa a incluir *programação*, *sequência* e *receita* com baixa ordem média de evocação.

Tabela 4.25a – Frequência x Ordem de Evocação – Algoritmo (programadores, N=23)

f %	*f* >= 4 / OME < 1,7	OME
7,69	Matemática	1,4
7,69	Código	1,6

f %	*f* >= 4 / OME >= 1,7	OME
4,62	Instrução	2,33
4,62	Tarefa	2,67

f %	*f* < 4/ OME < 1,7	OME
3,08	Sequência	1
3,08	Programação	1
3,08	Cálculo	1,5
3,08	Receita	1,5
3,08	Regras	1,5

f %	*f* < 4 / OME >= 1,7	OME
3,08	Passo a passo	2,5
3,08	Lógica	2,5
3,08	Números	2,5
3,08	Comandos	3

Numa situação inversa, onde apenas as evocações dos não-programadores são analisadas, a configuração se mantém praticamente a mesma nas quatro zonas (Tabela 4.25b): *matemática* e *números* no núcleo central; *programação* e *código* na primeira periferia. Em outras palavras, os participantes programadores parecem divergir de não-programadores quanto à centralidade de *código* e *número* nos elementos da representação de algoritmo, além dos primeiros valorizarem frequentemente apesar de menos intensamente as relações do termo com *instruções* e *tarefas*, ao invés de *programação*.

Tabela 4.25b – Frequência x Ordem de Evocação – Algoritmo (não-programadores, N=63)

f %	*f* >= 4 / OME < 1,8	OME
18,68	Matemática	1,71
13,19	Números	1,38

f %	*f* >= 4 / OME >= 1,8	OME
10,99	Programação	2,05
4,4	Código	2,13

f %	*f* < 4/ OME < 1,8	OME
3,08	Sequência	1
3,08	Programação	1
3,08	Cálculo	1,5
3,08	Receita	1,5
3,08	Regras	1,5

f %	*f* < 4 / OME >= 1,8	OME
3,3	Lógica	2,33
2,75	Complexo	2,2
2,2	Computação	2,25

Retornando às evocações de todos os participantes, os termos c*omputador* e *computação* apareceram apenas na segunda periferia, próximos a outros como *lógica*, *instrução* e *passo a passo*, que pertenceriam à mesma temática de *regras* da zona de contraste, e *tarefa*. A baixa frequência desses termos pode sugerir que os estudantes entrevistados não

relacionam diretamente algoritmos a computadores, numa concepção oposta das evocações sobre *automação* que pareceram vincular tais processos a um meio físico capaz de realiza-los. A análise de conteúdo sobre as relações entre a Computação e os algoritmos (Tabela 4.26) ajuda a esclarece essas questões.

Tabela 4.26 – Análise de conteúdo Algoritmo x Computação (N=86)*

Tema	***f***	***%***
Algoritmo como linguagem	10	11,0
Algoritmo como Matemática	13	14,3
Algoritmo como matéria-prima da Computação	23	25,3
Algoritmo como programação	20	22,0
Algoritmo como receita	18	19,8
Outros	2	2,2
Não respondeu	5	5,5

*As respostas contêm temas de uma ou mais categorias.

As categorias com o maior número de respostas foram as que agruparam ideias que definem os algoritmos como matérias-primas da Computação, como um tipo de programação ou como receita para programar os computadores. Os conteúdos referentes à dimensão matemática dos algoritmos relacionam-se à dificuldade de compreensão ou à precisão e ordem como características importantes.

Filtrando as categorias da análise de conteúdo pelo perfil do participante, a concepção do *algoritmo como matéria-prima da Computação* é mais frequente entre os não-programadores (28, 75%) que entre os programadores (21,74%), assim como como a categoria *algoritmo como matemática* (19,05% e 4,35%, respectivamente) e *algoritmo como programação* (25,40% e 17,39%). Apenas a ideia do *algoritmo como receita* é ligeiramente mais frequente entre os programadores (43,48%) que os não-programadores (12,70%). Nesta última categoria, os programadores mencionam principalmente passos ou sequências de instruções e tarefas, fundadas em alguma lógica ou regra de operação em tendo em vista um resultado. Entre os não-programadores, a temática da sequência de instruções também é comum, porém com citações mais explícitas de que tais sequências ocorreriam em um computador ou seriam relacionadas à Computação. A Tabela 4.27 fornece exemplos dessas respostas:

Tabela 4.27 – Análise de conteúdo Algoritmo x Computação: exemplos de respostas por tema

Temas	Não-programadores	Programadores
Algoritmo como matéria-prima da Computação	Faz tudo acontecer. Procedimento pelo qual um processo computacional é submetido para ter resultados. É um pedaço de lógica computacional que executa função específica. Pode ser um código ou pode ser só uma ideia que será transformada em código posteriormente.	Matéria-prima da computação, como se fosse a menor unidade da computação. Unidade mínima (essencial) integrante do projeto. O algoritmo é a ferramenta principal para computação. Usado para fazer todas as coisas acima (Princípios da Computação) funcionarem.
Algoritmo como programação	Linguagem de programação utilizada em um computador. Relacionado com o código, necessário para alguns comandos. Necessário para programação dentro do computador.	É a linguagem do computador - por meio de algoritmos é possível escrever um código operacional, ou seja, mandar o computador realizar uma determinada tarefa. Utilizado no momento da programação.
Algoritmo como receita	Regras, receitas das milhares de operações de um PC. Seria um conjunto de instruções ordenadas com dados computáveis. Sequência lógica, finita e definida de instruções.	Guia, receita de bolo, essencial. Uma sequência de tarefas que uma máquina computacional segue para realizar uma determinada ação. Série de regras ordenadas.
Algoritmo como Matemática	Matemática, chatice que ninguém suporta! Função matemática que coordena o computador. Cálculos diversos, exatidão e precisão.	Utilização de dados matemáticos para solucionar problemas.
Algoritmo como linguagem	Linguagem de programação utilizada em um computador. Linguagem matemática para computação. Os algoritmos de alguma forma estão muito ligados a computadores, como uma linguagem está ligada às pessoas.	Um protocolo de linguagem que padroniza comandos, uma cadeia lógica de comandos. É a linguagem do computador - por meio de algoritmos é possível escrever um código operacional, ou seja, mandar o computador realizar uma determinada tarefa.

Pode-se argumentar que os estudantes programadores compartilhariam uma concepção dos algoritmos menos dependente dos meios físicos de execução e mais relacionada à organização e sequenciamento de instruções tendo em vista uma tarefa a ser realizada. Tal argumento aproxima-se das questões apresentadas por Futschek (2006) sobre as aplicações do pensamento algorítmico nos processos de resolução de problemas – análise e especificação do problema, busca pelas ações básicas adequadas, construção do algoritmo correto para solucionar o problema – bem como das características listadas por Knuth (1997): o algoritmo é uma sequência ou passo a passo de instruções que visa computar dados para resolver um determinado problema.

4.5 – Teste de centralidade

As evocações induzidas pelos termos comuns às diferentes perspectivas do Pensamento Computacional (Tabela 4.28) tiveram suas hipóteses de centralidade investigadas junto a 20 estudantes com o intuito de esclarecer as dúvidas encontradas na análise prototípica.

Tabela 4.28 – Evocações com hipótese de centralidade por termo indutor

Termo	Evocações
Computação	Computador, Programação e Tecnologia
Algoritmo	Código, Programação, Matemática e Números
Abstração	Arte, Conceito, Criatividade, Imaginação e Pensamento
Automação	Automático, Máquina, Praticidade, Robótica e Tecnologia

Computação

No teste do primeiro conjunto de evocações, 14 dos 20 participantes indicaram que os computadores seriam muito importantes para o que eles entendem por Computação. No entanto, apenas dez desses relataram não conseguir pensar na Computação sem os computadores, utilizando argumentos como: as palavras possuem a mesma origem; todas as coisas que computam são computadores; tudo na Computação tem relação direta com os computadores; Computação e computadores são integrados ou um depende do outro; todo trabalho feito em Computação depende de um computador.

Tabela 4.29– Atribuição de muita importância e *mise en cause* para Computação (N=20)

Evocação	*f*	Conseguem pensar em Computação sem pensar no termo	
		Não	Sim
Computador	14	10	4
Programação	17	9	8
Tecnologia	17	15	2

Os outros quatro estudantes, que discordaram da indissociabilidade entre computador e Computação, forneceram justificativas baseadas em aplicações variadas dos princípios computacionais que independem das máquinas, como aquelas encontradas nas categorias da análise de conteúdo da Tabela 4.8[45]. Partindo desses resultados não é possível afirmar sem dúvidas a centralidade dos computadores na representação dos estudantes sobre a Computação, mas há contribuições importantes desse equipamento nos processos de ancoragem (as palavras possuem a mesma origem; tudo na Computação tem relação direta com computadores) e objetivação (tudo o que computa é um computador). Vale

[45] Ver p.84

lembrar que na análise prototípica das evocações sobre os Princípios da Computação, o computador é o principal meio físico citado pelos participantes, o que pode reforçar a centralidade do elemento numa perspectiva mais abrangente que a desse teste.

A *programação* foi indicada por 17 participantes como sendo muito importante para o entendimento deles sobre a Computação. Apenas nove deles afirmaram que não conseguiriam pensar nas duas ideias em separado, recorrendo a justificativas como: a Computação sempre tem programação; são praticamente sinônimos ou análogos; até os meios mais simples da Computação precisam de programação; os processos do computador sempre envolvem programação. Na situação inversa, os oito participantes que refutaram a relação estreita entre programação e Computação explicaram que são usuários dos computadores e ainda assim não programam ou conseguem manter a programação fora das suas atividades.

Retomando o quadro das atividades cotidianas dos estudantes de Design com os computadores (Tabela 4.3[46]) e combinando-o às respostas do teste, parece plausível entender que a programação desempenha um papel periférico na estrutura da representação. Trata-se de uma das formas de realizar a Computação no cotidiano dos estudantes, mas por não ser praticada por eles pode ter uma função menos importante para o funcionamento do sistema central da representação.

O termo *tecnologia* foi o que teve o maior número de indicações de muita importância para o entendimento da Computação pelos participantes do teste (17), bem como o de confirmações da atribuição dessa importância pela técnica de *mise en cause* (15). Para esses estudantes: seria impossível não relacionar ou não associar Computação e tecnologia; uma depende da outra; a Computação é um exemplo de tecnologia ou é tecnologia; a tecnologia está em tudo; a evolução tecnológica permitiu a evolução da Computação. Os dois participantes que refutaram a relação de muita importância entre os termos afirmaram que os meios da Computação não estão relacionados com a tecnologia ou que esta não está necessariamente atrelada à primeira. Considerando que o termo tecnologia foi posicionado na primeira periferia na análise prototípica (Tabela 4.7[47]), as justificativas dos participantes no teste introduzem pelo menos duas questões: 1) a tecnologia poderia estaria ligada aos elementos centrais da representação sobre a Computação por meio de transformações periféricas que seriam percebidas como fontes de mudanças irreversíveis no núcleo, no conforme exposto por Flament (2001); 2) seria

[46] Ver p.77
[47] Ver p.83

necessário realizar uma investigação no sentido inverso, com o intuito de compreender como os computadores e a Computação integram a estrutura representacional do conceito de tecnologia elaborada pelos estudantes.

Algoritmo

Os testes de centralidade para as evocações do termo indutor *algoritmo* (Tabela 4.30) ajudaram na compreensão das diferenças na participação das áreas de conhecimento mencionadas pelos participantes (matemática e programação) das suas respectivas ferramentas (código e números). O mesmo teste ainda possibilitou elucidar como o elemento código se relaciona com a representação de algoritmo elaborada pelos participantes da pesquisa.

Tabela 4.30 – Atribuição de muita importância e *mise en cause* para Algoritmo (N=20)

Evocação	*f*	Conseguem pensar em Algoritmo sem pensar no termo indutor	
		Não	Sim
Matemática	17	16	1
Números	13	10	3
Programação	18	10	8
Código	14	11	3

Matemática obteve 16 confirmações com apenas uma indicação de muita importância refutada, baseadas em justificativas coerentes com a análise de conteúdo da Tabela 4.26[48]: algoritmos são expressos em termos matemáticos; o estudo dos algoritmos no ensino médio era um conteúdo da matemática; algoritmos envolvem pensamento ou raciocínio lógico-matemático; são essencialmente formados por cálculos e lógicas; tudo em programação envolve ou está diretamente associado a matemática; não sei como ler um algoritmo sem a linguagem matemática.

Quanto a *números*, a interpretação realizada a partir da análise prototípica parece se sustentar pelos argumentos dos participantes. Para os dez estudantes que não conseguem pensar em algoritmos sem pensar em números, algoritmos lembram matemática e então lembram números ou estão associados a matemática e portanto a números; algoritmos geralmente envolvem números, mesmo que nos resultados. Para os que disseram conseguir pensar os dois termos em separado ou que indicaram que a ligação entre eles teria pouca importância, o aspecto ferramental dos números no contexto da matemática e dos algoritmos fica mais evidente. Seguem alguns exemplos:

[48] Ver p.104

> [...] além dos números, imagino diversas possibilidades de letras, grafemas, símbolos gráficos que podem representar o algoritmo. (*muita importância, refutação*)
>
> [...] os algoritmos não dependem de números para existirem. (*pouca importância, confirmação*)
>
> [...] são regras que não necessariamente precisam de números. apesar de que acho que ficam mais claras (pra máquina) as regras com números. (*muita importância, refutação*).
>
> [...] associo algoritmos a lógica, e não necessariamente essa lógica deve ser com números. (*muita importância, refutação*).

Quanto ao teste de *programação*, dos 18 participantes que indicaram que o termo seria muito importante para seu entendimento dos algoritmos, apenas 10 mantiveram tal posição na técnica de questionamento. As respostas que refutaram a relação direta entre os termos foram semelhantes às fornecidas para o questionamento da relação entre Computação e programação – é possível estudar ou desenvolver algoritmos sem saber programar – enquanto as que confirmaram a relação mencionaram que o algoritmo seria um conjunto lógico a ser programado; o algoritmo seria a unidade da programação; algoritmos são mencionados quando se aprende a programar ou é o que se faz quando se programa.

No que diz respeito ao termo *código*, tanto as justificativas que confirmaram (11) ou refutaram (três) a interdependência com a ideia de algoritmo são similares às coletadas para o teste de *programação*:

> A relação que conheço é advinda da matemática, portanto penso em números e não na palavra código. (*muita importância, refutação*)
>
> [...] porque na minha cabeça [código] é a forma de fazer a notação do algoritmo. (*muita importância, confirmação*)
>
> [...] acredito que em algum nível, todos os códigos usam alguma variação de algoritmos. (*muita importância, confirmação*).
>
> [...] acho que o algoritmo depende de um código padrão, que será usado para compor o algoritmo. Não consigo imaginar um algoritmo que não tenha um código trabalhando em segundo plano. (*muita importância, confirmação*)

> [...] porque para mim algoritmo se relaciona a pensamento lógico e código se relaciona com decoreba. (*pouca importância, confirmação*)
>
> Código é um termo abrangente e pouco tem a ver com algoritmo. (*pouca importância, confirmação*).

O conjunto das respostas para o teste das evocações referentes ao termo indutor *algoritmo* reforça a posição de *matemática* e *números* no núcleo central, sendo o segundo mais uma possibilidade de realização prática do primeiro. Tal situação poderia ser entendida como um indício de posicionamento periférico para *números*, uma vez que *código* e *programação* aparecem na técnica de questionamento cumprindo papéis semelhantes. Ainda assim, a baixa OME para as evocações de *número* (Tabela 4.25[49]) pode indicar que, dentre as formas de expressão percebidas para os algoritmos, os números seriam consideradas as mais importantes pelos participantes da pesquisa.

Abstração

O teste de centralidade para as evocações do termo indutor *abstração* (Tabela 4.31) sugeriu que a palavra arte pode ser considerada pelos estudantes de Design como algo menos central na estrutura representacional daquele termo. Das dez confirmações de muita importância, apenas cinco se mantiveram após a aplicação da técnica de questionamento, com base em argumentos ligados à formação acadêmica dos estudantes: o abstracionismo ou movimento abstracionista são importantes ou pertinentes para a área de Design; a arte seria a representação de algo e toda representação seria uma abstração.

Tabela 4.31 – Atribuição de muita importância e *mise en cause* para Abstração (N=20)

Evocação	*f*	Conseguem pensar em Abstração sem pensar no termo indutor	
		Não	Sim
Arte	10	5	5
Conceito	13	8	5
Criatividade	13	11	2
Imaginação	17	16	1
Pensamento	18	16	2

As respostas que refutaram a alta importância de *arte* para a o entendimento do conceito de *abstração* citaram que nem tudo que é abstrato seria arte; abstrair também poderia ser ocultar; arte está ligada a abstração mas a recíproca não é necessariamente verdadeira;

[49] Ver p.102

nem tudo que é abstrato é arte; abstração pode estar ligada a explicações conceituais sem envolver arte.

As respostas do teste de centralidade relativas às demais evocações parecem endossar a independência entre o conceito de abstração e arte para os estudantes, qualificando a interpretação dos resultados da análise prototípica da Tabela 4.20[50]. *Conceito*, *criatividade*, *imaginação* e *pensamento* parecem mais relevantes para o entendimento dos estudantes acerca da ideia de *abstração*, principalmente porque esta parece ser ancorada em processos imaginativos ou criativos, principalmente mentais:

Tabela 4.32 – Exemplos de respostas da técnica *mise en cause* para Abstração

Evocação	**Exemplos confirmatórios da importância**
Conceito	A abstração depende de um conceito
	O conceito em si já é pura abstração.
	Abstrair é isolar alguma coisa, algum conceito.
	A abstração passa pela necessidade de um conceito.
Criatividade	A criatividade é alcançada por meio da abstração.
	A abstração é um exercício de criatividade.
	[...] É preciso criatividade para ser abstrato.
	A criatividade é alimentada pela abstração [...].
Imaginação	Por mais que o conceito de abstração seja um tanto vago para mim, ele se aproxima muito mais de uma imaginação, observação, entendimento de algo, visto à distância.
	Não existe abstração sem imaginação.
	Imaginar já é um exercício de abstração.
	A imaginação é o ambiente em que a abstração é liberada de sua forma oculta de questionamento.
	A imaginação é o processo pessoal de criar, e a abstração é como um ingrediente essencial para a imaginação.
Pensamento	Considero pensamento no mesmo processo de imaginação, questionamento etc.
	Acredito que o pensamento é o meio pelo qual algo concreto se torna abstrato.
	Os pensamentos são por si só coisas abstratos. Eles só passam a existir quando a pessoa os coloca em prática.
	A abstração é um pensamento.
	Vejo o processo de abstração como algo diretamente ligado ao pensamento, ao raciocínio, lógico ou não, sendo um processo do intelecto.

Colocando de outra forma, o teste de centralidade para as evocações do termo *abstração* permite um outro entendimento daquela estrutura representacional, fundado em uma importância em termos de frequências de menções para a palavra *arte*, provavelmente em decorrência da relevância dela na formação acadêmica dos estudantes, mas que não se mantém quando os processos de objetivação e ancoragem são considerados. A *abstração*

[50] Ver p. 98

parece ser objetivada como um *pensamento* ou *faculdade imaginativa*, com qualidades que auxiliam tanto na *criação* quanto na interpretação de *conceitos*. Essas ideias podem ou não estar ancoradas em conhecimentos que os estudantes teriam acumulado sobre a História da Arte ou dos movimentos abstracionistas. De uma forma ou de outra, não parece possível restringir o entendimento do conceito de abstração elaborado pelos participantes do teste ao universo temático das artes.

Automação

Por fim, o teste de centralidade para as evocações induzidas pelo termo *automação* (Tabela 4.33) indicou confirmação da importância das palavras *automático*, *praticidade*, *robótica* e *tecnologia* para a representação elaborada pelos estudantes, enquanto a importância de *máquinas* foi mantida apenas por quatro estudantes.

Tabela 4.33– Atribuição de muita importância e *mise en cause* para Automação (N=20)

Evocação	***f***	**Conseguem pensar em Automação sem pensar no termo indutor**	
		Confirmada	**Refutada**
Automático	14	13	1
Praticidade	13	10	3
Máquinas	5	4	1
Robótica	14	10	4
Tecnologia	15	14	1

Embora as análises prototípica e de conteúdo (Tabelas 4.23 e Tabela 4.24[51]) tenham indicado que a necessidade de realização física do processo de *automação* estava presente no discurso dos estudantes, as justificativas para a pouca importância atribuída às máquinas na técnica do *mise en cause* relataram que pessoas ou outros dispositivos não-mecânicos também poderiam ser utilizados para automatizar processos.

Nesse sentido, o teste das demais evocações ofereceu subsídios relevantes para a compreensão das representações elaboradas pelos estudantes sobre a *automação*. Em relação à palavra *automático*, observou-se que: 1) a etimologia comum da palavra manteve-se como principal justificativa para a interdependência entre esta evocação e o termo indutor; 2) a automação é o processo de tornar algo automático, que funciona sozinho ou sem supervisão humana. Sobre a evocação *praticidade,* a automação figura como um processo que confere aumento na facilidade, eficiência ou redução da necessidade de interferência em processos. Por outro lado, dois estudantes que atribuíram

[51] Ver p.100 e 101

pouca importância para *praticidade* em relação ao conceito de *automação* lembraram precisamente de situações onde a automação gera complexidade, em decorrência da incapacidade de compreensão do funcionamento de algum equipamento autônomo.

Já a palavra *tecnologia* foi considerada muito importante por 14 estudantes, que mantiveram tal avaliação na técnica do questionamento empregando justificativas como: a automação não é possível sem tecnologia ou é sinônimo de tecnologia; os avanços da tecnologia permitem avanços na automação; os termos pertencem à mesma temática; automatizar é usar tecnologia. Quanto à palavra *robótica*, parece consenso entre os participantes da pesquisa de que esta se trata de um dos ramos de aplicação da *automação* ou vice-versa, mencionando exemplos como pilotos-automáticos e robôs em fábricas. Retomando a baixa importância atribuída à palavra *máquinas,* pode-se sugerir que os participantes objetivam a *automação* em robôs e outros tipos de equipamentos autônomos que substituem diretamente o trabalho humano, ao invés de máquinas menos independentes. Essa interpretação sustentaria a reconfiguração da estrutura representacional do termo *automação* (Tabela 4.23[52]), deslocando a ideia de *máquina* para a primeira periferia e mantendo apenas *automático* no núcleo central. Assim, *máquinas*, *robôs*, *tecnologias* e *praticidade* seriam elementos percebidos como manifestações concretas daquilo que é automático no cotidiano dos estudantes de Design.

4.6 – Segunda síntese: RS da Computação dos estudantes de Design

As ideias compartilhadas pelos participantes da pesquisa corroboram os resultados da primeira síntese, ao reforçarem a importância das contribuições dos usos cotidianos dos computadores – pessoais, acadêmicos e profissionais – nos processos de objetivação e ancoragem (Tabela 4.34) das RS acerca dos Princípios da Computação. Partindo dos resultados das análises prototípica e de conteúdo, qualificados pelo teste de centralidade, pode-se propor as seguintes asserções:

- A Computação é realizada em um meio físico (máquina, equipamento ou dispositivo) para acontecer e o principal desses meios é o computador.
- A Computação é um meio de comunicação, mas também um meio ou linguagem de expressão e uma ferramenta de auxílio à resolução de problemas de projeto.
- As RS acerca dos Princípios da Computação parecem ser ancoradas pelos estudantes nas suas atividades cotidianas com os computadores, objetivando os

[52] Ver p.100

conceitos em aplicações, dispositivos, equipamentos ou situações concretas que auxiliam na familiarização com aqueles princípios.

- Os elementos das RS elaboradas por programadores e não-programadores diferem principalmente quanto à sua abrangência: os primeiros parecem ter uma visão mais ampliada das possibilidades de aplicação dos Princípios da Computação, inclusive com independência relativa do computador enquanto meio de realização. Os demais objetivam a Computação no computador, ancorando os princípios da primeira nos recursos e aplicações que lhes são familiares a partir da experiência de uso do segundo.
- A Computação dispõe de um conjunto específico de saberes a serem aprendidos e esses saberes estão relacionados à matemática e aos números, bem como à programação e aos códigos.
- A Computação pode ser um meio de projeto, tanto no planejamento quanto na execução das atividades do designer. A execução ainda é principalmente realizada com o auxílio de programas, mas os estudantes têm ciência de as possibilidades de utilizar outras estratégias computacionais.
- A Computação seria um exemplo de tecnologia, que dependeria dos avanços de outras tecnologias para se desenvolver, ao mesmo tempo em que seus próprios avanços contribuiriam para o progresso em áreas como a robótica.
- O processo de automação estaria relacionado a operações e tarefas realizadas por dispositivos com algum grau de autonomia, enquanto o de abstração estaria relacionado ao pensamento, à imaginação e à criatividade dos seres humanos.

Tabela 4.34 – Processos de objetivação e ancoragem dos Princípios da Computação

Princípio	Ideias sobre as quais se ancora	Objetos e imagens que lhe conferem concretude
Computação	Computador como produto da Computação	Computador: meio físico que realiza a Computação Programas: processam e computam informações
	Computação como produtora do computador	Computador, algoritmos, programas, Internet
Comunicação	A Computação viabiliza novas formas de comunicação ou aumenta sua eficiência	Celulares, e-mails, Internet, redes de comunicação
	A Computação é uma mídia ou linguagem para a expressão e comunicação	Imagens digitais, textos, sistemas, jogos
	Comunicação na Computação como interação humano-computador	Interfaces gráficas, programas

Tabela 4.34 – Processos de objetivação e ancoragem dos Princípios da Computação (continuação)

Princípio	Ideias sobre as quais se ancora	Objetos e imagens que lhe conferem concretude
Coordenação	Computador como plataforma de auxílio à gestão ou organização do trabalho	Grupos de trabalho que se relacionam com auxílio dos computadores Gestão de projetos, suas etapas e dados
Armazenagem	A Computação amplia ou facilita a capacidade de armazenamento de dados	Memória (física), dispositivos de armazenamento (*pendrives*, discos rígidos e afins), bancos de dados
Avaliação	Monitoramento, supervisão, verificação e validação do desempenho do sistema (*software* e *hardware*) e da sua eficácia ou qualidade de operação	Testes, notas, provas, dados, relatórios
Projeto	Computação como meio de projeto	Programas de Design assistido por computador, computador como ferramenta de resolução de problemas
	Computação como auxílio ao desenvolvimento de projetos	Programas de gestão de projetos ou de processamentos de dados de projetos
Abstração	A abstração reduz a complexidade dos processos computacionais	Códigos, números e dados (abstratos) que geram programas e informações (concretas)
Automação	A Computação permite a automação das tarefas, processos, e tarefas – otimiza, facilita, simplifica	Robôs e máquinas autônomas capazes de substituírem o trabalho humano
Algoritmo	Algoritmo como sequência de instruções para as operações do computador	Receita, passo a passo
	Algoritmo como operações matemáticas que resolvem problemas com ou sem computador	Números e funções matemáticas
	Algoritmo como programação de computadores	Códigos

4.7 – Estudantes de Design e as práticas com a Computação e os computadores

O terceiro grupo de dados, referente à última parte do questionário do Anexo II, foi composto pelas respostas dos estudantes quanto a 1) o auxílio da Computação e 2) o auxílio dos computadores na solução de problemas de Design, sejam eles acadêmicos, profissionais ou do cotidiano. De acordo com os passos descritos no método, as respostas foram analisadas em duas etapas: a) análise de conteúdo, identificação e categorização temática dos relatos; b) cálculo da distância euclidiana entre termos relevantes identificados nas categorias temáticas, buscando coocorrência. Os dados foram coletados junto aos mesmos 86 estudantes cujas evocações e respostas foram discutidas nas análises prototípica e de conteúdo (seções 4.4 e 4.5).

Computação x Solução de problemas de Design

A Computação é pensada como ferramenta de projeto por 59 dos 86 participantes da pesquisa (doze programadores e 47 não programadores), principalmente no que diz respeito à simplificação de tarefas como desenhar, editar imagens, projetos gráficos e desenvolver sites utilizando aplicativos de *desktop publishing*; facilitar a realização de trabalhos que seriam muito complexos caso executados manualmente; busca de informações e levantamento de dados que auxiliam a projetar; coordenação e comunicação na realização de trabalhos; diminuindo o tempo, esforço ou investimentos necessários para desenvolver um projeto.

Tabela 4.35 – Análise de conteúdo Computação x solução de problemas de Design (N=86)*

Tema	***f***	***%***
Computação como ferramenta (que simplifica, otimiza, facilita)	59	64,13
Computação como habilidade cognitiva	12	13,04
Computação como mídia	15	16,30
Não respondeu	6	6,52

*As respostas contêm temas de uma ou mais categorias.

Doze participantes, sendo oito programadores e quatro não, forneceram respostas relacionadas à ideia da Computação como habilidade cognitiva, tais como:

> [...] Enxergar um problema de Design pela ótica de um pensamento computacional é uma (boa) maneira de vislumbrar possibilidades "conceituais" sem perder o foco do viés prático dos projetos de Design, isto é, manter a atenção em como o processo vai se desenrolar mesmo nas camadas mais "baixas". Isso se faz fundamentalmente importante se considerar que boa parte do que se coloca no mundo tem (ou deveria ter) participação direta da Computação. Acho que entender a computação amplia horizontes, resumindo. (*M, 26 – UFES, programador*)

> Ajuda a racionalizar a resolução de alguns problemas. Pensar nas condicionais, no que pode ser aproveitado, organizado, ordenado e, de certa forma, até automatizado. (*M, 24 – UFES, não programador*)

> A Computação auxilia a identificação de padrões ao categorizar estímulos e informações recebidos. Identificado um padrão, fica mais fácil antecipá-lo ou subvertê-lo. (*M, 27 – UnB, não programador*)

> O pensamento lógico adquirido devido ao uso de uma linguagem de Computação me auxiliou a descobrir saídas para os problemas com mais rapidez. (*F, 20 – UFES, programadora*).

Ao contrário desse grupo, outros quinze participantes não programadores, explicaram que a Computação seria uma mídia, assumindo o computador como objeto que permite o acesso a ela. Essa perspectiva aproxima-se da primeira categoria, que pensa a Computação como ferramenta, com a ressalva de que os recursos de comunicação e as possibilidades de pesquisa de informações e referências de projeto aparecem lado a lado com os usos dos programas típicos da área de Design:

> Sou uma péssima pessoa para trabalhos manuais, inclusive desenhos, símbolos. desenhar no computador é muito bom pra mim. Com um computador e internet descubro muitas referências visuais. (*F, 23 – UFES, não programadora*)

> A Computação facilitou o acesso à informações que antes eram mais difíceis de conseguir, como referências de trabalhos, além de permitir o surgimento de novas resoluções, como *softwares* entre outros. (*M, 24 – UFES, não programador*)

> Buscar informações ver referências distrair (às vezes é bom no processo de criação) conectar com outras pessoas trocar informações guardar dados organizar dados. (*M, 27 – UnB, não programador*)

> A Computação me ajuda na busca por referências para a realização de trabalhos e facilita a comunicação no dia-a-dia. (*F, 25 – AESO, não programador*)

> A Computação está presente em quase todas as etapas do meu trabalho, desde pesquisa até a geração de ideias (que antes era feita manualmente) são feitas via PC. A automação de certos processos, como o uso do Photoshop para redimensionar imagens, é outro facilitador. (*F, 26 – UnB, não programadora*)

> A Computação me auxilia através da pluralidade de recursos existentes na internet para a efetiva pesquisa acerca de tais problemas. (*M, 18 - UnB, não programador*)

Tal diferença parece reforçar os processos de objetivação da Computação no computador discutidos nas análises prototípica e de conteúdo, principalmente entre os que não programam e se utilizam mais intensamente do potencial instrumental da Computação. Esses estudantes ancorariam as contribuições percebidas por essa tecnologia nas possibilidades oferecidas pelos pacotes de aplicativos conhecidos mais do que nas

estratégias de solução de problemas mencionadas na categoria Computação como habilidade cognitiva.

No que diz respeito às distâncias euclidianas médias (DEM[53]) para a coocorrência dos termos relevantes identificados na análise de conteúdo, os pontos a seguir[54] apresentam outra possibilidade de análise das explicações fornecidas pelos estudantes:

- *Ajud*[55]* (Fig. 4.3)*:* Computação ajuda a pensar; ajuda em tudo que faço; ajuda porque tudo no computador só é possível por causa da Computação.
- *Auxilia*:* Computação auxilia a descobrir saídas para problemas; Computação auxilia nos processos de Design; Computação auxilia a identificação de padrões; Computação auxilia e possibilita criações no Design; Computação me auxilia através da pluralidade de recursos disponíveis.

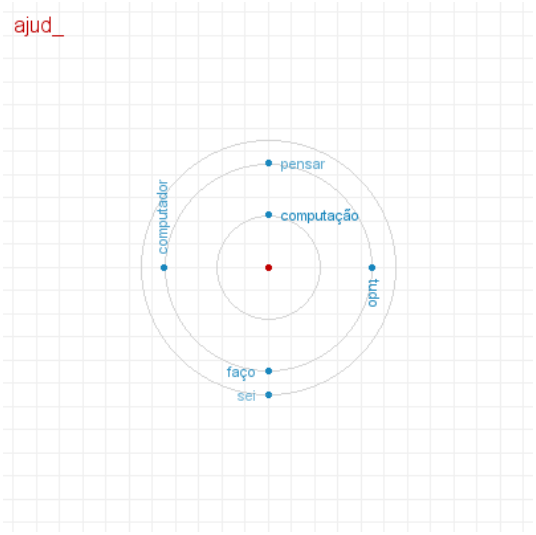


Figura 4.3 – DEM para ajud*

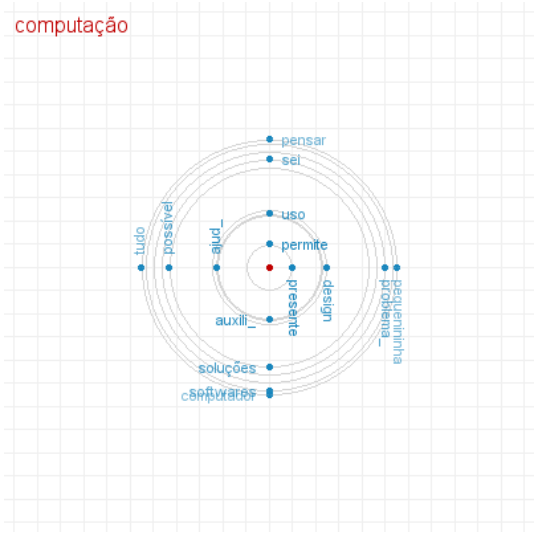


Figura 4.4 – DEM para computação

- *Computação* (Fig. 4.4): está presente no mundo de uma maneira geral; A Computação permite produzir conteúdo; cria os programas que uso; permite o entendimento de um pequeno mundo ou sistema na criação de novos; A permeia o Design completamente, tanto em soluções quanto em problemas; serve na elaboração de soluções; faz os computadores funcionarem.

[53] A distância euclidiana unidimensional é calculada a partir da raiz quadrada do quadrado da diferença entre dois pontos. No contexto da análise realizada, considerou-se que cada palavra possui uma posição *p* em relação ao início do corpus e calculou-se a distância até a posição p_1, p_2, p_n... de todos os outros termos. Palavras que coocorrem frequentemente com distâncias menores ao longo do corpus tendem a estar relacionadas. Os diferentes raios das circunferências representam as distâncias entre as palavras do corpus e o termo analisado ao centro (distância = 0). Ver Segaran (2008) para uma descrição avançada e exemplo de implementação do algoritmo em Python. A Tabela 12.1 do Anexo IV apresenta a lista completa de frequências dos termos.

[54] As Figuras 4.4 a 4.8 foram selecionadas para ilustrar graficamente as distâncias euclidianas medias dos termos mais relevantes encontrados no discurso dos participantes.

[55] O asterisco (*) indica que o processamento do texto lematizou as palavras, incluindo formas no singular e plural, diferentes pessoas e tempos verbais, desde que o aspecto semântico não fosse alterado.

- *Computador* (Fig. 4.5): sem as ferramentas do computador tudo seria mais difícil; tudo que faço no computador só é possível devido à Computação;
- *Criação* (Fig. 4.6): [ajuda] por meio dos *softwares* de planejamento e criação; por meio da criação de softwares, programas de busca de imagens, textos, referências; a Computação tornou possível a criação de ferramentas (*softwares*).
- *Dados*: a Computação organiza e guarda dados; transforma banco de dados em dados a serem impressos; nos conecta com outras pessoas para trocar informações, guardar e organizar dados.
- *Desenvolvimento:* de programas;
- *Forma*: [a Computação] permite produzir [conteúdo, imagens] e tornar isso visível de forma mais eficiente.
- *Problema** (Fig. 4.7): [a Computação] fornece ferramentas para solucionar problemas; podemos solucionar problemas com maior precisão e agilidade.
- *Programação*: [a Computação] ajuda na programação; aprender programação permite pensar soluções mais adequadas para problemas.
- *Programas* (Fig. 4.8): utilização ou uso nas tarefas; na criação; em *briefings*; projetos; acessar internet; desenvolver ideias; com [programas] Adobe, Microsoft Office.
- *Projeto**: uso de programas [Microsoft Office] para armazenar informações relativas ao projeto (projeto escrito, metodologia, briefing, ideias); uso de programas que facilitem a criação.

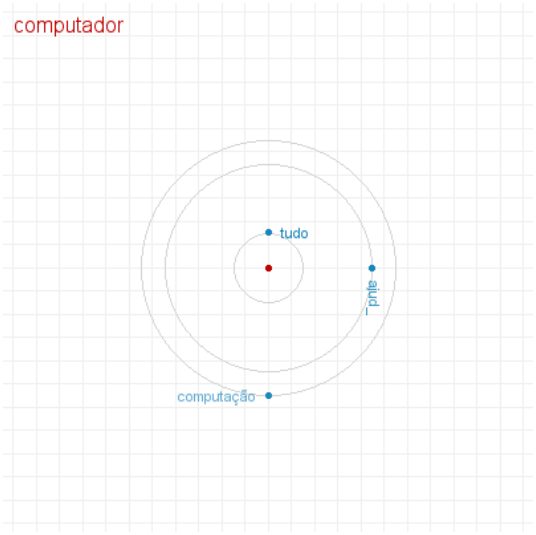


Figura 4.5 – DEM para computador

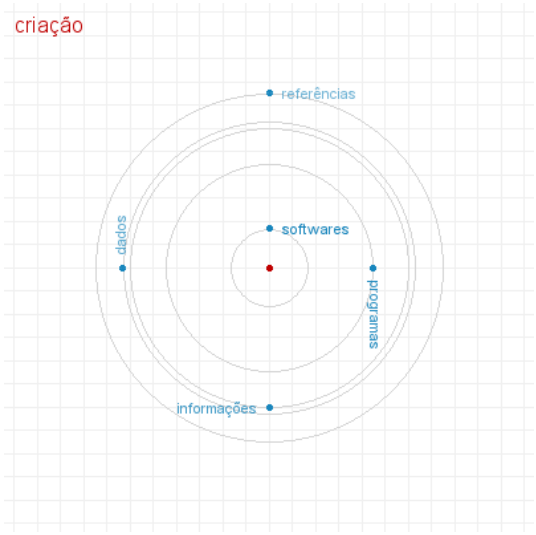


Figura 4.6 – DEM para criação

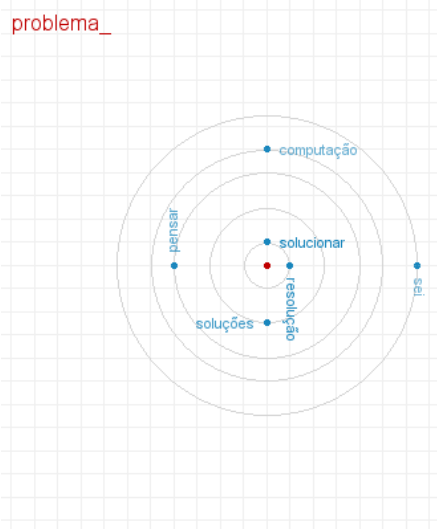


Figura 4.7 – DEM para problema*

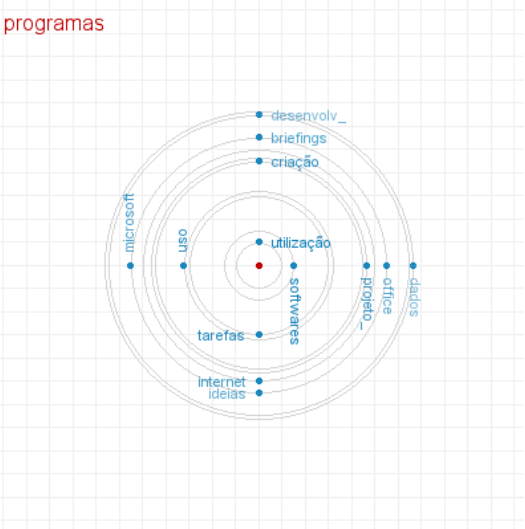


Figura 4.8 – DEM para programas

Computadores x Solução de problemas de Design

No que tange à segunda questão trabalhada, o auxílio dos computadores nos processos de resolução de problemas de Design pelos estudantes, as categorias resultantes da análise de conteúdo (Tabela 4.36) complementam os argumentos que situam o computador no núcleo central da estrutura das RS da Computação.

Tabela 4.36 – Análise de conteúdo Computadores x solução de problemas de Design (N=86)

Tema	***f***	***%***
Computador como ferramenta	75	70,09
Computador como mídia	28	26,17
Não respondeu	3	2,80
Outros	1	0,93

*As respostas contêm temas de uma ou mais categorias.

Nessa última pergunta, os temas referentes às habilidades cognitivas fundadas no conhecimento sobre a Computação que foram citados nas respostas à questão anterior ficaram ausentes, resumindo os relatos aos usos ferramentais dos programas oferecidos pelas máquinas, tanto entre programadores quanto não programadores, conforme se observa a seguir:

> Praticidade de fazer mil ações manuais em um lugar só. (*M, 23 – UFES, programador*)

> É a ferramenta de trabalho primária. Permite moldar as ideias em algo. Experimentar e re-moldar se for o caso [...] (*M, 26 – UFES, programador*)

> Eu não faço muita distinção entre computação e computador nesse caso, possivelmente por ver o computador como um meio, apenas. É o caso do abridor de garrafas e a garrafa. Eu só me importo com o que eu vou beber, basicamente. O computador é o suporte que me permite usar os programas que eu preciso. *(F, 29 – UFES, não programadora)*

> Os computadores e softwares de edição revolucionaram o Design Gráfico e possibilitaram o surgimento do Web Design. [...] Hoje é possível tirar uma foto digital, alterar seu tamanho, adicionar texto e imprimir vários originais em questão de segundos. Antes esse mesmo processo envolvia a ampliação/redução, revelação, fotocomposição e impressão, o que poderia levar dias. *(M, 23 – UnB, programador)*

> Servem como suporte para softwares gráficos que possibilitam a pré-visualização de soluções possíveis para determinados projetos, simplificando o processo de criação e economizando tempo, material e dinheiro. *(M, 25 – UnB, não programador)*

> As máquinas ajudam, pois tornam os trabalhos mais rápidos e tornam o trabalho mais prático ou até mesmo fazendo melhorias em algo feito a mão. (*F, 21 – UFPE, não programadora*).

A concepção do computador como mídia ou metameio também aparece das respostas, porém com frequência menor e conteúdos semelhantes àqueles mencionados pelos participantes acerca dos usos da Computação na resolução de problemas de Design. Seguem alguns exemplos:

> [...] pensando no acesso à informação, é indiretamente responsável por todas as referências que uso, uma vez que tenho acesso à infinidades de materiais de referência (textos, ilustrações, livros, filmes, vídeos, games, aplicativos). (*M, 26 – UFES, programador*)

> O computador me ajuda a pesquisar, me atualizar no mundo em volta e as tendências do momento. com o computador posso ter acesso a muito mais coisa do que conseguiria em revistas ou jornais. *(F, 18 – UFES, programadora)*

> O computador me ajuda apenas enquanto plataforma para pesquisa. Mas se eu utilizar um celular que tenha internet ele vai suprir igualmente minhas necessidades de pesquisa. A única vantagem que vejo no computador é o tamanho da tela. *(F, 25 – AESO, não programadora)*

> Os computadores, pra mim, são uma ferramenta que eu uso pra fazer diversas coisas que já falei ali em cima – me comunicar on-line, trabalhar, essas coisas. No caso, eu só uso coisas que já chegaram prontas pra mim, não crio nada do zero, minhas próprias ferramentas. *(F, 22 – UFES, não programadora)*

O cálculo das distâncias euclidianas médias entre as palavras que compõem o corpus dessa última questão (Tabela 12.2 do Anexo IV e Figuras 4.9 e 4.10) indicam coocorrências de termos que igualmente reforçam os usos dos computadores como ferramenta, mídia ou metameio. As palavras *computador* (*f*=31), *ferramenta* (*f*=14) e *uso* (*f*=13) aparecem com maior frequência no discurso dos estudantes, participando da articulação de ideias sobre as facilidades oferecidas pelo dispositivo enquanto ferramenta de projeto, de comunicação, de acesso a informações e para a organização de dados e informações de trabalho. Todas essas questões estão ligadas ao uso de programas e *softwares*.

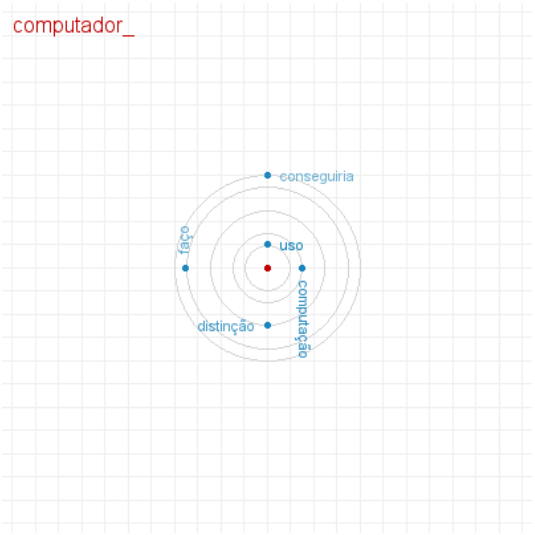


Figura 4.9 – DEM para computador*

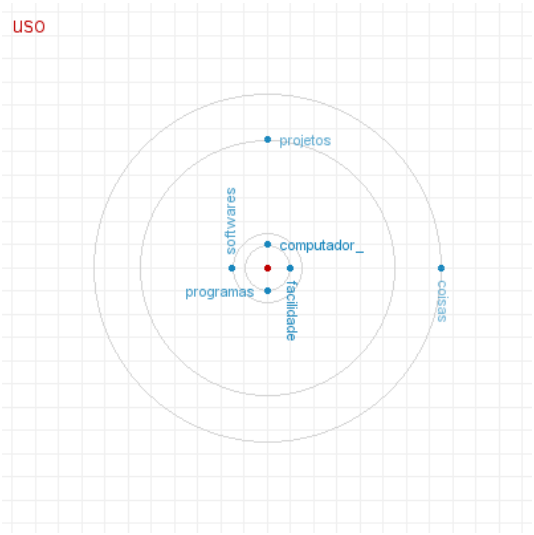


Figura 4.10 – DEM para uso

4.8 – Terceira síntese: RS da Computação e práticas com os computadores

Os resultados das análises anteriores (atividades, prototípica, de conteúdo e teste de centralidade) reconsiderados à luz das questões identificadas nas práticas dos estudantes de Design na solução de problemas utilizando a Computação e os computadores apontam caminhos para a terceira e última síntese desta parte da pesquisa. Em primeiro lugar, parece possível afirmar que o *potencial instrumental* é o mais utilizado pelos estudantes, uma vez que o computador figura como elemento central na estrutura da RS sobre a Computação, tanto entre programadores quanto não programadores. Tal elemento exerce sua função geradora (Sá, 1996) na construção de sentido dos demais elementos da

representação, uma vez que subordina Computação à lógica de operação do computador, das suas ferramentas e aplicações.

Em termos prescritivos, a centralidade incondicional do *computador* na representação contribui para explicar as práticas e concepções de natureza ferramental descritas pelos estudantes: uso de programas em diversas situações (metameio), mídia (acesso à informação), ferramenta de trabalho, de comunicação e assim por diante. Mesmo entre os estudantes programadores, a *programação* não aparece como prática relevante, apesar desse elemento estar presente na primeira periferia da representação e código no núcleo central. Isso pode significar que, apesar da possibilidade de programar estar colocada, aqueles estudantes experimentam um processo de transformação progressivo (Abric, 1993, 2001) da estrutura da representação, onde o exercício da programação ainda é visto como prática que não contradiz a imagem do *computador* como objetivação da Computação. Nesse sentido, pode-se argumentar que *programar* também teria um sentido ferramental para os estudantes, ao invés de figurar como uma prática que poderia contribuir para a emergência de uma representação mais relacionada ao *potencial representacional o* da Computação.

Nos casos nos quais o *potencial representacional* aparece, principalmente entre os estudantes programadores, a fonte das mudanças tem origem externa ao exercício específico dos saberes ligados ao Design: 1) inserção diferenciada no mercado de trabalho, com independência ou certo grau de autonomia, em posições que dialogam diretamente com a programação (por exemplo, Web Design) e 2) desenvolvimento de projetos pessoais que dependem de programação de alguma forma (*blogs*, jogos). Um possível aumento nos usos dos computadores por estudantes de Design e as pressões do mercado de trabalho poderiam, portanto, se configurar como um contexto favorável a mudanças graduais na estrutura da RS sobre a Computação, considerando que aqueles participantes da pesquisa que manifestaram percepções menos ferramentais sobre a área o fizeram a relatando mudanças primeiramente nas suas práticas, seguidas por mudanças no entendimento dos prescritores condicionais periféricos ligados à ideia de programação, sem ainda afetar as prescrições absolutas do núcleo. Os resultados do teste de centralidade das evocações de *programação* (Tabela 4.29) sustentam esse argumento ao indicar que mesmo que esta atividade seja muito importante para o entendimento que os estudantes elaboram sobre a Computação, as duas ideias ainda não são indissociáveis para a maioria deles.

Em síntese, para os participantes é possível pensar sobre a Computação e usar os computadores sem programação porque a programação não é uma prática cotidiana na

solução de problemas de Design, por mais que o dispositivo e os princípios daquela área de conhecimentos estejam presentes no processo de alguma forma. O computador é o elemento que organiza a RS da Computação dos estudantes, dos seus princípios e regula a transformação das suas práticas.

5.1 – Das representações sociais da Computação à Zona de Desenvolvimento Proximal

Os elementos das representações sociais (RS) sobre os Princípios da Computação compartilhados pelos estudantes e discutidos no capítulo anterior oferecem alternativas para o planejamento de um contexto propício para a aprendizagem dos Princípios da Computação sugeridos por Brennan, Chung e Hawson (2011), Wing (2008a) e Denning e Martell (2007b). Retomando os trabalhos citados na revisão de literatura desta tese, duas principais estratégias pedagógicas vêm sendo adotadas em iniciativas de desenvolvimento do Pensamento Computacional (CT): 1) criação de currículos que abordem os Princípios da Computação independentemente de ferramentas disponíveis e que dialoguem com outras disciplinas ou temáticas com as quais os aprendizes estão familiarizados; e 2) desenvolvimento de linguagens de propósito ou públicos específicos que explorem os Princípios da Computação implícita ou explicitamente, aproveitando-se daquelas especificidades.

A estratégia pedagógica desenvolvida pelo autor[56] integra as alternativas do currículo e da linguagem num sistema único e mais próximo das práticas e representações levantadas pela pesquisa junto aos estudantes de Design. Segundo Kay (1989), as experiências de Seymour Papert e seus colaboradores com a linguagem LOGO mostraram que uma ferramenta projetada levando-se em consideração as questões dos usuários poderia ser mais bem-sucedida que uma solução aleatória. No caso dos estudantes que participaram da pesquisa frente aos Princípios da Computação a serem abordados, essas questões podem ser sintetizadas em três grupos, a saber:

I. Diversificação das imagens que objetivam a Computação para além dos computadores, visando fomentar a discussão do *potencial representacional* da área com os estudantes;

[56] No planejamento original deste trabalho, a linguagem Processing e o currículo Computer Science Unplugged seriam utilizados como base para o desenvolvimento de um contexto de aprendizagem no curso de Design da UFES. Com as dificuldades enfrentadas pela interrupção das aulas na turma que participaria do experimento no período da greve das universidades federais brasileiras em 2012, optou-se pelo desenho metodológico apresentado.

II. Criação de oportunidades para que os Princípios da Computação possam ser abordados de forma concreta, com temáticas ou situações familiares para os estudantes, para em seguida abordá-los de forma mais abstrata e geral;

III. Combinar a diversificação dos objetos que materializam a Computação e a abordagem dos seus princípios em processos de resolução de problemas que contribuam para a percepção de utilidade daqueles conhecimentos pelos estudantes nas suas práticas de projeto.

A Tabela 5.1 apresenta uma proposta de articulação entre os resultados da pesquisa[57] e as oportunidades identificadas para a construção da abordagem pedagógica que contemple as sete janelas do conhecimento sobre a Computação (Denning e Martell, 2007b), os processos essenciais de abstração e automação (Wing 2008a), os conceitos, práticas e perspectivas do Pensamento Computacional (Brennan, Chung e Hawson, 2011) e o conceito de algoritmo (Knuth, 1997). A estrutura da tabela foi organizada da seguinte forma: Na primeira coluna, são listados os Princípios da Computação; a segunda coluna apresenta as RS elaboradas pelos estudantes de Design acerca daqueles Princípios; a terceira coluna lista as práticas enunciadas pelos estudantes relacionadas às RS; a quarta coluna apresenta abordagens pedagógicas que visam incentivar transformações nas RS relacionadas a cada Princípio por meio de novas práticas; a quinta coluna descreve a estratégia adotada nesta tese para oferecer as novas práticas aos estudantes.

O conceito geral da estratégia pedagógica baseia-se num jogo de tabuleiro cujo objetivo é controlar um foguete no espaço, desviando de asteroides e coletando estrelas. Esse controle é feito pela construção de algoritmos escritos em uma linguagem de programação específica que abstrai os comandos do foguete, aplicando os Princípios da Computação de forma potencialmente familiar para os estudantes de Design. A linguagem foi batizada *RocketSocket*, fazendo referência ao uso dos encaixes ("*sockets*") como forma de controle do foguete ("*rocket*").

Tabela 5.1 – Dos Princípios da Computação à Estratégia Pedagógica

Princípio	RS dos Estudantes	Práticas	Abordagem	Estratégia
Computação	Computador como produto da Computação Computação como produtora do computador	Computador como ferramenta e metameio Desenvolvimento de programas e algoritmos	Introduzir outras aplicações, conceitos e imagens que objetivem e ancorem a Computação para além do computador	Adotar um foguete espacial programável para coletar estrelas e desviar de asteroides como contexto de aprendizagem

[57] A Tabela 4.34 das páginas 114-115 é a base dos dados apresentados.

Tabela 5.1 – Dos Princípios da Computação à Estratégia Pedagógica (continuação)

Princípio	RS dos Estudantes	Práticas	Abordagem	Estratégia
Comunicação	A Computação é uma mídia ou linguagem para a expressão e comunicação	Troca de mensagens, comunicação online	Conceituar a programação como um processo de comunicação entre o aprendiz e o dispositivo computacional	Construir uma linguagem de programação específica para o controle do foguete baseada em comandos intuitivos e de semântica familiar ao aprendiz
	Comunicação na Computação como interação humano-computador	Uso de interfaces gráficas e programas		
Coordenação	Computador como plataforma de auxílio à gestão ou organização do trabalho	Trabalho em equipe, divisão de tarefas, gestão de projetos	Discutir a colaboração e divisão de tarefas entre dispositivos computacionais	Explorar missões nas quais dois foguetes precisam ser programados para trabalharem em conjunto
Armazenagem	A Computação amplia ou facilita a capacidade de armazenamento de dados	Armazenamento de dados para uso futuro	Explorar registro e recuperação de dados em memórias variadas	Dotar o foguete de uma memória limitada para a manipulação de dados relativos aos planos de voo
Avaliação	Monitoramento, supervisão, verificação e validação do desempenho do sistema (*software* e *hardware*) e da sua eficácia ou qualidade de operação	Testes, provas	Discutir a eficiência das variadas soluções para problemas computacionais nos termos de algoritmos	Oferecer diferentes níveis de dificuldade para os trajetos de coleta das estrelas e desvio dos asteroides, limitando uso de comandos e incentivando o aprendiz a explorar soluções alternativas
Projeto	Computação como meio de projeto	Uso de programas, realização de pesquisas, organização de dados de projeto	Oferecer ferramentas de apoio à análise do problema e síntese da solução, com foco na revisão crítica e discussão dos algoritmos	Incluir a construção de planos gráficos de voo anteriores à construção do código, permitindo ao aprendiz refletir sobre o problema antes de partir para a solução programada
	Computação como auxílio ao desenvolvimento de projetos			
Abstração	A abstração reduz a complexidade dos processos computacionais	Transformação de códigos, números e dados em programas e informações	Abordar a construção de algoritmos e processos em diferentes níveis de abstração, bem como o relacionamento entre cada nível	Dotar a linguagem de programação de recursos de abstração a partir dos elementos primitivos
Automação	A Computação permite a automação das tarefas, processos, e tarefas – otimiza, facilita, simplifica	Substituição do trabalho humano por processos automáticos	Discutir a automatização de processos em termos de otimização, simplificação e facilitação de operações	Dotar a linguagem de programação de estruturas de controle que permitam a otimização das instruções de voo do foguete
Algoritmo	Algoritmo como sequência de instruções para as operações do computador	Uso de sequências de instruções no computador tendo um objetivo	Introduzir os conceitos do CT: sequências, laços, condicionais, paralelismo, operadores e dados	Definir a temática dos problemas de coleta das estrelas e desvio dos asteroides de forma que empreguem cada um dos conceitos do CT
	Algoritmo como operações matemáticas que resolvem problemas com ou sem computador	Resolução de equações e uso de funções matemáticas	Explorar dimensões matemáticas dos algoritmos	Incluir elementos primitivos com questões matemáticas na linguagem, bem como nos problemas a serem resolvidos com os planos de voo
	Algoritmo como programação de computadores	Desenvolvimento de programas	Discutir o algoritmo como solução independente da linguagem ou meio de realização	Permitir que o aprendiz, ao longo do processo, construa sua própria linguagem e implemente seus planos de voo nela, permitindo a reflexão sobre os princípios envolvidos no programa apesar da sintaxe da linguagem do foguete

Estrutura da linguagem RocketSocket

Abelson e Sussman (1996) explicam que ideias são expressas por meio de linguagens de programação utilizando elementos primitivos, seus meios de combinação e de abstração. Por elementos primitivos (Fig. 5.1a) entende-se os átomos da linguagem, as unidades mínimas de instrução. No caso do foguete, as instruções mínimas referem-se aos controles de voo – ligar e desligar o motor, avançar ou girar – e também a operadores aritméticos e estruturas de controle[58] do programa. Já os meios de combinação da linguagem (Fig. 5.1b) são as regras orientam os encadeamentos válidos de dois ou mais átomos em expressões mais complexas. Esses encadeamentos são a implementação ou programação do plano de voo do foguete nos termos de algoritmos que solucionam o problema do trajeto: ligar o motor na partida, girar para coletar as estrelas ou desviar dos asteroides, desligar o motor no término. Os meios de abstração (Fig. 5.1c) são as possibilidades de combinação de encadeamentos de forma que estes passem a representar novos átomos que podem ser úteis em diferentes situações. Um determinado algoritmo para o foguete realizar um ziguezague pode ser representado como uma caixa preta cuja aparência é de instrução única, mas que no seu interior há uma sequência complexa de instruções.

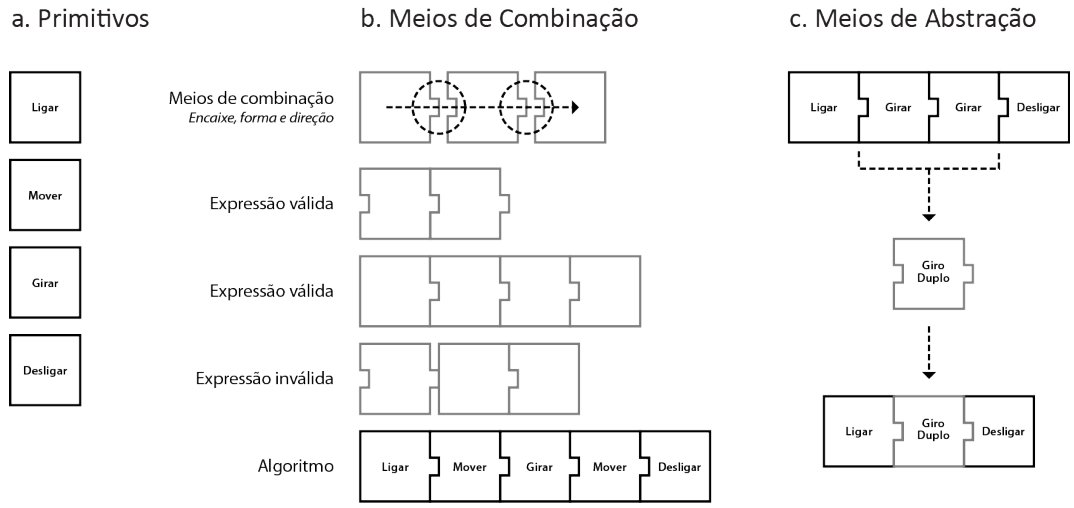


Figura 5.1 – Elementos primitivos, meios de combinação e abstração

A linguagem de programação do foguete, bem como seus elementos primitivos, meios de combinação e abstração foram criados tendo em vista a orientação construcionista do processo de aprendizagem dos princípios do CT:

[58] Segundo Dale e Lewis (2011, p.22), Estruturas de controle são formas de alterar o fluxo de controle do algoritmo, determinando a ordem na qual as instruções em um programa serão executadas. Linguagens de programação estruturada modernas oferecem controle do fluxo via seleção (condicionais), repetição (laços e iterações) e subprogramas (outros programas executados pelo programa atual).

- Seguindo a concepção de Bruner (1976) e de Kay (1987, 1989) acerca das formas de representação, optou-se por definir os átomos da linguagem como blocos de encaixe que representam os movimentos permitidos para o foguete. O primeiro contato do aprendiz com o controle do foguete seria pela *ação* ou manipulação direta dos comandos (Fig. 5.2a).

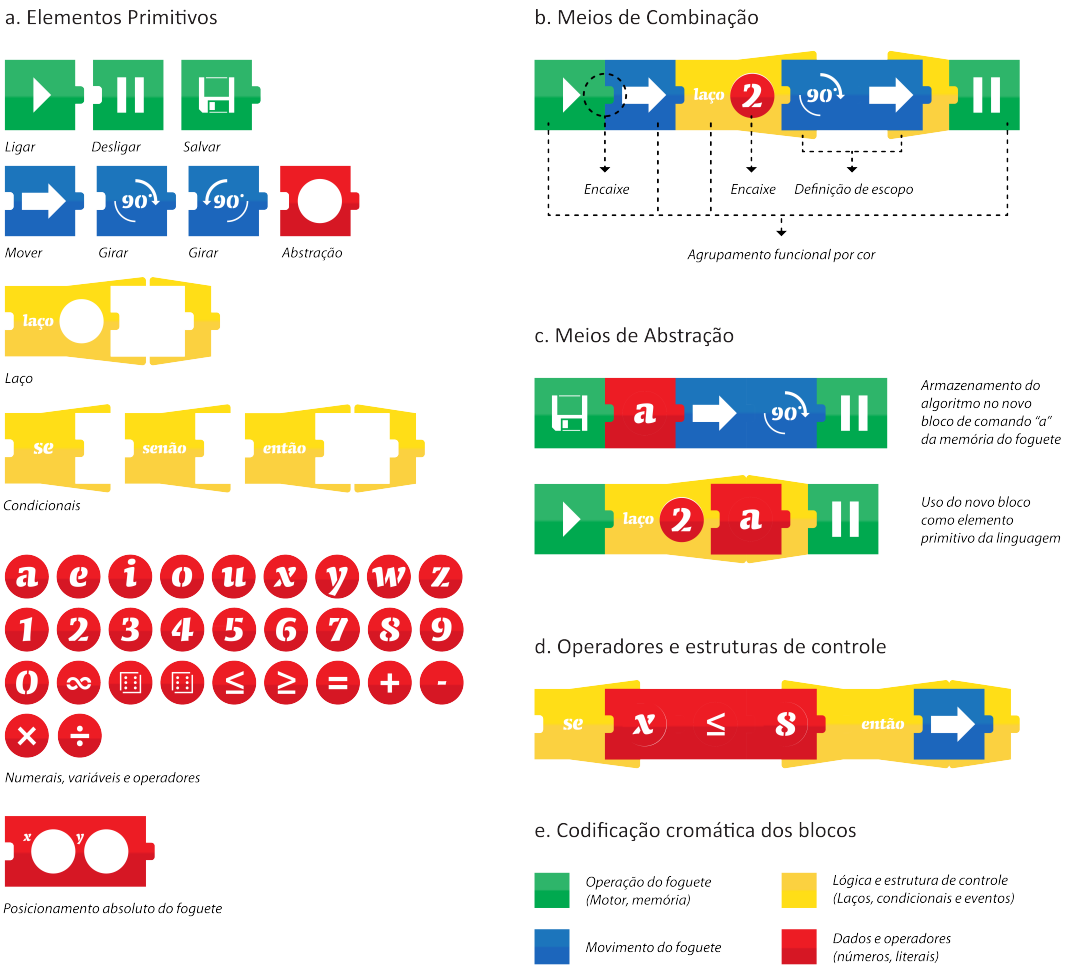


Figura 5.2 – Detalhamento da linguagem

- Os meios de combinação da linguagem estão explícitos na forma dos blocos (Fig. 5.2b), dos encaixes possíveis e das suas cores, de maneira que o aprendiz pode compreender a lógica de encadeamento inicialmente pela *imagem* gerada pela construção do código e gradualmente refletir sobre o aspecto *simbólico* das expressões que constrói.
- Os meios de abstração (Fig. 5.2c) operam da mesma forma que os elementos primitivos, permitindo o uso dos encaixes como abertura para a construção de

abstrações de acordo com os avanços do aprendiz. Um determinado algoritmo pode ser encapsulado em um novo elemento primitivo a qualquer momento, permitindo desenvolvimentos sucessivos da linguagem a partir da experiência.

- Os demais elementos primitivos da linguagem – laços, condicionais, operadores aritméticos, numerais e variáveis – completam as possibilidades por meio da mesma estratégia de combinação pela forma, encaixe e cores (Fig. 5.2d).

- O tabuleiro (Fig. 5.3), que representa o espaço sideral, é uma grade com número n igual de linhas e colunas, totalizando n^2 posições navegáveis pelo foguete. Os elementos primitivos fundamentais da linguagem são o comando de avançar uma casa na grade, seguindo a orientação vertical ou horizontal atual, e os comandos de girar o foguete 90 graus em sentido horário ou anti-horário. Todos os planos de voo consistem em sequências de deslocamentos e giros pela grade, buscando sempre coletar as estrelas e desviar dos asteroides.

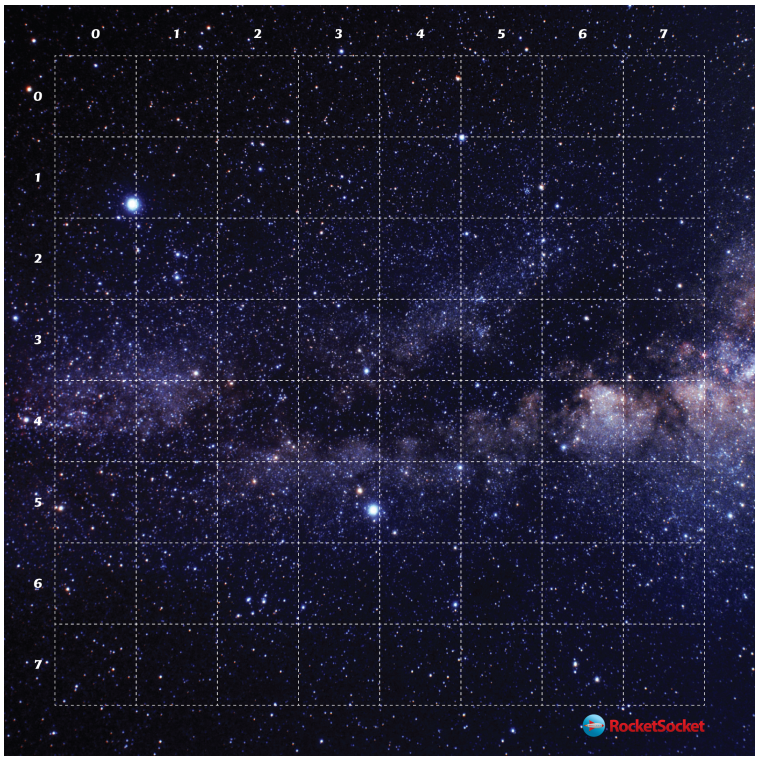


Figura 5.3 – Tabuleiro

Programação, causalidade e finalidade

Ao contrário das implementações tradicionais da linguagem LOGO, que situam o aprendiz no centro do sistema de coordenadas de um micromundo (Kay, 1989, p.126), a linguagem proposta requer destarte a decentração do sujeito que precisa assumir a perspectiva do

piloto do foguete (sua posição, direção atual, possibilidades de movimento) para compreender os resultados que as transformações dos comandos desencadearão. Para ser programado, o código de controle do foguete pode ser *construído* criativamente em dois sentidos, buscando por um lado contemplar os diferentes perfis de aprendizes (Turkle e Papert, 1991) e por outro permitir percursos microgenéticos tanto causais quanto finais (Blanchet, 1997): 1) *ascendente* ou causal-final, apropriado para os iniciantes, que parte dos encaixes físicos (forma e cor compatíveis – *representação ativa*), segue pela composição dos elementos da linguagem (os *comandos* identificados pelas imagens – *representação icônica*) em sentenças válidas para a resolução do problema (o algoritmo em si – *representação simbólica*); ou 2) *descendente* ou final-causal, para os aprendizes mais experientes, que parte da concepção do algoritmo (*representação simbólica*) em direção à busca dos comandos para compor sentenças que o realizem (*representação icônica*) para por fim realizar os encaixes das peças (*representação ativa*). Em cada problema, o aprendiz recebe um número específico de elementos primitivos que podem ou não ser utilizados na solução, de forma que a exploração das possibilidades e a existência de mais de uma alternativa seja parte do processo.

Práticas e perspectivas do Pensamento Computacional

No processo de resolução de problemas utilizando a linguagem, o percurso da causalidade à finalidade é expresso primeiramente em planos de voo de papel (Anexo V), onde o aprendiz representa graficamente o encadeamento dos controles do foguete na situação e o trajeto que ele acredita resultar daquelas instruções antes de efetivamente construir o algoritmo com os blocos de encaixe. Caso haja algum problema com o código construído, o aprendiz é confrontado com a representação equivocada do problema para que busque uma nova solução, exercitando a prática de depuração. Os mesmos planos de voo podem ser consultados pelo aprendiz para que aproveite alguma experiência anterior em novas situações-problema ou construa abstrações a partir delas.

Tal desenho para a construção dos algoritmos está diretamente alinhado às práticas do CT defendidas por Brennan, Chung e Hawson (2011): *ser interativo e incremental* – do plano de voo ao código; *testar e depurar* – plano de voo *versus* resultado das peças no tabuleiro; *reusar e remixar* – consulta aos planos de voo anteriores; *abstrair e modularizar* – abstração de algoritmos criados anteriormente em novos elementos primitivos. Ao mesmo tempo, a linguagem também foi pensada de forma a discutir as perspectivas do CT, oferecendo oportunidades de expressão criativa pelo aprendiz, em paralelo à interação e

colaboração com o tutor no desenvolvimento dos programas e à possibilidade de questionar a própria lógica das linguagens de programação.

Planos de voo e resolução de problemas

No centro do processo de aprendizagem da linguagem *RocketSocket* está a construção de planos de voo para o foguete coletar estrelas desviando de asteroides. Para compor a estrutura de jogo de tabuleiro, os diferentes problemas de trajeto foram representados em cartas de baralho (Fig. 5.4) que contêm no verso:

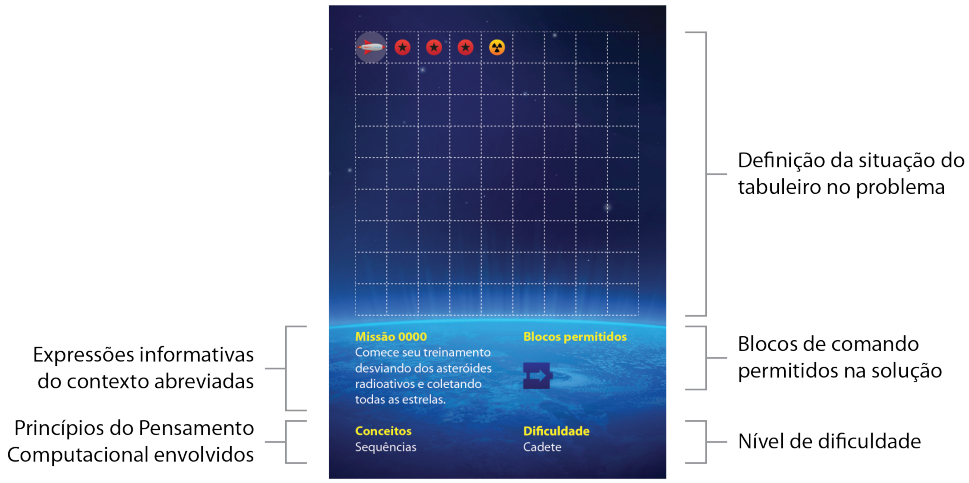


Figura 5.4 – Modelo de carta (verso)

- *A situação do tabuleiro*: posição inicial do foguete, dos asteroides e estrelas;
- *Blocos de comando permitidos*: lista dos blocos que serão disponibilizados pelo tutor para a construção do algoritmo do plano de voo;
- *Missão da carta:* orientação para a construção do plano de voo utilizando expressões informativas do contexto abreviadas (Wertsch, 1985), convidando o aprendiz a elaborar sua própria representação do problema. As cartas são numeradas em binário (0000, 0001, 0010 e assim por diante), oferecendo mais uma oportunidade de contato do aprendiz com os temas da Computação;
- *Conceitos:* indicação explícita dos Princípios da Computação envolvidos;
- *Nível de dificuldade:* uma indicação da complexidade da missão nos níveis Cadete (iniciante, *problema bem formado*), Piloto (intermediário, *problema mal formado*) e Capitão Rocket (avançado, *problema capcioso*).

O tutor-experimentador utiliza um roteiro (Anexo VI) que contém instruções para cada missão baseadas numa perspectiva referencial de expressões de referência comum não-abreviada. A cada carta selecionada, o tutor-experimentador segue as instruções do roteiro para orientar o aprendiz quanto às questões da missão e ilustrar, caso necessário, os princípios do CT envolvidos utilizando situações-problema isomórficas.

O estudo de caso da abordagem pedagógica foi baseada num desenho metodológico que combinou a investigação microgenética das unidades significativas procedimentais e representativas na resolução de problemas (Blanchet, 1997) baseados, por um lado, nos conceitos, práticas e perspectivas do Pensamento Computacional (CT) sugeridos por Brennan, Chung e Hawson (2011), e por outro na orientação construcionista (Harel e Papert, 1991; Bruner, 1976; Kay, 1987, 1989) de considerar diferentes perfis de aprendizes e propor contextos de aprendizagem que valorizem tanto modalidades de representação do conhecimento de funcionamento ascendente (*ativa > icônica > simbólica* ou *causal > final*) quanto descendente (*simbólica > icônica > ativa* ou *final > causal*).

A opção pelo delineamento metodológico no formato de estudo de caso resultou do interesse específico por questões particulares do contexto de aprendizagem dos Princípios da Computação na interação com o *RocketSocket*. Segundo Yin (2001), a investigação de estudos de caso lida com situações tecnicamente únicas, onde haveria mais variáveis de interesse do que pontos de dados e, por isso, esse método seria baseado em várias fontes de evidencias a serem trianguladas, tomando partido de proposições teóricas para conduzir a coleta e a análise de dados. A investigação do uso do *RocketSocket* por estudantes de Design na mediação do processo de aprendizagem dos Princípios da Computação foi construída a partir da definição técnica do estudo de caso citada, observando o maior número possível de variáveis em um número reduzido de participantes, assumindo a triangulação teórica apresentada na introdução desta tese e os resultados do estudo do capítulo 4 como orientação para a coleta e análise de dados.

Participantes

Considerando os perfis identificados na primeira parte da pesquisa (cap. 4), foram selecionados dois alunos do Curso de Desenho Industrial da UFES, sendo um programador e outro não. A seleção desses participantes foi realizada por conveniência, procurando-se por alunos do curso de ambos os sexos com disponibilidade de participar da pesquisa, mantendo o cuidado de não reforçar estereótipos de gênero quanto aos usos dos

computadores[59]. Após aceitarem o convite, os estudantes responderam o mesmo questionário da primeira fase da pesquisa com o intuito de certificar o alinhamento deles com o perfil dos grupos identificados anteriormente. A escolha do número reduzido de participantes para o estudo de caso não buscou contemplar todos os desdobramentos das representações sociais dos Princípios da Computação sobre os processos de resolução de problemas de natureza computacional, mas servir como estudos de casos prototípicos (programadores e não-programadores) acerca da apropriação daquelas representações no desenrolar dos processos cognitivos subjacentes à resolução dos problemas.

> Falar de saber, em termos de conhecimentos familiares, significa passar de um saber geral canônico para conhecimentos particulares, tal como o sujeito, que funciona numa tarefa específica, deles se apropria. Assim, a análise da tarefa (do problema colocado à criança), não pode ser feita sem a análise das ações e representações do sujeito. Reciprocamente, estas devem compreender as características do problema colocado supondo que os conhecimentos formados pelo sujeito dependem também do que é oferecido pela situação (Saada-Robert, 1997, p.161).

Instrumentos

Um protótipo funcional da linguagem *RocketSocket*, descrita no capítulo 5, foi construído em MDF[60] cortado à laser com tabuleiro e peças para a exploração dos conceitos, práticas e perspectivas do Pensamento Computacional. O tabuleiro escolhido foi a versão de três bits (oito casas horizontais por oito verticais) para manter a simplicidade dos movimentos e facilitar a identificação da solução pelos participantes. Os elementos primitivos foram construídos no mesmo processo do tabuleiro e peças.

Tabela 6.1 – Elementos primitivos *RocketSocket*

Elemento	Quantidade
Movimento do foguete	8
Girar 90º sentido horário e anti-horário	2 cada
Liga, desliga	1 cada
Laço	2
Armazenar	1
Estrelas	16
Meteoros radioativos	8
Foguetes	2

[59] As recomendações de Kay (2007) para que não se reforce diferenças de gênero no uso dos computadores em contextos educacionais estão alinhadas à abordagem construcionista: construção de uma cultura dos computadores positiva, com atividades neutras em termos de gênero, com usos significativos para o usuário e que contemple diferentes perfis de aprendizes.

[60] Placa de fibra de madeira de média densidade, fabricada a partir da aglutinação de fibras, resinas e outros materiais. Foi selecionado pelo baixo custo, facilidade de corte, pintura e acabamento.

Além dos elementos primitivos, foram desenvolvidos dois modelos de planos de voo (Anexo V) para que os participantes pudessem representar graficamente as soluções que propuseram para cada problema. O modelo I foi concebido para utilização nos problemas baseados nos blocos:

- *Área 1:* espaço para que o participante escreva o número do problema e a tentativa de resolução em curso;
- *Área 2*: espaço para que o participante desenhe os blocos de comando na mesma sequência que pretende utilizá-los;
- *Área 3*: nove miniaturas do tabuleiro para que o participante represente a sequência de estados do foguete e dos asteroides durante a execução dos comandos indicados na área 2.
- *Área 4*: utilizada no caso de erros, trata-se de um campo específico para que o participante descreva a dificuldade encontrada e indique em qual passo ocorreu.

O modelo II foi concebido para que o participante construa sua própria linguagem de comandos para o foguete, contendo os mesmos conceitos e oferecendo as mesmas práticas e perspectivas que os blocos de encaixe. Essa segunda versão foi desenvolvida para investigar se o participante do estudo representou simbolicamente aquele comando e compreendeu a finalidade dele na solução do problema apresentado:

- *Área 1:* mesma função do modelo I;
- *Área 2*: espaço para que o participante liste quais comandos precisaria para movimentar o foguete e como os nomearia;
- *Área 3*: espaço para que o participante utilize os comandos criados por ele na solução do trajeto;
- *Área 4*: nove miniaturas do tabuleiro para que o participante represente a sequência de estados do foguete e dos asteroides durante a execução dos comandos nomeados na área 2;
- *Área 5*: utilizada no caso de erros, trata-se de um campo específico para que o participante descreva a dificuldade encontrada e indique em qual passo ocorreu.

Procedimentos experimentais

Foi agendado um encontro individual onde os participantes puderam utilizar a linguagem *RocketSocket* para resolver cinco problemas selecionados arbitrariamente a partir do baralho de situações-problema, seguindo o roteiro do estudo de caso (Anexo VI). No início de cada sessão os participantes foram indagados sobre como definiriam e exemplificariam: a) um comando em um computador; b) um programa de computador; c) uma linguagem de programação de computadores.

Buscando experimentar o potencial da linguagem e desenvolvê-la gradualmente, optou-se por abordar nos testes apenas os conceitos de *sequências, laços*, *abstração* e *automação* dos Princípios da Computação, controlando a sequência de resolução (Fig. 6.1). A opção por não permitir o sorteio teve objetivo de aumentar gradualmente a dificuldade das situações-problema e a necessidade de utilização dos blocos conforme sua função: 1) sequências, 2) laços, 3) automação e 4) abstração.

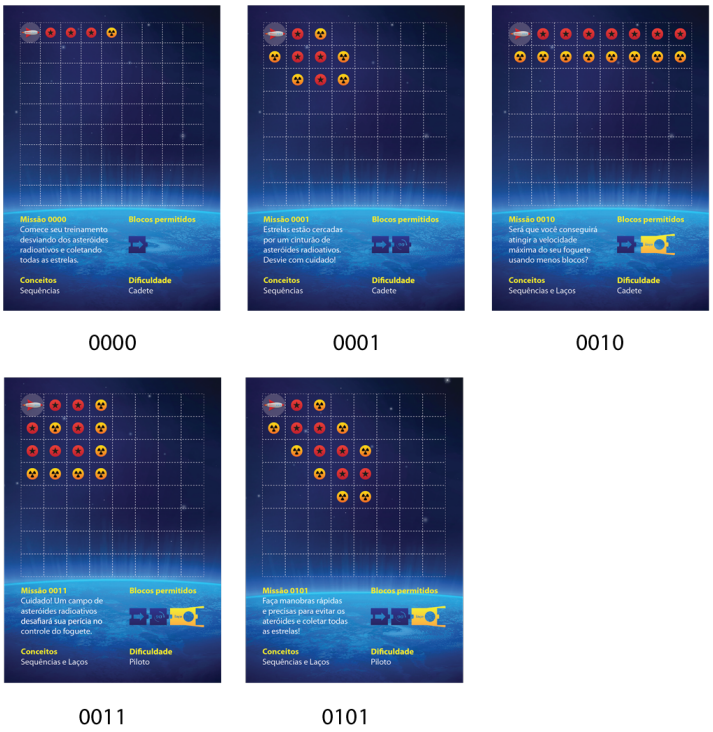


Figura 6.1 – Cartas utilizadas no teste, sendo 0011 e 0101 isomórficas quanto ao tipo de problema.

Após responderem as questões sobre as definições de comando, programa e linguagem de programação, os participantes foram indagados sobre a possibilidade de existir uma linguagem de programação específica para controlar um foguete que coleta estrelas e desvia de asteroides, tendo o tabuleiro como tela do computador de bordo. O objetivo

principal dessa questão foi relacionar as definições fornecidas anteriormente com o contexto de resolução de problemas que seria apresentado.

Para o início do estudo de caso propriamente dito, os participantes receberam uma breve explanação dos comandos e funcionamento da linguagem (Anexo VII), oferecendo um visão geral da relação entre o tabuleiro, os blocos de encaixe e o plano de voo. Na sequência, foi perguntado ao participante se desejaria ver uma situação-exemplo de uso dos comandos indicados na primeira carta (0000). Quando a resposta foi afirmativa, o tabuleiro foi arranjado para demonstrar a situação-exemplo quantas vezes o participante solicitasse, respondendo quaisquer dúvidas. Nos casos quando a resposta foi negativa ou após as demonstrações, o tabuleiro foi organizado conforme o diagrama indicado na carta. O desenho metodológico apresentado foi inspirado na técnica de fornecer *objetivos mais restrições* (Mosca, Silveira e Burigo, 1993, p. 62), que consiste em apresentar e repetir as restrições sintáticas e semânticas da linguagem quantas vezes forem necessárias, assegurando que o problema para o participante se mantenha no universo da coleta de estrelas e não na programação da linguagem.

Após a conclusão da tarefa da primeira carta, esta foi descartada e todo o procedimento foi repetido mais quatro vezes. Os planos de voo utilizados em cada um dos cinco problemas de cada sessão foram devidamente identificados. Todas as sessões foram gravadas em áudio e vídeo, utilizando uma câmera posicionada sobre um tripé que capturava todo o espaço de trabalho do participante – tabuleiro, blocos de comando ou folhas para anotação dos planos de voo (Fig. 6.2 e 6.3). Uma segunda câmera foi utilizada pelo experimentador para registrar movimentos do participante em outras perspectivas (Fig. 6.4).

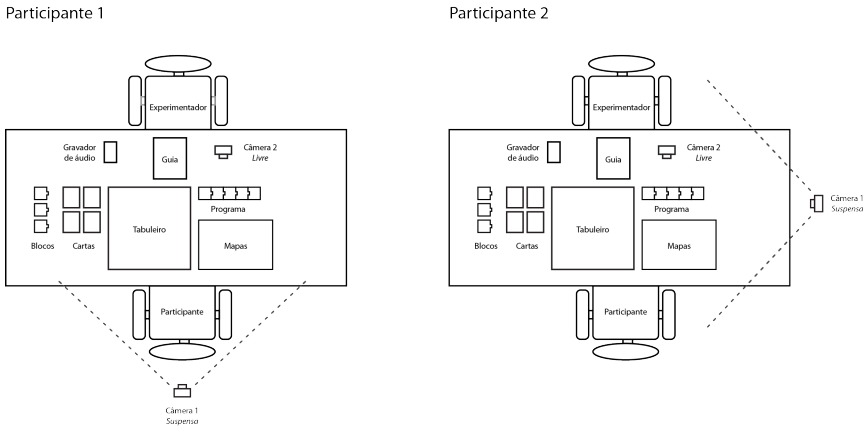


Figura 6.2 – Configuração do teste

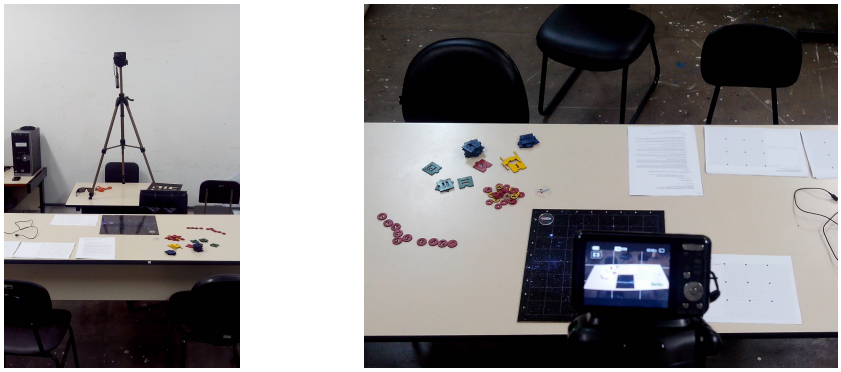


Figura 6.3 – Posicionamento e visão da câmera 1 (suspensa)

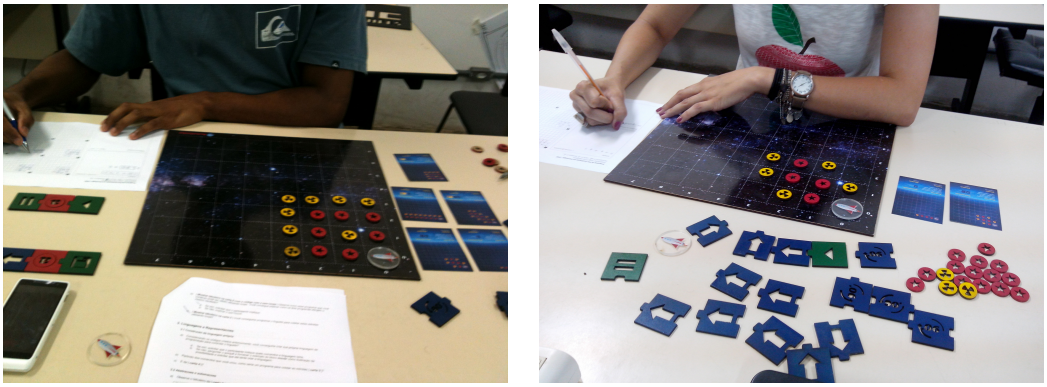


Figura 6.4 – Visões da câmera 2 (livre)

Procedimentos de análise

A descrição dos protocolos dos percursos dos dois perfis prototípicos seguiu as recomendações de Saada-Robert (1997, p. 163; 1997, p. 188) e Eichler e Fagundes (2001, p. 509), buscando marcar as diferenças nas unidades significativas de ação (Blanchet, 1997, p. 126) que poderiam estar relacionadas às diferentes representações sobre a Computação e seus princípios:

- *Unidades procedimentais*, referentes ao controle da atividade prática direta do participante sobre os blocos de comando, nos planos de voo, na composição de expressões válidas no âmbito da linguagem para controlar o foguete coletando estrelas e desviando dos meteoros.
- *Unidades representativas*, cuja *função causal* por um lado relaciona-se às transformações no estado do tabuleiro pela manipulação dos elementos primitivos da linguagem tendo seus meios de combinação e abstração como sistema de regras; e por outro sua *função teleonômica* diz respeito ao plano de ação organizado em etapas percebidas como possíveis tendo o trajeto apropriado do foguete como finalidade.

As unidades representativas foram observadas principalmente por meio dos planos de voo registrados nas folhas impressas, uma vez que os quadros que representam os estados do tabuleiro a cada passo na execução do algoritmo funcionam como objetos para pensar o problema. Já as unidades procedimentais foram observadas tanto na seleção dos comandos e composição do código no plano de voo ou com os blocos de encaixe, enquanto o participante manipulava suas alternativas para resolver cada passo do problema introduzido pelo trajeto da carta em questão.

Os protocolos individuais dos participantes, coletados em áudio, fotos e por meio dos planos de voo impressos foram analisados e excertos de cada tipo de registro foram utilizados no relato dos resultados conforme sua relevância para a análise dos procedimentos e estratégias identificados. A análise dos vídeos seguiu os passos recomendados por Meira (1994):

a) Visualizações completas e sem interrupções dos vídeos, realizando anotações preliminares relacionadas ao problema de pesquisa;

b) Produção de um *índice de eventos* que permitiu o acesso mais rápido aos segmentos específicos do vídeo;

c) Identificação dos eventos relacionados ao problema de pesquisa a partir do índice de eventos, dando origem ao trabalho interpretativo;

d) Transcrição literal dos eventos selecionados com o maior número de detalhes possível;

e) Mais visualizações dos segmentos apoiadas pelas transcrições visando construir interpretações plausíveis dos microprocessos envolvidos;

f) Inclusão de ilustrações dos exemplos prototípicos retirados dos vídeos e excertos das transcrições na comunicação dos resultados, auxiliando na compreensão da interpretação proposta pelo pesquisador.

6.1 – Resultados

Definições iniciais – Participante Programador

O primeiro participante, programador doravante denominado *P1* (M, 25 anos), ingressou no curso de Design da UFES em 2007. Começou a utilizar os computadores por volta dos 10 anos, tendo como principais atividades trabalhar, realizar pesquisas e lazer. Relatou ter aprendido a programar aos 20 anos, sendo mais fluente na linguagem de programação PHP, com a qual desenvolve sistemas de gerenciamento. P1 informou ter amigos, colegas

de faculdade e de trabalho que também são programadores e explicou que seu interesse por aprender a programar veio da busca por entendimento do relacionamento entre interfaces gráficas digitais e os códigos que as fazem funcionar. Para o participante, aprender a programar o ajudou a atingir níveis mais aprimorados para solucionar problemas gráficos e de interação com o usuário.

As evocações de P1 sobre os Princípios da Computação indicam que o participante compartilha as mesmas RS identificadas no capítulo 4 para os demais estudantes programadores, especialmente no que tange à abrangência da representação da Computação não necessariamente objetivada nos computadores pessoais. P1 também compartilha representações sobre os processos de automação, abstração e sobre o conceito de algoritmo semelhantes àqueles sintetizados na Tabela 4.25[61]. Por fim, o participante respondeu que a Computação o ajuda na solução de problemas de Design acadêmicos, profissionais ou do cotidiano por meio do raciocínio lógico, enquanto os computadores seriam um suporte para o uso de programas específicos que desempenham tarefas específicas. Assim, P1 manifestou nas respostas questões identificadas nas sínteses anteriores que distinguem o *potencial representacional* da Computação e o uso instrumental dos computadores, na visão dos estudantes de Design que sabem programar.

O estudo de caso com P1 durou aproximadamente 1h30 para responder as questões iniciais e resolver os cinco trajetos. Segundo P1, um comando em um computador "seria um input de alguma tarefa que você queira realizar e que o computador vai te responder de alguma forma" e forneceu a abertura de um programa por meio do clique em ícones ou atalhos como exemplo. Clicar no ícone seria um comando dado ao computador, cuja resposta seria a abertura do programa desejado. No que diz respeito à definição de um programa, P1 explicou que seria como uma plataforma com objetivos específicos que traria algum retorno. O exemplo fornecido foi um programa de desenho, cujas ferramentas proporcionam condições de um desenho ser feito, tratado, salvo ou exportado. Um programa teria um "começo e um fim bem definidos do que você quer fazer e do que aquele programa executa especificamente".

Quanto à definição de uma linguagem de programação, P1 disse que se trata de um meio de construir um programa. Este teria uma tarefa a executar e tal tarefa seria construída utilizando linguagens de programação. Os exemplos fornecidos pelo participante são da sua experiência de programação de aplicações para a Web utilizando a linguagem PHP.

[61] Ver página 102

Sobre a possibilidade de existir uma linguagem de programação capaz de controlar o foguete nos termos descritos pelo pesquisador, P1 disse acreditar que as abstrações das linguagens permitiriam que um computador específico fosse programado para controlar outro, como o do foguete.

P1 – Sequências (cartas 0000 e 0001)

A primeira carta consiste num trajeto simples em linha reta, com três estrelas a serem coletadas. No plano de voo (Fig. 6.5), P1 construiu corretamente a sequência de comandos e os estados do tabuleiro até a solução, utilizando a letra F para indicar a posição do foguete, a letra O para os asteroides e asteriscos para as estrelas. O participante perguntou ao experimentador se deveria desenhar o quinto estado do tabuleiro, uma vez que não haveria diferença no arranjo do quarto estado como consequência da execução do bloco de comando que desliga o motor do foguete. O experimentador perguntou se o mesmo havia sido feito para o estado inicial do tabuleiro, que representa o comando ligar o motor, e o participante disse que sim e que faria o mesmo para o estado final.

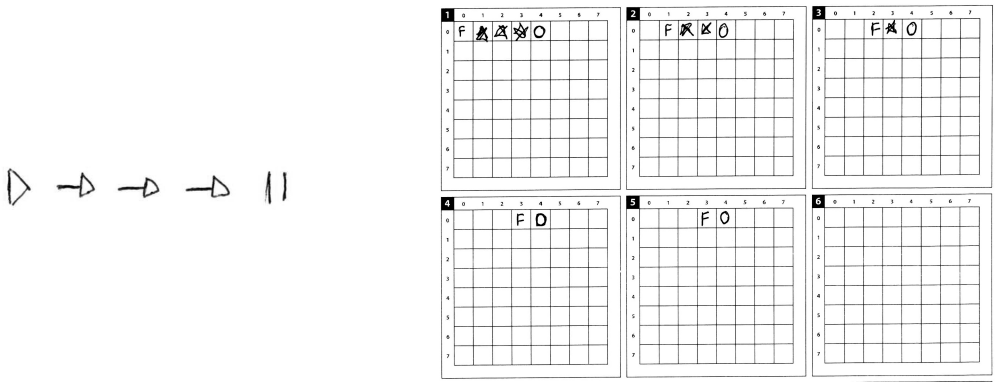


Figura 6.5 – Programa e estados do tabuleiro desenhados por P1 para a carta 0000

A segunda carta implica na utilização do bloco de comando *Girar 90º*, que não foi inicialmente apresentado a P1. O experimentador indagou se seria possível solucionar o trajeto utilizando os comandos apresentados até então, com o intuito de investigar se P1 partiria dos procedimentos aprendidos na situação anterior ou da estrutura mais geral da linguagem e da finalidade do problema para avaliar as possibilidades de ação. A resposta do participante foi afirmativa e o resultado do planejamento do trajeto no plano (Fig. 6.6) confirma que P1 assumiu que mudanças na orientação do foguete para as direções vertical e horizontal seriam possíveis na linguagem mesmo sem ter utilizado blocos que desempenhariam aquelas funções. Em outras palavras, o participante definiu sua primeira

solução para a carta 0001 a partir de uma representação *descendente* ou *final* do problema, ignorando as alternativas de ação válidas em termos de procedimentos.

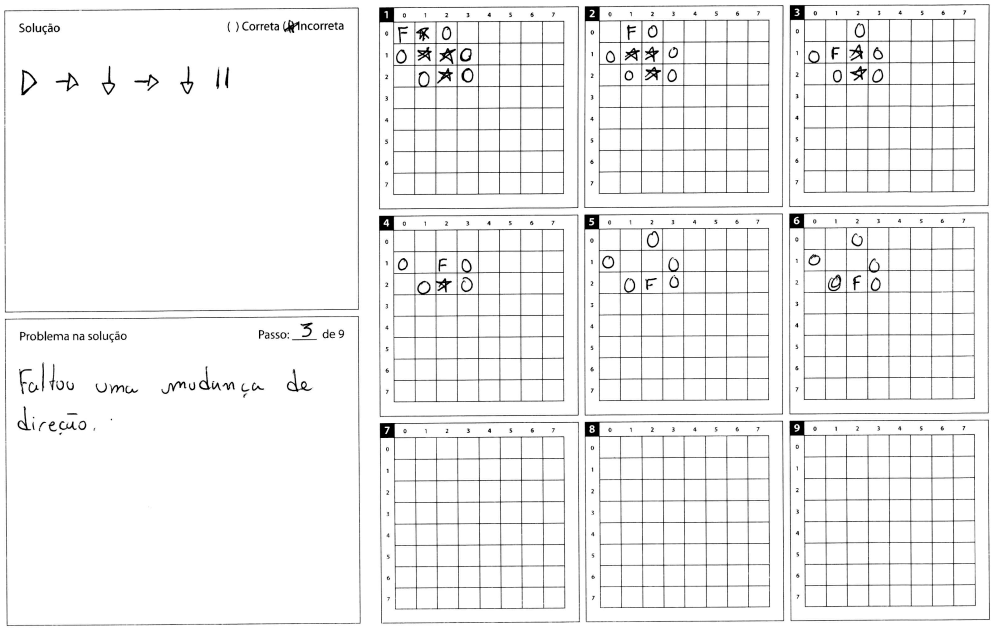
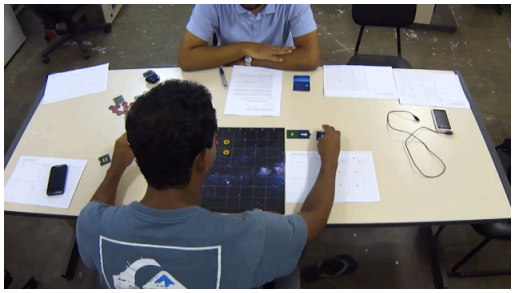
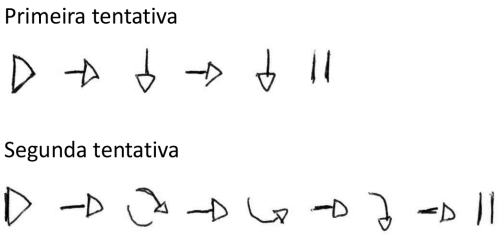


Figura 6.6 – Programa e estados do tabuleiro desenhados por P1 para a carta 0001

No momento da construção dos encaixes dos blocos de comando seguindo o planejamento do plano, P1 relatou (Fig. 6.7): "Pra baixo não dá. Não dá pra encaixar". Questionado se faltava alguma coisa para executar a solução planejada, o participante relatou que se tratava de "uma falta de encaixes": Um bloco de comando com a seta para baixo estaria faltando para executar sua solução. Nesse momento, P1 foi lembrado que o foguete só se move na direção para a qual está apontando e que só há um tipo de bloco de movimento, o que levou o participante à conclusão de que deveria girar o foguete para movê-lo na direção desejada. A mudança na representação da solução foi representada na segunda tentativa de resolver o problema no plano de voo (Fig. 6.8).





Figuras 6.7 e 6.8 – Momento da montagem do programa nos blocos quando o participante percebeu a impossibilidade de executar o plano de voo original para a carta 0001. A segunda tentativa incorporou o bloco *Girar 90º* como parte da solução.

Ainda no processo de representação dos estados do tabuleiro na segunda tentativa da carta 0001, P1 perguntou ao experimentador se seria necessário representar as consequências do bloco de comando *girar 90º* no plano, uma vez que o foguete continuaria na mesma posição. O experimentador reforçou que o plano deveria representar cada estado do tabuleiro e o participante optou por simplesmente repetir o estado anterior.

A principal questão observada neste momento tem relação com o uso da letra *F* como forma escolhida por P1 para indicar a posição do foguete no plano de voo. Em nenhum momento o participante girou a letra para indicar a direção do foguete, o que pode ter dificultado a construção da correspondência visual entre a representação do tabuleiro e o estado real das peças. Na execução passo a passo do código escrito com os blocos de comando, o participante conferiu sucessivas vezes se o número de estados do plano de voo correspondeu à quantidade de etapas necessárias para a solução do trajeto no tabuleiro. Em outros termos, frente às diferenças entre a representação dos estados do problema e a situação física das peças no tabuleiro, o participante exibiu controle de funcionamento *causal* na execução do código, por mais que o plano de voo estivesse previamente resolvido em sua função teleonômica.

Esse evento pode indicar que, mesmo entre os aprendizes com experiência em linguagens de programação, o plano de voo representado nas folhas funciona como um objeto que exige mudanças de controle cognitivo final-causal e vice-versa, permitindo ao tutor-experimentador acompanhar o desenrolar da solução.

P1 – Comandos, programas e linguagens

Após a experiência com as duas primeiras cartas, P1 foi questionado se as definições de comando, programa e linguagem de programação fornecidas anteriormente estariam presentes nas tarefas que acabara de executar. O participante respondeu positivamente, explicando que

> Sim, porque tem um começo e um fim, digamos assim. Ele tem uma instrução inicial, que seria coletar as estrelas. Ele tem um processo, que é como as estrelas serão coletadas e qual o caminho que o foguete iria percorrer. E tem um fim, bem definido.

Para P1, os comandos seriam o conjunto de instruções completas para o trajeto, enquanto o programa seria o funcionamento do comando, ou "o comando colocado em prática". Já a linguagem seria o que sustentaria o programa – "seu suporte, como ele é construído". O participante enfatizou que a construção do programa usando a linguagem dependeria das

instruções que são passadas ao foguete – ligar, andar para frente, mudar de direção e assim sucessivamente.

P1 – Laços (carta 0010)

Na construção da solução do trajeto para carta 0010 (Fig. 6.9), P1 precisou de duas tentativas. Na primeira, esqueceu de indicar qual comando seria executado pelo laço que coletaria as sete estrelas. O problema foi percebido quando P1 consultou os comandos que havia desenhado na área soluçado do plano de voo antes de finalizar a representação do primeiro estado do tabuleiro.

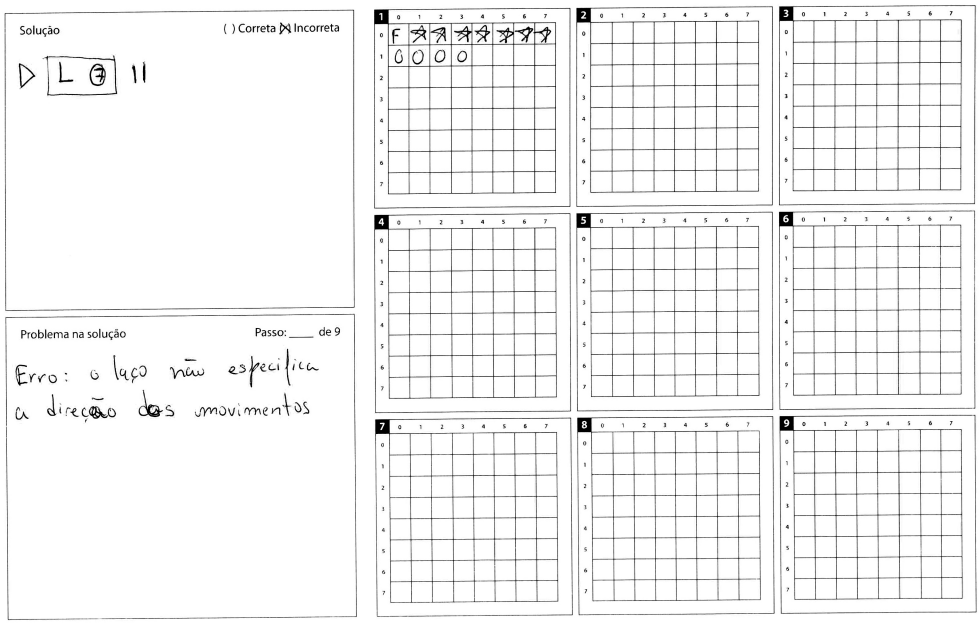
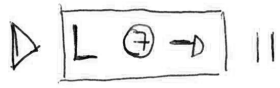


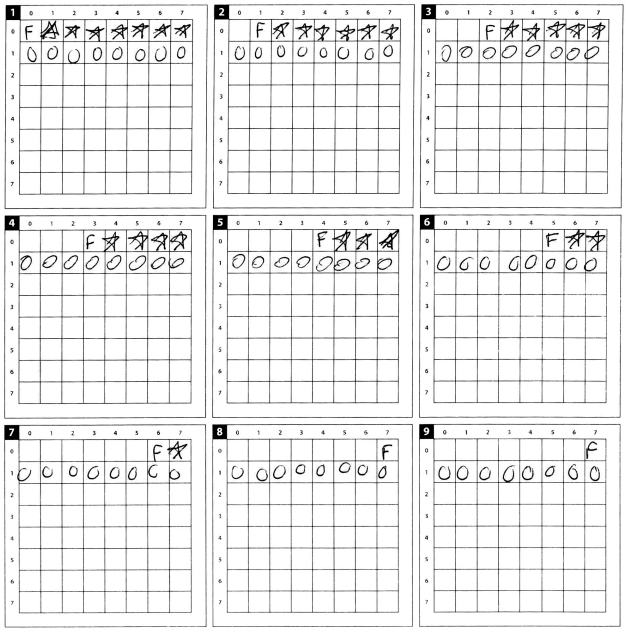
Figura 6.9 – Programa e estados do tabuleiro desenhados por P1 para a carta 0010

Na segunda tentativa (Fig. 6.10), o participante incluiu o comando de mover o foguete no interior do laço, completou o plano e construiu a expressão com os códigos corretamente. O retângulo que envolve a letra *L* e o número de repetições do laço no plano de voo, utilizado por P1 nas duas tentativas, tinha a função de "delimitar aquilo que se repete no laço". Essa preocupação pode estar ligada à experiência do participante com outras linguagens de programação, onde a definição do escopo (instruções restritas ao laço) é fundamental para o correto funcionamento dos algoritmos.

Primeira tentativa

Segunda tentativa





Figuras 6.10 e 6.11 – Programa corrigido e estados do tabuleiro na carta 0010

P1 – Do RocketSocket à linguagem própria (carta 0101)

A segunda parte do estudo com P1, referente à criação de uma linguagem do próprio participante para controlar o foguete, resultou nos comandos reproduzidos na Fig. 6.12. Durante a elaboração dos seus comandos, o participante perguntou se poderia explicar um *Laço* como "repetição de movimentos com quantidade finita". Ao ser questionado sobre o motivo de tal definição, P1 utilizou argumentos relacionados à necessidade das instruções serem específicas e bem definidas, "com início, meio e fim", assim como fez nas perguntas iniciais do estudo de caso para explicar o que seriam comandos, programas e linguagens de programação. Argumentos semelhantes também foram citados pelo participante nas questões sobre a possibilidade da *RocketSocket* ser efetivamente uma linguagem de programação, indicando que são conceitos importantes acerca da forma que P1 pensa a estrutura e funcionamento de uma linguagem de programação.

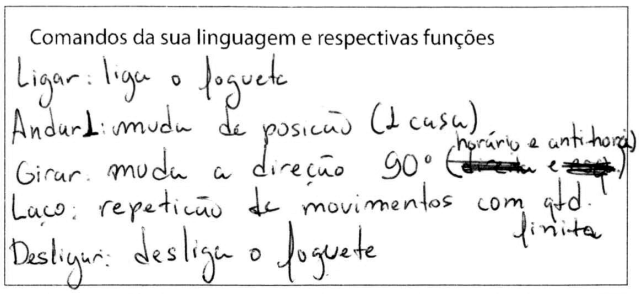


Figura 6.12 – Comandos criados por P1 para controlar o foguete

Na aplicação da linguagem própria, a carta 0101 foi utilizada como trajeto a ser construído. Optou-se pela inversão da sequência das cartas, 0101 antes da 0011, pois a última oferece mais oportunidades para que o aprendiz explore soluções criativas para o trajeto por meio de automações. O problema proposto pode ser resolvido com ou sem a utilização de laços, cabendo ao participante elaborar a solução conforme suas preferências. Durante o processo de planejamento do voo, P1 esboçou na mesa uma expressão baseada nos blocos de comando *RocketSocket* (Fig. 6.13).

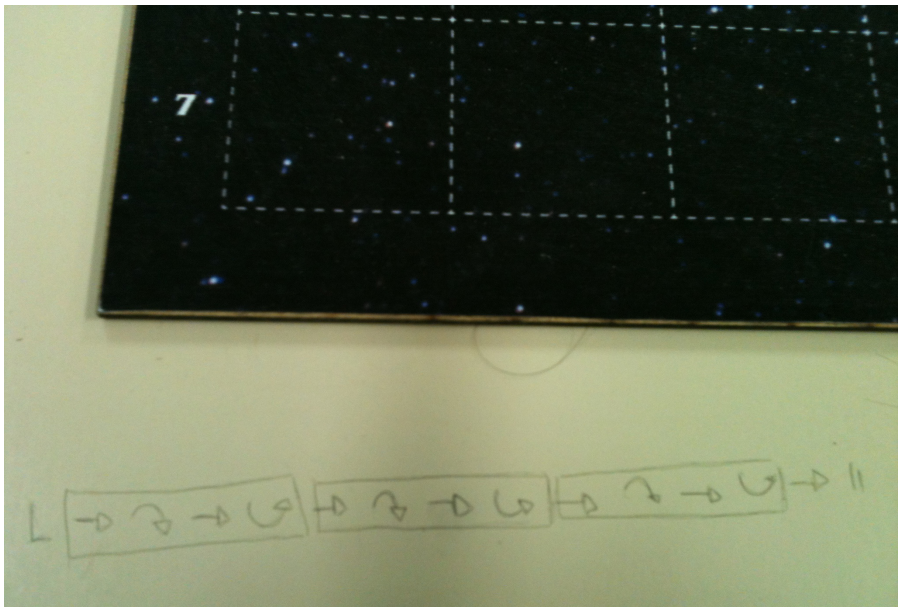


Figura 6.13 – Esboço feito por P1 usando a linguagem RocketSocket

Ao ser perguntado sobre o que estaria fazendo, P1 disse: "é um rascunho do meu pensamento, para não desenhar errado". Apesar de ter elaborado sua própria linguagem, o participante adotou a mesma notação dos problemas anteriores para esboçar a solução. A justificativa para essa conduta foi a sensação de mais facilidade para pensar o trajeto do foguete usando os blocos de comando, a forma de combinação e sequenciamento deles, para depois traduzir o código gerado para a nova linguagem criada. Neste ponto pode-se argumentar que os blocos de comando e seus meios de combinação, tanto nas peças de encaixe quanto no plano de voo, foram utilizados pelo participante como instrumentos mediadores do processo ou *objetos-para-pensar* (Saada-Robert, 1997, p.174) a construção da solução: Tendo o objetivo da tarefa definido, a lógica e regras de encadeamento dos blocos de comando em expressões válidas também parecem participar do controle cognitivo do participante em busca da solução, de maneira que o código do trajeto resultaria não apenas da representação apropriada da situação-problema, mas da

representação do problema nos termos da linguagem *RocketSocket*. No que tange ao programa desenhado no plano de voo (Fig. 6.14), duas questões merecem destaque:

I. O código foi escrito com sequência de comandos na orientação vertical, com as instruções escritas linhas a linha, sem nenhum tipo de instrução ou recomendação do experimentador a esse respeito. Essa prática é comum nas linguagens de programação cujo código é escrito em texto, o que pode explicar a escolha de P1;

II. P1 antecipou por iniciativa própria a identificação de padrões que poderiam gerar automações no código, mesmo percebendo que seria possível resolver o trajeto com sequencias simples de giros e movimentos do foguete: "É porque aqui eu consegui mapear de alguma forma uma otimização do meu programa, que seria a utilização de um laço, pois parece que existem movimentos recorrentes".

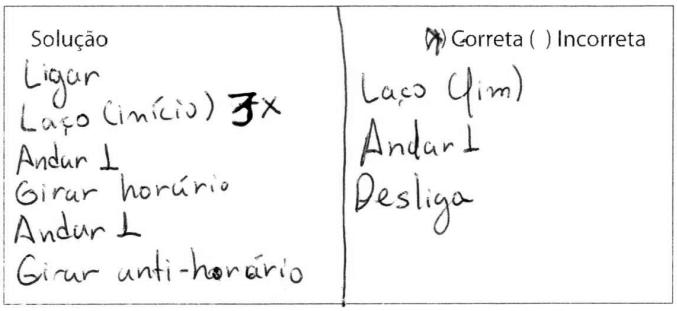


Figura 6.14 – Programa escrito na linguagem própria de P1

P1 – Automações e abstrações (carta 0011)

Considerando que P1 antecipou a identificação de oportunidades para automatizar partes do código, os meios de abstração da linguagem *RocketSocket* foram facilmente explicados. O experimentador utilizou a solução criada por P1 no trajeto da carta anterior para demonstrar o funcionamento do blocos de memória e abstração (Fig. 6.15).

O trajeto da carta 0011 é um problema isomórfico ao da carta 0101, com possibilidade de resolução com a mesma estratégia de giros e movimentos sequenciais ou pelo uso de laços que combinem giros e movimentos. Como esperado, P1 optou por identificar o padrão com possibilidade de automação no trajeto e o transformou na variável *a* que foi em seguida armazenada na memória e utilizada no interior de um laço (Fig. 6.16).

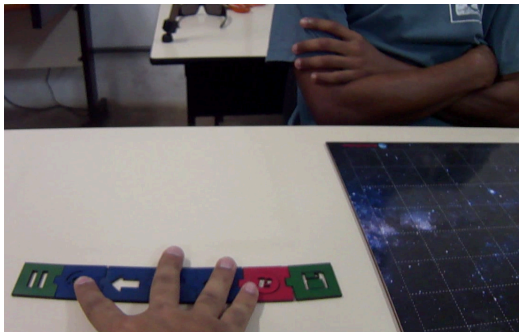


Figura 6.15 – Explicação sobre o funcionamento da memória do foguete e meios de abstração

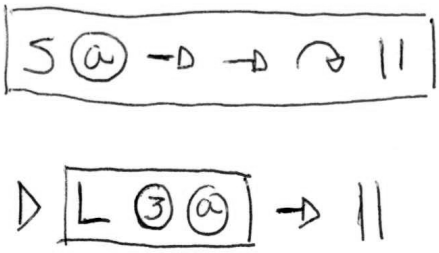


Figura 6.16 – Programa para a carta 0011 utilizando abstrações e memória

O desenrolar da solução se estendeu por duas páginas (Fig. 6.17), mas sem dificuldades de representação pelo participante. A simplicidade do código construído com os blocos de encaixe foi descrita por P1 como fácil de montar, justamente pela redução considerável no número de instruções em decorrência da criação da abstração.

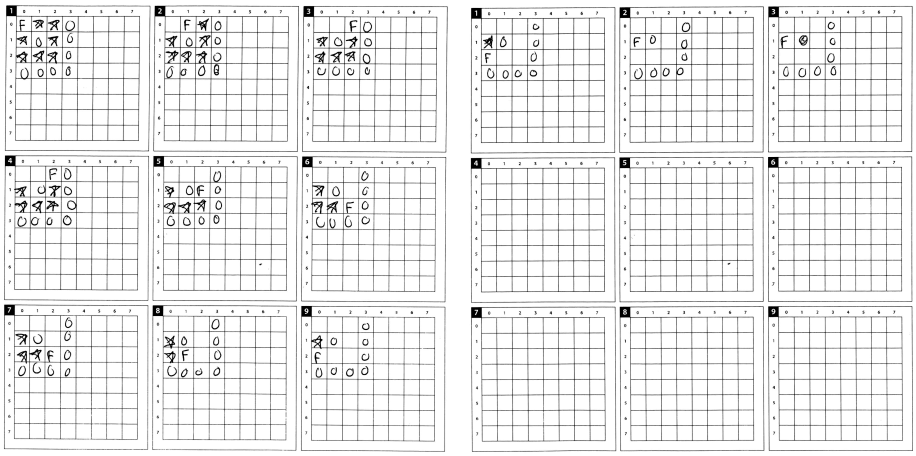


Figura 6.17 – Estados do tabuleiro representados no plano de voo (duas páginas)

Definições iniciais – Participante Não-Programadora

A segunda participante, P2 (F, 24 anos), ingressou no curso de Design da UFES em 2009. Começou a utilizar os computadores por volta dos sete anos e citou e-mails, pesquisas trabalhos de faculdade e entretenimento como principais atividades realizadas com os dispositivos. De maneira geral, as evocações da participante indicam compartilhamento das RS sobre os Princípios da Computação com o grupo dos não-programadores, especialmente quanto aos conceitos abordados pelo *RocketSocket*: computação, abstração, automação e algoritmo. Na solução de problemas de Design acadêmicos, profissionais ou do cotidiano, P2 relatou que a Computação a auxilia agilizando processos manuais e permitindo a busca e troca de ideias, em sintonia com respostas amplamente discutidas

quanto os usos instrumentais da Computação no capítulo 4 desta tese. Seguindo essa visão, P2 informou que os computadores seriam as ferramentas que a auxiliam diariamente no processo de comunicação e pesquisa de informações do interesse dela.

O estudo de caso com P2 durou aproximadamente 1h15 para responder as questões iniciais e desenvolver a solução para os cinco trajetos apresentados. Para a participante, um comando "é algo que ativa uma função" e forneceu como exemplos as ações do teclado desencadeando resultados na tela – "seta para a direita movendo objetos para a direita" – ou botões em jogos que desencadeiam ações nos personagens. Já um programa seria "onde tem todos os comandos e cada comando quer dizer uma coisa e nesse programa vou realizar uma atividade". P2 sugeriu que o programa seria como uma caixa, onde os comandos estariam localizados, e como exemplo citou o Microsoft Word com os comandos de formatação de texto em negrito, itálico, entre outros. Por fim, uma linguagem de programação seria diferente da linguagem utilizada na comunicação verbal, mas haveria semelhanças para que as pessoas entendam os códigos. P2 explicou que, para ela, linguagens de programação seriam uma série de signos, símbolos, funções e dados que juntos formariam uma determinada função ou atividade. A participante não soube exemplificar uma linguagem de programação específica, embora tenha citado "zeros e uns" (linguagem binária) como um exemplo de linguagem "muito técnica".

Ao ser indagada sobre a possibilidade de existir uma linguagem de programação para controlar o foguete no contexto apresentado, P2 disse que se o foguete for uma máquina ele pode ser controlado por outra máquina. A participante explicou que poderia haver um sistema com diversos comandos e fiações conectando tudo ao motor do foguete, tornando o controle pela linguagem de programação possível.

P2 – Sequências (cartas 0000 e 0001)

Na resolução da primeira carta (Fig. 6.18), P2 definiu a letra *F* para indicar a posição do foguete nos estados do tabuleiro, *A* para os asteroides e fez desenhos das estrelas. Assim como o participante P1, P2 perguntou sobre a necessidade de representar o quinto estado do tabuleiro e foi instruída a refletir sobre o que havia feito no primeiro. A participante decidiu então representar o último estado, onde o motor estaria desligado.

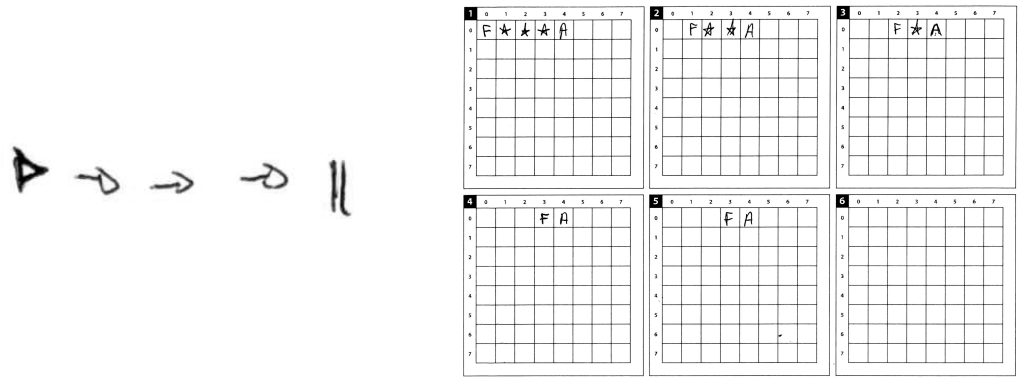


Figura 6.18 – Programa e estados do tabuleiro desenhados por P2 para a carta 0000

A construção do trajeto da carta 0001, que depende do uso dos blocos de comando de giro até então indisponíveis para a participante, foi percebida como possível por P2 da mesma forma que P1. A participante realizou todo o planejamento do voo (Fig. 6.19) e só percebeu a impossibilidade de construir o programa com os blocos de encaixe no momento em começou a composição da expressão (Fig. 6.20): "Pra baixo não dá, não dá pra encaixar."

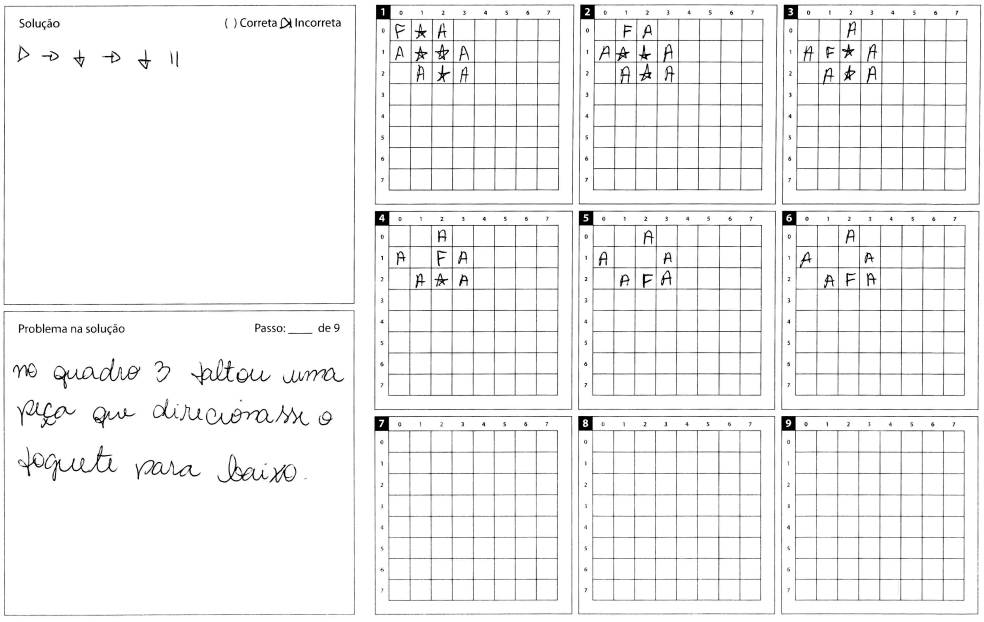
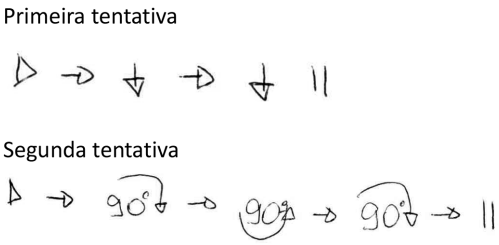


Figura 6.19 – Programa e estados do tabuleiro desenhados por P2 para a carta 0001

Ao ser questionada sobre o que estaria faltando para resolver o problema, P2 primeiro disse não saber e depois disse que seria uma peça com encaixe para baixo. Na sequência, foi lembrada do modo de voo do foguete, orientado pela direção do bico, e concluiu que deveria girar o foguete para a direção certa. O experimentador apresentou os blocos de *Giro 90º* nos sentidos horário e anti-horário e perguntou se resolveriam o problema. A

participante então elaborou o programa da Fig. 6.21 e passou para a representação do plano de voo correspondente.



Figuras 6.20 e 6.21 – Tentativa da participante P2 de encaixar o bloco de movimento na direção vertical e programa corrigido após a apresentação do bloco girar.

No processo de representação dos efeitos do primeiro giro no plano de voo, P2 apresentou dúvidas sobre o sentido correto para girar o foguete. O experimentador sugeriu que a participante girasse o próprio foguete sobre o tabuleiro para auxiliá-la na decisão e a dúvida foi solucionada. A dificuldade de estabelecer correspondências entre o código e a situação do tabuleiro pareceu semelhante àquela demonstrada por P1, também durante os usos dos blocos de comando de giro do foguete: Mesmo com o trajeto definido pelo código desenhado no plano de voo, a participante retomou procedimentos locais, utilizando objetos que a auxiliaram na avaliação das suas decisões antes de passar aos próximos estados da solução.

Tais observações reforçam o potencial oferecido tanto pelo plano de voo quanto pelas peças físicas enquanto objetos mediadores do processo de aprendizagem da lógica de operação da linguagem e dos Princípios da Computação, mesmo entre estudantes sem experiência de programação. O problema de compreender o resultado do giro do foguete em termos puramente trigonométricos parece ter sido deslocado para uma situação contextualizada, onde ângulos são mapeados para direções e ações de um objeto concreto. A tomada de decisão pelo aprendiz pode partir inicialmente dos aspectos físicos e icônicos (a orientação do desenho do foguete sobre o tabuleiro) para gradualmente migrar para questões mais abstratas que poderiam abordar diretamente os conceitos computacionais envolvidos: Quantos giros horários ou anti-horários de 90º sucessivos seriam necessários para se obter determinada direção e como construir esse movimento corretamente utilizando os comandos da linguagem.

P2 – Comandos, programas e linguagens

Partindo da conclusão dos trajetos das duas primeiras cartas, P2 foi perguntada se as definições que ela havia dado para comandos, programas e linguagens de programação estariam presentes nas soluções construídas por ela com os blocos de comando. Para a participante, a linguagem de programação estaria presente pois haveria "ícones" com significados específicos, como por exemplo ligar o motor e mover o foguete. Questionada se tais significados seriam comandos, P2 concordou dizendo que seriam "comandos de andar pra frente", apontando para o bloco de movimento, e concluiu dizendo que "isso tudo (os blocos) é uma linguagem que faz o programa acontecer".

Por mais que esta última afirmação forneça indícios de que P2 conseguiria pensar as distinções entre comandos, programas e a própria linguagem de programação de forma semelhante a P1, a explicação da participante sobre a relação entre os três conceitos sugere uma visão mais integrada[62] do conjunto. Na visão de P2, o programa de fato seriam os blocos de comando, que fazem parte de uma linguagem, "acontecendo na tela" (tabuleiro). Em outras palavras, um programa é algo vivo, em execução em algum lugar, assim com os exemplos fornecidos pela participante no início do estudo de caso – Microsoft Word e jogos. Como consequência dessa visão, P2 relatou que entende que o conjunto apresentado seria uma linguagem de programação com comandos capazes de gerar programas que controlariam o foguete da forma exibida no tabuleiro.

P2 – Laços (carta 0010)

Durante a construção do trajeto para a primeira carta que utilizou *Laços*, P2 questionou-se como representar os estados do tabuleiro gerados por aquele novo bloco no plano de voo: apenas dois estados, representando o início e o fim do *Laço*; ou sete estados, referentes a cada uma das transformações no tabuleiro desencadeadas pelo *Laço*. Essa dúvida, que não apareceu no caso de P1 provavelmente em decorrência da experiência acumulada por aquele participante utilizando laços em outras linguagens de programação, está associada a ideias fundamentais sobre estruturas de controle do fluxo do programa.

Embora a explicação do experimentador tenha abordado explicitamente a equivalência de resultados entre sequências de comandos repetidos e laços com o mesmo comando quanto à montagem do código com os blocos (Fig. 6.22), P2 pareceu não compreender que os estados do tabuleiro também seriam idênticos para as duas possibilidades de escrita do

[62] A palavra integrada, nesse contexto, não reflete um entendimento superior dos comandos, programas ou linguagem por P2. Trata-se de uma visão que distingue com menos clareza os elementos, onde um é conceituado sempre em função do outro.

programa. A repetição do exemplo de funcionamento dos laços utilizando os blocos sanou as dúvidas de P2 (Fig. 6.23), que conseguiu representar corretamente os estados no plano.

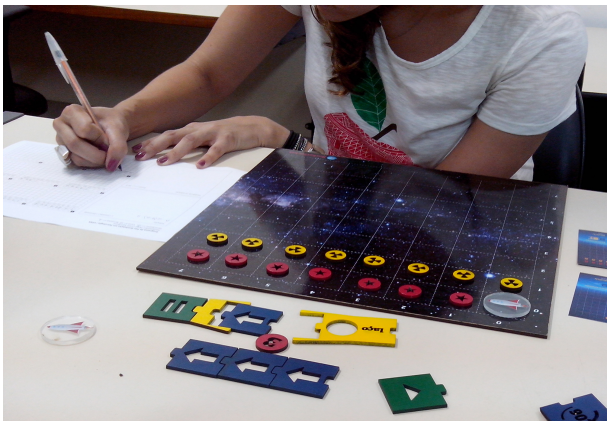


Figura 6.22 – Explicação do experimentador para P2 sobre a equivalência entre sequências de comandos e de comandos no laço.

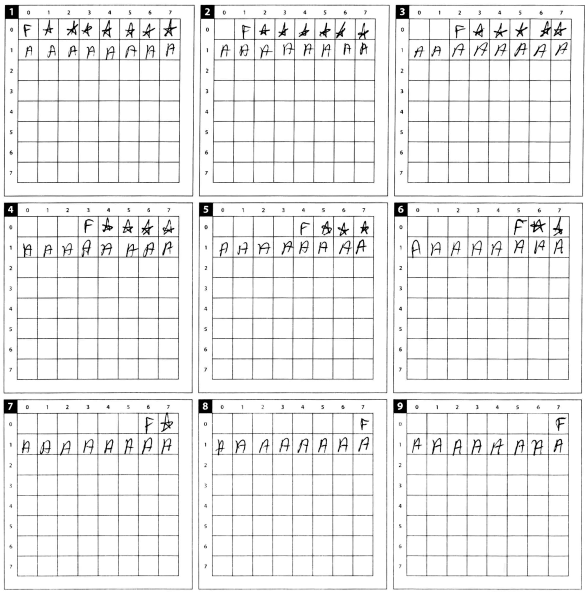


Figura 6.23 – Plano de voo relacionado à Figura 6.22, na qual os sete estados do laço estão representados além dos estados inicial e final.

P2 – Do RocketSocket à linguagem própria (carta 0011)

Na segunda parte do estudo de caso, a participante foi perguntada sobre a possibilidade de construir sua própria linguagem de programação para controlar o foguete, com o mesmo funcionamento e possibilidades oferecidas pelos blocos de comando. No primeiro momento, P2 respondeu não conseguir realizar a tarefa e que, para ela, seria mais fácil aprender uma linguagem do que criar uma nova. O experimentador enfatizou que a linguagem não deveria fazer nada além do que P2 havia feito até o momento com os blocos

de comando e nos planos de voo. A participante afirmou então que a tarefa seria possível se a linguagem fosse direcionada apenas a fazer o que ela já havia feito no estudo de caso. Os comandos criados por P2 (Fig. 6.24) com suas respetivas justificativas foram:

- *Start*, representado por um sinal de visto, com função de ligar o motor;
- *Multiplicação*, representado pelo sinal de multiplicação, com a mesma função do *Laço*. Segundo P2, as pessoas estão acostumadas a encontrar esse sinal para indicar multiplicações, então faria sentido utilizá-lo em coisas que se repetem múltiplas vezes como o *Laço*;
- *Segue*, escrito por extenso, indicando o movimento do foguete. P2 esclareceu que como não há direção pré-definida no comando, "seguir" significaria caminhar na direção atual;
- *Finalizar*, escrito por extenso, com a função de desligar o motor, e que nas palavras de P2 "seria equivalente ao *pause"* do *RocketSocket*.
- *Números*, representados pelos algarismos, que seriam utilizados nos laços, mas que não foram representados pela participante no plano de voo;
- *Girar*, escrito por extenso, com função de girar o foguete. O experimentador perguntou se apenas um comando girar seria suficiente para resolver todos os problemas e P2 disse que seria necessário ter o comando *Girar* seguido do "lado" – direita, esquerda, cima e baixo – sempre em 90º.

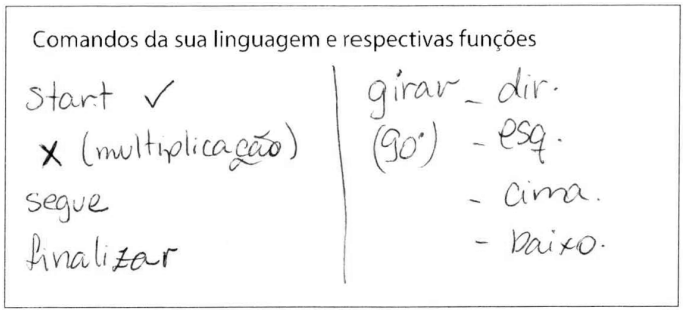


Figura 6.24 – Comandos da linguagem criada por P2.

Diferentemente de P1, a carta escolhida para a aplicação da linguagem construída por P2 foi a 0011 ao invés da 0101. Essa escolha se deu em decorrência da insegurança da participante frente à tarefa de construir e utilizar sua própria linguagem. Embora os problemas das cartas sejam isomórficos, baseados no uso de giros e voos alternados, com ou sem laços, a tarefa da carta 0011 (ver p.137) é *visualmente* mais simples.

No planejamento do trajeto (Fig. 6.25), P2 escreveu os comandos como se fossem palavras em frase, diferentemente do arranjo utilizado por P1. O fato da participante não ser programadora e não conhecer as práticas comuns na escrita de código em outras linguagens de programação pode explicar a escolha por tal encadeamento dos comandos. A linguagem *RocketSocket* utiliza meios de combinação dos elementos primitivos estruturados horizontalmente, o que pode ter servido de referência para a participante, além da própria linguagem escrita alfabética.

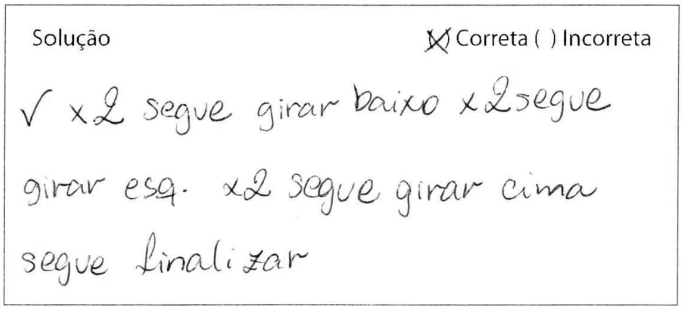


Figura 6.25 – Programa escrito por P2 utilizando a linguagem própria.

Ainda sobre o trajeto, um último ponto pode ser discutido: Apesar da repetição explícita de comandos decorrentes da estrutura do problema da carta 0011, a participante optou por elaborar um código com três laços para solucionar em separado os subproblemas de mover o foguete (*x2 segue*) e não percebeu a possibilidade de utilizar um único laço para solucionar todo o trajeto[63]. A solução apresentada mantém o controle dos avanços rumo ao objetivo final da tarefa no sentido ascendente, orientado mais por decisões locais das partes do problema a serem resolvidas do que por um plano mais global que articula as possibilidades da estrutura da linguagem e a natureza do trajeto.

P2 – Automações e abstrações (carta 0001)

A estratégia de apresentação dos conceitos de automação, abstração e o funcionamento do bloco de memória foi baseada no trajeto construído pela participante para a carta 0011. O experimentador solicitou a P2 (Fig. 5.26) que procurasse elementos ou conjuntos de elementos que se repetiram no código e que geraram resultados semelhantes no plano de voo e tabuleiro.

Na primeira análise, P2 concentrou-se em identificar elementos primitivos que se repetem ao invés de sequências de comandos cujo funcionamento gerou resultados repetidos:

[63] Para uma solução alternativa utilizando apenas um laço, checar o código elaborado por P1 nas páginas 147-148.

Ah, repete o de seguir e o de girar. *[Onde que está isso? Me mostre.]* Quando ele vai pra cá [*apontando o código e o estado correspondente no plano*], ele segue, segue... Quando ele gira, gira mais de uma vez... Gira três vezes. Um, dois, três [*apontando para o plano*].

As repetições identificadas sugerem que a busca da participante por padrões ainda estava concentrada nos elementos primitivos isoladamente, não no seu encadeamento rumo à solução do problema do trajeto. De certa forma, P2 entendeu a recorrência de elementos no código em termos concretos, ou seja, pela repetição numérica dos blocos de comando ou das suas ações relacionadas nos estados do plano de voo.

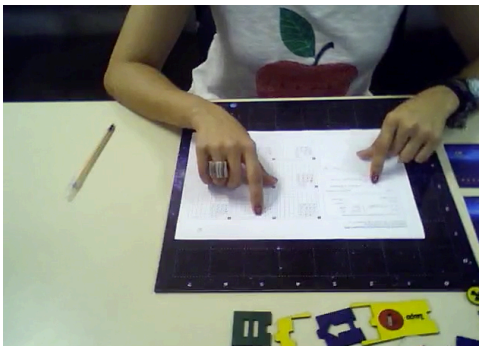


Figura 6.26 – Repetições identificadas por P2 no código e estados do plano de voo.

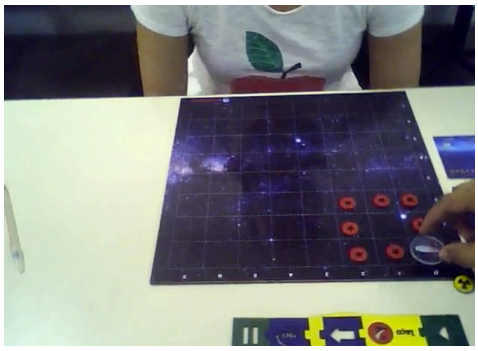


Figura 6.27 – Explicação do tutor sobre a repetição da sequência de comandos no trajeto.

Figura 6.28 – Explicação do bloco de memória e do conceito de automação no trajeto da carta 0011

A intervenção do experimentador (Fig. 6.27) concentrou-se então em explicar à participante que haveria repetições também nas sequências de comando, uma vez que partes do trajeto realizado pelo foguete demandavam manobras semelhantes. Partindo dessa explicação, P2 analisou novamente os estados no plano e identificou o padrão recorrente no trajeto: “Anda duas vezes e gira pra direita três vezes... Tudo três vezes”. Uma vez compreendida a questão, o experimentador apresentou o funcionamento do bloco de memória e os meios de abstração da linguagem *RocketSocket*, nomeando o código

identificado pela participante como "*a*" e utilizando o conjunto para resolver novamente a situação-problema da carta 0011 (Fig. 6.28).

Finalmente, para que a participante experimentasse o uso do bloco de memória e o conceito de abstração, o experimentador pediu que a mesma escolhesse alguma das cartas cujo trajeto já havia sido resolvido ou avaliasse a última carta não utilizada (0101). P2 optou por analisar as cartas e planos de voo já construídos, escolhendo a carta 0001 para reconstruir o código (Fig. 6.29) cujo padrão seria: "Girar horário, girar anti-horário, girar horário. Girar horário, pra frente. Repete duas vezes".

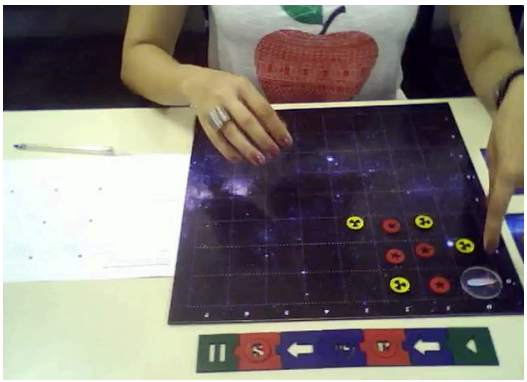


Figura 6.29– Abstrações e uso do bloco de memória para a carta 0001

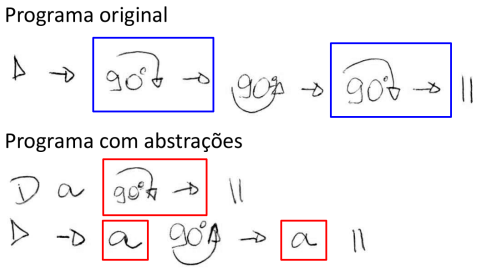


Figura 6.30 – Programa original para a carta 0001 e a versão reescrita com abstrações

O programa reconstruído pela participante armazenou na memória (Fig. 6.30, linha iniciada por *D*) abstrações para combinar blocos de comando (*girar 90º no sentido horário e mover o foguete um passo*) em algo equivalente a um novo bloco, que por sua vez foi utilizado duas vezes no programa, intercalado com as outras operações necessárias para resolver o trajeto (Fig. 6.30, padrões indicados em azul, abstrações em vermelho). De forma diferente da tentativa inicial de identificar comandos isoladamente, P2 conseguiu reconhecer um padrão em termos de encadeamento. A utilização da abstração em eventos distintos do programa demonstrou compreensão da ideia de encapsulamento e reutilização de códigos, fundamental para as práticas do Pensamento Computacional.

6.2 – Programador versus Não-programador num Contexto Construcionista

O estudo de caso realizado com P1 e P2 teve como objetivo investigar se a proposta pedagógica elaborada ofereceria iguais oportunidades de aprendizagem dos Princípios da Computação para diferentes perfis de aprendizes, mais especificamente estudantes de Design com e sem conhecimentos prévios de programação. A hipótese que norteou a construção da linguagem *RocketSocket* assumiu que programadores e não-programadores adotariam estratégias distintas para solucionar problemas de natureza computacional,

partindo da relação entre as representações sociais (RS) compartilhadas por cada grupo acerca dos Princípios da Computação e as respectivas práticas com os computadores. Assim, tais condutas seriam observadas no processo de resolução dos problemas numa perspectiva funcional, que considera não apenas a estrutura do conhecimento utilizado no percurso (no caso, RS dos Princípios da Computação) como também sua função (como aquelas RS são aplicadas pelo indivíduo para compreender e tomar decisões nos diversos estados da tarefa até a solução).

A linguagem *RocketSocket,* portanto, deveria contribuir para a formação de uma zona de desenvolvimento proximal (ZDP), considerando diferentes perspectivas referenciais. Nesta ZDP, programadores e não-programadores poderiam partir das suas representações e práticas anteriores com a Computação e construir gradualmente novas estruturas de conhecimento acerca daquela área por meio da prática com os elementos primitivos, meios de combinação e abstração da linguagem na resolução dos problemas dos trajetos do foguete. Após a análise dos resultados dos estudos de caso dos dois participantes, quatro pontos destacam-se sobre as possibilidades da linguagem para que programadores e não-programadores possam pensar computacionalmente os problemas:

A linguagem como sistema de regras que orienta tanto a elaboração da representação do problema pelo aprendiz, quanto o processo de tomada de decisão no desenrolar da solução.

Tanto P1 quanto P2 exibiram condutas semelhantes quanto às expectativas de comandos existentes na linguagem (bloco mover para baixo na carta 0001). Embora o equívoco tenha sido intencionalmente induzido pelo experimentador, a representação da solução elaborada pelos participantes sugere definição da finalidade da tarefa a partir da estrutura de possibilidades da linguagem. P1 e P2 tinham todos os comandos em mãos, podendo adotar procedimentos apropriados em termos causais a partir da experiência concreta recente, mas construíram a estratégia de resolução a partir da lógica global da linguagem (há peças de movimento, então que haja uma de movimento vertical) e da própria temática da situação de aprendizagem (foguetes podem voar em qualquer direção).

As opções pela inclusão de determinados blocos de comando na linguagem ao invés de outros parecem ter auxiliado na interpretação dos problemas a partir de regras que, uma vez assimiladas pelo aprendiz, tornaram-se objetos do funcionamento cognitivo tanto quanto as próprias peças físicas. P1 demonstrou tal conduta em dois momentos: 1) ao identificar padrões no programa da carta 0101 para construí-los com laços mesmo sem o

incentivo do experimentador; e 2) ao utilizar a notação *RocketSocket* na mesa para pensar a solução do trajeto e depois traduzi-la para os comandos criados por ele.

A relação entre o programa desenhado no plano, a representação dos estados da solução, e a posterior construção do programa com os blocos de encaixe para execução no tabuleiro físico.

A construção dos programas como sequências de comandos que geram estados do trajeto que se transformam gradualmente no plano de voo se apresentou como um segundo objeto para que os aprendizes pensem os problemas. Quatro tipos de episódios experimentados pelos participantes foram muito importantes para tal conclusão, em especial: 1) como proceder na representação do estado inicial e final dos trajetos no plano de voo (P1 e P2); 2) a necessidade de representar ou não no plano estados decorrentes do comando *Girar,* que não modifica a posição de nenhum elemento do tabuleiro (P1); 3) a quantidade de estados a serem gerados pela execução do *Laço* (P2); 4) a comparação passo-a-passo dos estados do plano de voo com a respectiva execução no tabuleiro físico como estratégia de conferência do programa escrito (P1 e P2).

Episódios como esses poderiam ser utilizados em discussões sobre os processos de análise e síntese no Design Computacional discutidos por Christopher Alexander (1964): desmembrar o problema em subproblemas em sentido descendente para sintetizar subsoluções em sentido ascendente que enfim resolverão o problema como um todo. Cada estado do tabuleiro representado no plano seria isomórfico a um nível da árvore da solução proposta por Alexander, ou fragmentos do programa seriam combinados em conjuntos aninhados até atingir a solução final para o trajeto (Fig. 6.31 e 6.32).

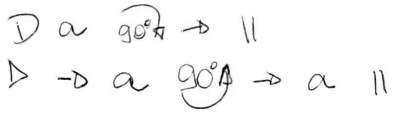


Figura 6.31– Programa elaborado por P1 utilizando abstrações para a carta 0001

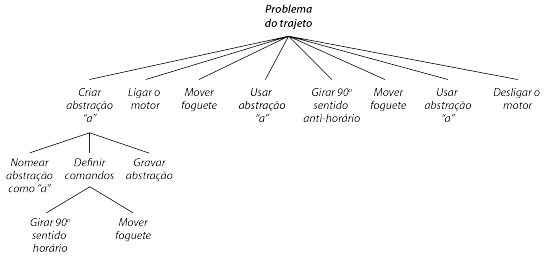


Figura 6.32 – Decomposição em árvore das demandas do do problema da carta 0001

Causalidade e Finalidade versus Representações e Práticas Sociais

A diferença mais importante entre os participantes observada ao longo do estudo de caso diz respeito à antecipação do reconhecimento de padrões recorrentes no trajeto da carta

0101 por P1. Considerando que a hipótese central desta tese é fundada na ideia de que diferentes RS sobre a Computação participariam de forma distinta dos processos de resolução de problemas de natureza computacional, a diferença observada na conduta dos participantes não apenas contribui para sustentar tal hipótese como também indica que a experiência com linguagens de programação pode desenvolver competências e habilidades mais globais de resolução de problemas, em sintonia com o argumento de Wing (2006) que defende a criação de oportunidades de aprendizagem do Pensamento Computacional para qualquer pessoa.

Retomando o episódio: O participante P1 utilizou a notação da linguagem *RocketSocket* na mesa para esboçar e pensar a solução para o trajeto na linguagem que ele mesmo havia criado e identificar os padrões recorrentes do trajeto. Se por um lado pode-se argumentar que P1 sentiu a necessidade de traduzir a solução de uma sintaxe mais familiar (*RocketSocket*) para uma menos (a recém-criada), por outro não se pode ignorar o fato de que a busca pelo padrão em si exerceu controle cognitivo igual ou superior ao da notação adotada pelo participante para desenvolver o programa. Em outras palavras, parte da finalidade definida pelo participante para o problema foi orientada por princípios mais globais e abstratos do seu funcionamento cognitivo – reconhecimento de padrões, uso de estruturas de controle do código – do que o objetivo da tarefa em si – coletar as estrelas desviando dos asteroides. Como consequência disso, toda as transformações causais no estado do tabuleiro por meio da combinação dos elementos primitivos da linguagem estavam subordinadas tanto às regras de operação da linguagem quanto às práticas mais gerais do participante enquanto programador.

Por outro lado, a inexperiência de P2 como programadora manteve as funções causal e teleonômica das unidades significativas observadas num nível mais contextualizado, quase sempre restrito ao sistema de regras da linguagem *RocketSocket* e às informações fornecidas pelo tutor-experimentador ao longo do processo. Colocando de outra forma, as possibilidades de ação percebidas pela participante foram igualmente orientadas por suas representações e práticas em relação à Computação, ainda que estas sejam incipientes.

Microgênese do Pensamento Computacional: Unidades Significativas

A Tabela 6.2 sintetiza as principais observações realizadas a partir das condutas prototípicas dos participantes, relacionando as unidades significativas procedimentais e representativas aos eventos ocorridos no processo de resolução do problema.

Tabela 6.2 – Eventos e Unidades Significativas

Evento	Unidade Procedimental	Unidade Representativa	
		Causalidade	**Finalidade**
Uso de Comandos	Ligar o motor do foguete	Marcação do início do programa	Definir o estado inicial da solução
	Desligar o motor do foguete	Marcação do fim do programa	Definir o estado final da solução
	Mover o foguete	Deslocamento	Coletar estrelas ou evitar colisão com asteroides
	Girar o foguete	Deslocamento	Coletar estrelas ou evitar colisão com asteroides
		Mudança de orientação	Coletar estrelas nas direções vertical e horizontal
	Usar um Laço de *n* repetições	Repetir um conjunto de comandos por *n* vezes	Repetir manobras do foguete no trajeto Simplificar a repetição de manobras Automatizar manobras que seguem padrões
	Gravar na Memória	Criar um novo bloco de comando que combina sequências de outros comando, meios de abstração da linguagem	Simplificar o código Criar soluções reutilizáveis
	Utilizar bloco de comando armazenado na memória	Invocar a execução de um bloco de comando criado pelo aprendiz	Simplificar o código Reutilizar soluções
Construção de sequências de comandos	Escrita do programa no plano Manipulação e encaixe dos blocos de comando (físicos)	Meios de combinação e abstração da linguagem, fluxo do programa, sucessão de eventos	Construção e execução do trajeto do foguete
Uso do Plano de Voo	Escrita do programa	Meios de combinação e abstração da linguagem, fluxo do programa, sucessão de eventos	Planejamento da solução do trajeto
	Desenho do foguete, estrelas e asteroides no plano	Representação dos resultados dos comandos do programa sobre o tabuleiro	Testar e avaliar a solução planejada
Execução do plano	Executar os blocos de comando (peças físicas) passo-a-passo	Marcar a sequência de execução dos comandos para transformar o tabuleiro	Execução dos comandos (problema local)
	Movimentar o foguete e coletar (remover) as estrelas	Realizar os estados do programa do plano sobre o tabuleiro	Execução do programa (problema global)

O grupo de eventos mais elementar observado corresponde aos usos dos comandos da linguagem, considerando seus respectivos significados isoladamente enquanto procedimentos: ligar ou desligar o motor, mover ou girar o foguete, criar laços, gravar abstrações na memória e utilizar as abstrações armazenadas. As funções causais dos eventos desse grupo estão subordinadas ao resultado esperado para cada comando: ligar o motor dá início ao programa, enquanto utilizar o bloco mover resultará no deslocamento do foguete.

As finalidade desses eventos, em contrapartida, referem-se ao plano mais geral da solução do trajeto, contextualizando o uso de cada comando a partir do que precisa ser feito para que todas as estrelas sejam coletadas e os asteroides evitados: desloca-se o foguete numa direção para que colete a estrela ou desvie de um asteroide; cria-se um laço para que determinado comando ou sequência de comandos se repita. Certos eventos não possuem relação direta com a tarefa, mas com a realização da solução: ligar e desligar o foguete marcam o início e o fim da solução elaborada; um laço também pode ter como finalidade repetir manobras ou simplificar sua execução em certos padrões de trajetos.

O terceiro grupo de eventos foi observado entre o primeiro e segundo grupo, uma vez que corresponde especificamente aos usos dos planos de voo na passagem do planejamento para a execução dos trajetos. Há dois procedimentos envolvidos: o primeiro consiste na escrita do programa na área específica, desenhando os comandos, enquanto o segundo corresponde ao desenho das respectivas posições do foguete, estrelas e asteroides. Em termos causais, a escrita do programa é regulada pelo mesmo sistema de regras que a construção de sequências de comandos do segundo grupo de eventos, porém com finalidade distinta. O plano de voo tem como objetivo planejar ou antever o trajeto a ser gerado pelo programa desenhado antes da sua execução nos blocos de comando, como num ensaio das transformações que serão realizadas no tabuleiro físico. Sendo assim, a representação dos estados do tabuleiro a partir simulação da execução do programa escrito no plano de voo tem o teste e avaliação daquela solução planejada como finalidade.

O último grupo é formado pelos eventos observados durante a execução do programa já construído com as peças de encaixe, etapa final da tarefa. O procedimento de execução é marcado inicialmente pela indicação, com gestos e leitura em voz alta, da parte do programa (comando isolado ou conjunto de comandos no *Laço*) que desencadeará mudanças no arranjo do tabuleiro naquele momento. Em seguida o participante move o foguete sobre o tabuleiro, coleta (remove) a estrela daquela casa e retorna às peças de encaixe para executar o passo seguinte do programa. A marcação do passo atual sobre o

programa escrito nas peças de encaixe tem finalidade local em relação ao problema, funcionando como um indicador de progressos rumo à solução. Já as transformações no tabuleiro estão subordinadas aos objetivos globais do problema, reduzindo a diferença entre o estado inicial e a conclusão do trajeto planejado a cada avanço na execução programa.

Limitações do estudo e direções futuras para o desenvolvimento da abordagem pedagógica

Os casos prototípicos apresentados foram uma primeira investigação acerca do potencial pedagógico da linguagem *RocketSocket* e precisam ser considerados dentro das limitações do desenho metodológico utilizado. Em primeiro lugar, os conceitos, práticas e perspectivas investigados no estudo de caso facilitaram a observação das microgêneses cognitivas ao mesmo tempo em que excluíram do estudo outros pontos fundamentais, tais como: eventos, paralelismo, manipulação de dados numéricos e literais, uso de operadores aritméticos e condicionais na linguagem. Essas questões merecem investigações específicas no futuro, principalmente por envolverem conhecimentos de Lógica e Álgebra combinados aos da Computação.

Quanto à estrutura da tarefa, a opção por fornecer objetivos mais restrições para o problema (Mosca, Silveira e Burigo, 1993) reduziu as variáveis a serem consideradas pelo participante na elaboração dos programas. Em outras situações, a posição inicial no tabuleiro não precisará ser sempre idêntica; a quantidade de peças necessárias para se chegar à solução pode ser inferior ao número disponibilizado pelo experimentador, obrigando o aprendiz a utilizar laços e abstrações; poderia haver tempos limites distintos para elaborar a solução, criando diferentes níveis de dificuldade para o processo. Cabe ressaltar que além da técnica de fornecer objetivos mais restrições, seria possível adotar o *jogo livre*, no qual o participante exploraria livremente a linguagem segundo seus próprios interesses; ou ainda fornecer *objetivos sem restrições*, deixando para o aprendiz o papel de compreender as questões sintáticas da linguagem enquanto busca a solução para a tarefa.

No que concerne a conduta do experimentador, as aplicações futuras da abordagem em disciplinas regulares de cursos superiores de Design provavelmente serão marcadas por intervenções menos frequentes e intensas por parte do professor e por mais autonomia do aprendiz sobre a tarefa. Espera-se que nessas situações a dinâmica da turma compense esse distanciamento, com os aprendizes ajudando-se mutuamente, trocando experiências e discutindo coletivamente as possibilidades de resolução.

Considerando o uso da linguagem *RocketSocket* por grupos maiores de aprendizes, após a conclusão do estudo de caso foi iniciado o desenvolvimento de uma versão digital da linguagem para uso em computadores e *tablets* (Fig. 6.33). Os prós dessa versão são a avaliação em tempo real das ações do usuário e execução animada da solução, substituindo em parte o papel do professor. Os contras, que requerem investigações mais cuidadosas, dizem respeito à inexistência da dimensão física da linguagem, uma vez que nem os blocos de comando nem o tabuleiro ou suas peças serão manipulados pelo aprendiz. A versão digital também dispensa o plano de voo, uma vez que a exploração do trajeto seria feita diretamente no tabuleiro digital, o que pode gerar mudanças nas condutas observadas ao longo deste estudo.

Figura 6.33 – Protótipo da versão digital da linguagem RocketSocket no *tablet* Apple iPad.

Finalmente, uma possibilidade percebida com o início do desenvolvimento da versão digital da linguagem *RocketSocket* refere-se à elaboração de situações-problema pelos próprios aprendizes ou professores que optarem por utilizar a abordagem. Para isso, bastaria oferecer um modelo em branco das cartas impressas ou fornecer um editor para os problemas na versão digital. Dificuldades identificadas pelos professores nos usos da linguagem com suas turmas, bem como temas e disciplinas específicas poderiam ser trabalhadas por meio de trajetos que abordem questões particulares.

O percurso da revisão de literatura à experimentação da linguagem foi orientado pelo esquema geral da tese apresentado no primeiro capítulo (Fig. 1.14, p.65). Recuperamos a última parte daquele esquema para nortear a integração teórica que se segue rumo ao encerramento deste trabalho.

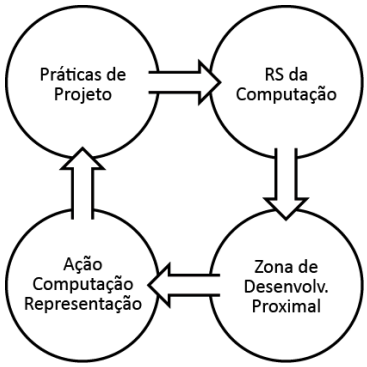


Figura 7.1 – Esquema geral da tese

7.1 – Práticas de Projeto

A linguagem *RocketSocket*, elaborada a partir da identificação das representações sociais (RS) dos estudantes de Design acerca dos Princípios da Computação, se propôs a oferecer oportunidades para que aqueles aprendizes possam se tornar *escritores* e não apenas *leitores* em meios computacionais. A distinção feita por Alan Kay (1989) entre as habilidades de acessar materiais e ferramentas produzidos por outros em um meio e as de criar materiais e ferramentas para outras pessoas pôde ser explicitamente observada nos relatos dos estudantes entrevistados. Apesar de utilizarem os computadores e os recursos da Computação cotidianamente, os estudantes de Design que participaram da pesquisa ainda são mais *usuários* que *projetistas* nesse contexto.

Com base nos dados dos estudos apresentados nos capítulos 4 e 6, a mudança na condição de *usuários-leitores* a *projetistas-escritores* dos estudantes em relação aos computadores dependeria menos do aumento do acesso aos equipamentos – alternativa criticada na apresentação desta tese – do que de uma oferta de oportunidades de aprendizado dos Princípios da Computação que contemple diferentes perfis de aprendizes. Os estudantes

entrevistados já desenvolvem trabalhos, comunicam-se e administram o andamento dos seus projetos acadêmicos e pessoais com o auxílio dos computadores, dos seus recursos e programas. Este dispositivo foi claramente descrito a partir do seu *potencial instrumental* experimentado pelos estudantes, com usos fortemente relacionados aos paradigmas enunciados por Moggridge (2007): meio de expressão, veículo e ferramenta.

Por outro lado, o *potencial representacional* da Computação ainda não ocupa o mesmo local de destaque. Apenas entre os estudantes que se definiram como programadores foram observadas questões associadas às condutas da *escrita* nos meios computacionais, com relatos acerca do aumento da autonomia e independência, desenvolvimento do pensamento lógico e melhor capacidade de comunicação com outros profissionais da área da informática. Não obstante, entre estes mesmos estudantes, a programação não figurou entre as três principais atividades realizadas mais frequentemente nos computadores, dado que foi o primeiro indício do distanciamento entre as RS que os participantes compartilham sobre os computadores e a prática da programação.

O segundo indício do referido distanciamento entre usar o dispositivo e programá-lo foi identificado nas respostas dos participantes quanto ao auxílio dos computadores e da Computação na resolução de problemas cotidianos de Design acadêmicos, profissionais ou pessoais. Tais relatos sustentaram a confirmação do caráter das prescrições nos termos da centralidade incondicional do computador pelos estudantes e do posicionamento periférico de *programação* na estrutura da representação. Programar ainda é uma prática com potencial mais *ferramental* que *representacional*, assim como utilizar programas gráficos ou acessar a Internet, embora contribua para materializar as RS da Computação no cotidiano dos estudantes que as compartilham.

Conforme citado no primeiro capítulo desta tese, Wing (2006) argumentou que o estudo da Computação teria como características: a) saber mais do que programar, pensando em diferentes níveis de abstração; b) desenvolver habilidades fundamentais ao invés de mecânicas; c) entender a forma como os seres humanos e não as máquinas pensam; d) combinar e complementar os pensamentos matemático, científico e da engenharia; e) abordar ideias e conceitos para que as pessoas resolvam problemas, e não simplesmente concentrar-se nos usos dos dispositivos (*hardware*) e programas (*software*). Nesse sentido e considerando os dados desta pesquisa, é de suma importância que 1) abordagens pedagógicas como a linguagem *RocketSocket* desvinculem gradualmente a Computação do computador enquanto objeto que lhe concede materialidade, oferecendo alternativas que desenvolvam a percepção dos aprendizes quanto à amplitude e diversidade de cenários

possíveis para o emprego dos Princípios da Computação; e 2) incentivem o entendimento de que a programação, mais do que um uso ferramental da máquina, consiste numa oportunidade para aprender, discutir e explorar aqueles Princípios, reconhecendo sua pertinência em diversas áreas e situações de resolução de problemas, com ou sem a presença de um computador como ferramenta.

Os estudos de caso prototípicos descritos no capítulo 6 apontaram caminhos do potencial da linguagem na referida ampliação da gama de objetos que dão concretude à Computação e da criação de oportunidades para que o aprendiz perceba a pertinência dos Princípios para suas competências e habilidades mais gerais de resolução de problemas.

7.2 – Representações Sociais da Computação e Zona de Desenvolvimento Proximal

As RS dos Princípios da Computação identificadas na pesquisa, sintetizadas na Tabela 4.34[64], foram interpretadas como o nível de desenvolvimento real dos estudantes de Design entrevistados acerca daqueles conhecimentos. Certamente não é possível assumir que qualquer estudante dos cursos superiores de Design compartilhe aquelas representações ou esteja no mesmo nível de desenvolvimento. Contudo, as várias instituições de ensino superior que ofertam graduações em Design possuem em comum diretrizes curriculares nacionais que, em maior ou menor grau, conectam os projetos pedagógicos e consequentemente a formação dos estudantes da área em todo o país.

Partindo desse princípio, as RS identificadas, sem desconsiderar as limitações descritas no final do capítulo 5, podem servir de parâmetro para entender a relação de um grupo de indivíduos que, se não experimentam necessariamente o mesmo percurso acadêmico ou as mesmas ofertas profissionais no mercado de trabalho da sua respectiva região, possuem em comum uma mesma profissão com um conjunto de práticas específicas, fundada em teorias, métodos e referências compartilhadas. Deste modo, a articulação teórica apresentada na introdução desta tese foi fundamental para realizar a transformação das conclusões do estudo das RS na abordagem materializada pela linguagem *RocketSocket*, conforme detalhado a seguir.

Construcionismo

A adoção da perspectiva Construcionista (Turkle e Papert, 1991), foi fundamental para que a linguagem *RocketSocket* contemplasse, desde a sua concepção, diferentes perfis de aprendizes em diferentes níveis de desenvolvimento da sua relação com os Princípios da

[64] Ver p.115.

Computação. É imprescindível destacar que a efetiva flexibilidade da linguagem frente à diversidade de perfis de estudantes de Design do ensino superior ainda precisa ser investigada e o passo seguinte à conclusão desta tese consistirá em prosseguir no desenvolvimento e aperfeiçoamento gradual da abordagem.

Os estudos de caso prototípicos demonstraram que pelo menos os perfis mais gerais encontrados entre os participantes da pesquisa, *programadores* e *não-programadores*, experimentaram positivamente as possibilidades da linguagem. A Zona de Desenvolvimento Proximal (ZDP) construída pelos participantes P1 e P2 na interação com os elementos da linguagem, seus meios de combinação e abstração, sob a orientação do tutor, promoveu algum grau de aprendizado qualitativo em direção um novo nível de relacionamento com a Computação e seus Princípios. A recomendação de Papert (1980) sobre a importância de se utilizar linguagens apropriadas para a comunicação com o aprendiz ao longo do processo revelou-se extremamente produtiva, especialmente quanto às peculiaridades da área da Computação. A escolha da temática do foguete coletor de estrelas criou um contexto familiar, com operações e possibilidades de ação concretas para os participantes do estudo de caso. De toda forma, a compreensão acerca do que efetivamente foi aprendido e por quais razões, bem como os ganhos cognitivos para as habilidades e competências mais gerais de resolução de problemas também são temas para pesquisas e aperfeiçoamentos futuros da linguagem.

Contextualização dos processos representacionais

Abric (2001, p.200) afirmou que o comportamento de um grupo no processo de resolução de problemas seria mais determinado pela representação que faz da tarefa do que pelo tipo de tarefa em si. De forma similar, Jodelet (1985) explicou que a RS elaborada pelo grupo define os objetivos e procedimentos de solução para seus integrantes, nem sempre levando em conta a estrutura da tarefa. Essa ênfase nas contribuições dos conhecimentos e saberes compartilhados pelo grupo na capacidade de pensar e compreender as ações possíveis para solucionar o problema foi o ponto de partida para articular a Teoria das Representações Sociais de Moscovici, a Psicologia Sócio-Histórica de Vygotsky e a Epistemologia Genética de Piaget.

Em primeiro lugar, verificar as RS dos estudantes sobre os Princípios da Computação forneceu os contornos gerais do conhecimento partilhado por eles sobre aquela disciplina. A opção por investigar o fenômeno com o suporte da Teoria das Representações Sociais auxiliou na compreensão dos elementos das representações dos estudantes no que tange às suas funções (Abric, 2001): como se comunicam, compartilham e difundem os saberes

sobre a Computação; como constroem suas identidades de estudantes e profissionais de Design na interação com um saber que os diferencia de programadores e outros profissionais da informática no mercado de trabalho, com implicações sobre a autonomia para realizar projetos; nas prescrições daqueles saberes quanto às suas finalidades e pertinências na resolução de tarefas de projeto; como justificam sua relação com a Computação, considerando tanto outros estudantes designers que diferenciam-se por serem programadores ou não, quanto colegas de trabalho, amigos e familiares que possuem experiências distintas.

A abordagem estrutural da Teoria (Abric, 1993) permitiu compreender as funções e relações desses elementos em termos prescritivos, bem como o processo de geração mútuo de representações e práticas: por um lado, a forma pela qual os estudantes representam os Princípios da Computação manifestou-se nos usos do *potencial instrumental* dos computadores; por outro novas práticas como o uso da linguagem *RocketSocket* poderiam gradualmente promover transformações no sistema periférico da representação e, no longo prazo, talvez modificar a própria representação dos Princípios em direção ao *potencial representacional* da Computação.

Essa modificação no núcleo da representação, embora não investigada formalmente nesta tese, parece plausível com base na comparação entre o desempenho de P1 e P2 no estudo de caso do capítulo 6: Programador e não-programador exibiram condutas claramente distintas na escolha das ações possíveis em eventos do processo de resolução dos trajetos do foguete, principalmente no uso criativo dos recursos da linguagem e na definição da finalidade da tarefa. A conclusão a que se chega nesta tese é que essa diferença pode ser explicada por diferentes RS e pelos efeitos que elas geraram nas práticas e funcionamento cognitivo dos participantes. Em outras palavras, concordando visão de Abric (1993, 2001) sobre o desenrolar das transformações das representações pelas práticas, a experiência contínua dos estudantes de Design com a linguagem *RocketSocket* poderia dar início a um processo de transformação das RS deles sobre os Princípios da Computação, que por sua vez poderiam ser observadas pelo aparecimento de novas práticas e usos da linguagem ou mesmo em outras situações de projeto.

Em segundo lugar, a articulação proposta entre a teoria de Vygotsky (1997) ampliada por Werstch (1985) e as Teoria das Representações Sociais facilitou a concepção da linguagem *RocketSocket* como objeto mediador para a construção de uma ZDP na qual operaram mecanismos de objetivação e ancoragem (Moscovici, 2003): 1) os conceitos, práticas e perspectivas do Pensamento Computacional (Brennan, Chung e Hawson, 2011),

potencialmente não-familiares para os estudantes não-programadores foram ancorados na narrativa espacial da coleta de estrelas e asteroides, onde o foguete objetivou um *dispositivo que computa* mas que não é exatamente um computador, com instruções conhecidas a serem programadas – voar, girar, ligar e desligar o motor; 2) aprender a construir os trajetos é a tarefa que ocorre no interior dessa ZDP; 3) a mediação do processo se dá pelos recursos da linguagem que dialoga com as representações do aprendiz sobre a narrativa enquanto introduz os conceitos não-familiares (algoritmos, comandos, sequências, laços, abstrações e automações) de forma familiar (trajetos, controle do foguete, repetições, memória do foguete).

Em terceiro lugar, no desenrolar do processo construção dos trajetos no interior da ZDP, as contribuições da Epistemologia Genética sobre a relação entre os sujeitos psicológico e epistemológico (Inhelder e Caprona, 1997) completou a triangulação das teorias para explicar a passagem de um saber compartilhado, socialmente construído, às ações e condutas individualizadas de um solucionador que atua sobre objetos cujos significados são construídos na interação do aprendiz com o contexto da tarefa. Uma vez identificadas as RS compartilhadas pelo grupo de estudantes de Design e definidas a narrativa e estrutura do contexto de aprendizagem (a situação-problema do foguete coletor de estrelas que desvia dos asteroides, o tabuleiro, os elementos da linguagem *RocketSocket*, seus meios de combinação e abstração), restou investigar como todo esse cenário seria apropriado e colocado em ação por um estudante. Esta última parte da tese analisou, portanto, as microgêneses cognitivas das condutas de um solucionador que seleciona e avalia a pertinência de um determinado saber nas transformações dos estados de um problema rumo à solução.

7.3 – Ação, Computação, Representação

A triangulação teórica RS-ZDP-Microgêneses e o estudo descrito no capítulo 4 sustentaram elaboração da abordagem pedagógica que culminou na linguagem *RocketSocket,* seguindo um percurso que partiu da identificação da estrutura (das RS) em direção à função dos conhecimentos sobre a Computação no processo de resolução de problemas. O estudo de caso apresentado no capítulo 6 pode ser entendido como um microcosmo do mesmo percurso, com a diferença de que ao invés de verificar como os estudantes se relacionaram com os Princípios da Computação e os computadores até então, verificou-se o relacionamento dos participantes com alguns daqueles Princípios num tempo, espaço e dispositivo computacional bem definidos.

Figura 7.2 – Esquema final da tese: ação, computação, representação

Em concordância com Saada-Robert (1997, p.159), foi observada [65] (Fig. 7.2) uma passagem dos conhecimentos gerais e difusos do solucionador (relacionados às representações sobre *comandos*, *programas*, *linguagens de programação* e sobre os próprios Princípios da Computação, além daquilo que se sabe sobre foguetes viajando pelo espaço) para esquemas de ação precisos, selecionados e avaliados em função das possibilidades percebidas para a transformação dos estados de problemas específicos – aquele foguete particular que se move uma casa por vez sobre o tabuleiro na direção para a qual seu bico aponta, podendo girar 90º nos sentidos horário e anti-horário, utilizando automações e abstrações enquanto coleta estrelas e desvia de asteroides num trajeto.

Por fim, conclui-se que a opção por construir uma abordagem pedagógica centrada na *ação* de um solucionador que *computa* com base em *representações* sobre os Princípios da Computação demonstrou-se promissora e potencialmente relevante para os estudantes de Design. Uma série de pontos que requerem aprofundamentos e novas investigações foram identificados, de maneira que os produtos desta tese são apenas o primeiro passo em direção à oferta efetiva de oportunidades de aprendizado universal dos Princípios da Computação e do consequente desenvolvimento do Pensamento Computacional.

[65] Ver as síntese das páginas 80, 113, 122 e 158 para um detalhamento dos dados que compõem a Fig. 7.2.

Abelson, H., e Sussman, G. J. (1996). *Structure and Interpretation of Computer Programs*, 2nd ed. Cambridge: MIT Press.

Abric, J. (1993). Central System, Peripheral System: their functions and roles in the dynamics of Social Representations. *Papers on Social Representations*, 2(2), 75-78.

Abric, J. (2001). Prácticas Sociales, Representaciones Sociales. In: Abric, J. (Org). *Prácticas sociales y representaciones*. D. F. México: Ediciones Coyoacán.

Alexander, C. (1964). *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.

Amorim, C. (2005). Beyond Algorithmic Thinking: An Old New Challenge for Science Education. *International History, Philosophy, Sociology & Science Teaching Conference*. Disponível em http://www.ihpst2005.leeds.ac.uk/papers.htm. Acesso em 16 de setembro de 2013.

Araque, J. Q., Maiden, R. P., Bravo, N., Estrada, I., Evans, R., Hubchik, K., Kirby, K., Reddy, M. (2013). Computer usage and access in low-income urban communities. *Computers in Human Behavior*, 29(4), 1393-1401.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48- 54.

Bardin, L. (2006). *Análise de conteúdo*. Lisboa: Edições 70.

Barrella, F.M.F. (1991). 1um, 2dois, 3tres: Buscando Significados Através do Logo. In: Valente, J.A. (org.) *Liberando a Mente: Computadores na Educação Especial*. Campinas, Gráfica Central da UNICAMP, 1991.

Bense, M. (1968). *Pequena Estética*. São Paulo: Perspectiva.

Blanchet, A. (1997). As unidades procedimentais, causais e teleonômicas no estudo dos processos cognitivos. In: Inhelder, B. e Cellérier, G. (Orgs). *O percurso das descobertas das crianças*. Lisboa: Instituto Piaget.

Branco, A. U. e Salomão, S. J. (2001). Cooperação, competição e individualismo: pesquisa e contemporaneidade. *Temas em Psicologia da SBP*, 9(1), 11-18.

Brennan, K., Chung, M. e Hawson. J. (2011). *Creative Computing: a design-based introduction to Computational Thinking*. Disponível em http://tinyurl.com/brennan-ct. Acesso em 25 de setembro de 2013.

Bruner, J. (1964). The course of cognitive growth. *American Psychologist*, 19(1), 1-15.

Bruner, J. (1976). *Uma nova teoria de aprendizagem*. Rio de Janeiro: Edições Bloch.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2), 67-69.

Canter, M. (1986). The New Workstation: CD ROM Authoring Systems. In: In: Packer, R. & Jordan, K. (2001) *Multimedia: from Wagner to Virtual Reality*. New York: W.W. Norton & Company.

Carbonaro, M., Szafron, D., Cutumisu, M. e Schaeffer. J. (2010). Computer-game construction: A gender-neutral attractor to Computing Science. *Computuers & Educucation,* 55(3), 1098-1111.

Cellérier, G. (1997). Organização e funcionamento dos esquemas. In: Inhelder, B. e Cellérier, G. (Orgs). *O percurso das descobertas das crianças*. Lisboa: Instituto Piaget.

Ceruzzi, P. (2003). *A History of Modern Computing, Second Edition*. Cambridge: MIT Press.

Chaib, M. (2002). Frankenstein na sala de aula: as representações sociais docentes sobre informática. *Nuances*, 8(8), 47-64.

Chi, M. T. H., e Glaser, R. (1985). Problem solving ability. In: R. Sternberg (Ed.), *Human Abilities: An Information-Processing Approach*. San Francisco: W. H. Freeman & Co.

Contarello, A., & Sarrica, M. (2007). ICTs, social thinking and subjective well-being – The internet and its representations in everyday life. *Computers in Human Behavior*, 23(2), 1016–1032. http://doi.org/10.1016/j.chb.2005.08.013

Dale, N. e Lewis, J. (2011). *Ciência da Computação*, 4a edição. Rio de Janeiro: LTC.

De Smedt, T., Lechat, L. e Daelemans, W. (2011). Generative Art Inspired by Nature, Using NodeBox. In: *Lecture Notes in Computer Science*, v.6625, 264-272.

Denning, P. J. (2003). Great principles of computing. *Communications of the ACM*, 46(11), November 2003, pp.15-20.

Denning, P. J. (2004). Great principles in computing curricula, *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, p.336-341, March 03-07, 2004, Norfolk, Virginia, USA.

Denning, P. J. (2007). Computing is a natural science. *Communications of the ACM*, 50(7), July 2007, pp.13-18.

Denning, P. J., e Martell, C. (2007a). *Criteria for Principle Statements*. Disponível em http://cs.gmu.edu/cne/pjd/GP/gp_criteria.html. Acesso em 20 setembro de 2013.

Denning, P. J., e Martell, C. (2007b). *Summary of Great Principles*. Disponível em http://cs.gmu.edu/cne/pjd/GP/gp_summary.html. Acesso em 20 de setembro de 2013.

Denning, Peter J., Comer, Douglas E., Gries, D., Mulder, Michael C., Tucker, A., Turner, J., & Young, Paul R. (1988). Computing as a discipline: preliminary report of the ACM task force on the core of computer science. *SIGCSE Bulletin*, 20 (1), February 1988, 41-41.

Diverio, T., e Menezes, P. (2011). *Teoria da Computação: máquinas universais e computabilidade*. Porto Alegre: Bookman.

Doise, W. (1992). L'ancrage dans les études sur les représentations sociales. *Bulletin de psychologie, Tome 45 (4–7),* 405, p. 189-195.

Doise, W. (2002). Da Psicologia Social à Psicologia Societal. *Psicologia: Teoria e Pesquisa*, 18(1), pp. 27-35.

Doise, W. e Mugny, G. (1997). *Psicologia Social e Desenvolvimento Cognitivo*. Lisboa: Instituto Piaget.

Duveen, G. (1994). Crianças enquanto atores sociais: as representações sociais em desenvolvimento. In: Guareschi, P. e Jovchelovitch, S. (orgs). *Textos em Representações Sociais*. Petrópolis: Editora Vozes.

Eichler, M. e Fagundes, L. C. (2001). A Microgênese da Explicação de um Problema Ambiental: Os casos Paulo e Piter. *Psicologia: Reflexão e Crítica*, 14(3), 505-520.

Engeström, Y. (2009). Aprendizagem expansiva: por uma reconceituação pela teoria da atividade. In: Illeris, K. (Org). *Teorias contemporâneas da aprendizagem*. Porto Alegre: Grupo A / Penso Editora.

Eysenck, M. W., Keane, M. T. (2007). *Manual de Psicologia Cognitiva*. Porto Alegre: Artmed.

Ferruzzi, E. C. (2001). *Considerações sobre a Linguagem de Programação Logo*. Seminário Apresentado no GEIAAM – Grupo de Estudos de Inteligência Artificial Aplicada à Matemática - UFSC - Setembro/2001.

Flament, C. (2001). Estructura, dinâmica y transformación de las Representaciones Sociales. In: Abric, J. (Org). *Prácticas sociales y representaciones*. D. F. México: Ediciones Coyoacán.

Flavell, J. H.; Cooper, A. e Louiselle, R. H. (1958). Effect of the number of pre-utilization functions on functional fixedness in problem solving. *Psychological Reports*, 4, 343-350.

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. *Lecture Notes in Computer Science*, 4226, 159-168.

Galanter, P. (2003). What is Generative Art? Complexity theory as a context for art theory. in *International Conference on Generative Art*. 2003. Milan: Generative Design Lab.

Garcia, R. (1996). Dialética, psicogênese e história das ciências. In: Piaget, J. *As formas Elementares da Dialética*. São Paulo: Casa do Psicólogo.

Gardner, M. (1970). Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223, 120-123.

Greeno, J. (1994). Gibson's Affordances. *Psychological Review*, 101(2), 336-342.

Guterres, J. O., Eichler, M. L. e Del Pino, J. C. (2007). Análise de um caso exemplar da microgênese da identificação e da classificação de minerais. In: *VI ENPEC - Encontro Nacional de Pesquisa em Educação em Ciências*, 2007, Florianópolis. Anais do VI ENPEC, 2007.

Hakkarainen, P. (2012). 'No good for shovelling snow and carrying firewood': Social representation of computers and the internet by elderly Finnish non-users. *New Media & Society*, 14(7), 1198-1215.

Harel, I. e Papert. S. (1991). *Constructionism*. New York: Ablex Publishing.

Harvey, B. (1997). *Computer Science Logo Style volume 1*. Cambridge: MIT Press.

Hernandez, C. R. B. (2006). Thinking parametric design: introducing parametric Gaudi. *Design Studies*, 7(3), 309-324.

Inhelder, B. e Caprona, D. (1997). Em direção ao construtivismo psicológico: Estruturas? Procedimentos? Os dois são indissociáveis. In: Inhelder, B. e Cellérier, G. (Orgs). *O percurso das descobertas das crianças*. Lisboa: Instituto Piaget.

Inhelder, B., e Piaget, J. (1979). Procedures et structures. *Archives de Psychologie*, 47, 165-176. Tradução de João Alberto da Silva.

IRCAM (2009). *A Brief Story of MAX*. Disponível em http://tinyurl.com/max-ircam. Acesso em 18 de setembro de 2013.

Jodelet, D. (1985) La Representación Social: fenómenos concepto y teoría. In Moscovici, S. (Org). *Psicología social II*. Barcelona: Paidos.

Jones, J. C. (1992). *Design Methods*. New York: John Wiley and Sons.

Kay, A. (1972). A Personal Computer for Children of All Ages. In *Proceedings of the ACM annual conference* - Volume 1 (ACM '72), Vol. 1. ACM, New York, NY, USA.

Kay, A. (1987). *Doing with images makes symbols*. University Video Communications. Disponível em http://archive.org/details/AlanKeyD1987. Acesso em 20 de setembro de 2013.

Kay, A. (1989). User Interface: A personal view. In: Packer, R. & Jordan, K. (2001) *Multimedia: from Wagner to Virtual Reality*. New York: W.W. Norton & Company.

Kay, A. e Goldberg, A. (1977) Personal Dynamic Media. *VPRI Memo* M-1977-001.

Kay, R. (2007). Gender differences in computer attitudes, ability, and use in the elementary classroom. *The Literacy and Numeracy Secretariat*, (8), 1-4.

Kelman, C. A. e Branco, A. U. (2004). Análise microgenética em pesquisa com alunos surdos. *Revista Brasileira de Educação Especial*, 10(1), 93-106.

Knuth, D. (1997). *The Art of Computer Programming*, Third Edition, Volume 1. Massachusetts: Addison-Wesley.

Oliveira, M. K. (2010). *Vygotsky: aprendizado e desenvolvimento um processo sócio histórico*. São Paulo: Editora Scipione.

Langley, P., & Rogers, S. (2005). An extended theory of human problem solving. *Proceedings of the Twenty-Seventh Annual Meeting of the Cognitive Science Society*. Stresa, Italy.

La Taille, Y. (1992). O Lugar da Interação Social na Concepção de Jean Piaget. In: La Taille, Y., Oliveira, M. K. e Dantas, H. (Orgs). *Piaget, Vygotsky, Wallon: Teorias Psicogenéticas em Discussão*. São Paulo: Summus Editorial.

Leman, P. J. (1998). Social relations, social influence and the development of knowledge. *Papers on Social Representations*, 7(1-2), 41-46.

Leontyev, A. (2009). *Activity and Counciousness*. Marxists Internet Archive. Disponível em http://www.marxists.org/archive/leontev/works/activity-consciousness.pdf. Acesso em 10 de outubro de 2013.

Lieser, W. (2010). *Arte Digital: Novos caminhos na arte*. Potsdam: H.F. Ullman.

Lourenço, O. (2002). *Psicologia de Desenvolvimento Cognitivo: Teoria, Dados e Implicações 2ª ed*. Coimbra: Almedina.

Luria, A. R. (2010). *Desenvolvimento Cognitivo*. São Paulo: Ícone Editora.

Maeda, J. (2001). *Design by Numbers*. Cambridge: MIT Press.

Maeda, J. (2004). *Creative Code*. London: Thames and Hudson.

Manovich, L. (2013). *Software Takes Command*. New York: Bloomsbury Academic.

Matlin, M. W. (2004). *Psicologia Cognitiva*, 5ª ed. São Paulo: LTC.

Meggs, P. (1997). *A History of Graphic Design*. New York: Wiley.

Meira, L. (1994). Análise microgenética e videografia: ferramentas de pesquisa em psicologia cognitive. *Temas em Psicologia*, 2(3), 59-71.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review,* 63 (2), 81–97.

Ministério da Educação e Cultura (2004). RESOLUÇÃO Nº 5, DE 8 DE MARÇO DE 2004. Disponível em http://portal.mec.gov.br/cne/arquivos/pdf/rces05_04.pdf. Acesso em 15out. 2014.

Mitchell, W. J. (1990). A New Agenda for Computer-Aided Architectural Design. In McCullough, M., Mitchell, W. J, e Purcell, P. (orgs). *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*. Cambridge: MIT Press.

Moggridge, B. (2007). *Designing Interactions*. Cambridge: MIT Press.

Montangero, J. e Maurice-Naville, D. (1994). *Piaget ou a Inteligência em Evolução*. Porto Alegre: Artes Médicas.

Moscovici, S. (2003). *Representações Sociais: investigações em psicologia social*. Petrópolis: Editora Vozes.

Moro, M. L. F. (2000). A epistemologia genética e a interação social de crianças. *Psicologia: Reflexão e Crítica*, 13(2), 295-310.

Mosca, P. R. F., Silveira, J. F. P., e Burigo, E. (1993). Processos cognitivos infantis na resolução de problemas no campo da matemática: o caso da interação com programas-semente. *Psicologia: Reflexão e Crítica*, 6(1/2), 57-83.

National Research Council (1999). *Fluent With Information Technology by the National Research Council*. Washington: National Academy Press.

Newell, A. (1978). Information-Processing Theory of Human Problem Solving. In W.K. Estes (ed.) *Handbook of Learning and Cognitive Processes*, vol. V. NJ: Lawrence Erlbaum Associates.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a Theory of Human Problem Solving. *Psychological Review*, 65(3), 151-166.

Newell, A., & Simon, H. A. (1976). Computer Science as Empirical Inquiry: Symbols and Search. *Communications of the ACM*, 19(3), 113–126.

Oliveira, M. K. (1992). Vygotsky e o Processo de Formaço de Conceitos. . In: La Taille, Y., Oliveira, M. K. e Dantas, H. (Orgs). *Piaget, Vygotsky, Wallon: Teorias Psicogenéticas em Discussão*. São Paulo: Summus Editorial.

Palmonari, A. e Cerrato, J. (2011). Representações Sociais e psicologia social. In: Almeida, A. M. O., Santos, M. F. S e Trindade, Z. A. (Orgs). *Teoria das Representações Sociais 50 anos*. Brasília: TechnoPolitik.

Papert, S. (1971). A Computer Laboratory for Elementary Schools. *MIT AI Lab. LOGO Memo* 1, October 1971.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1994). *A Máquina das Crianças*. Porto Alegre: Artes Médicas.

Papert, S. e Solomon, C. J. (1971). Twenty things to do with a computer. *MIT AI Lab. LOGO Memo* 3, July 1971.

Pasqualotti, A., Barone, D. A. C., e Doll, J. (2012). Communication, technology and ageing: elderly, senior citizen groups and interaction process in the information age. *Saúde e Sociedade*, 21(2), 435-445.

Pellegrino, G. (2003). Representations and Uses of the Intranet: A comparative Case Study. *Bulletin of Science, Technology and Society*, 23(4), 281-296.

Piaget, J. (1964). Development and learning. In: Gauvain, M. e Cole, M. (orgs). *Readings on the development of Children*. New York: W. H. Freeman and Company.

Piaget, J. (1973). *Estudos Sociológicos*. Rio de Janeiro: Forense Universitária.

Piaget, J. (1987). *O Nascimento da Inteligência na Criança*. Rio de Janeiro: Editora Guanabara

Piaget, J. (2003). *Seis estudos de Psicologia*. São Paulo: Forense Universitária.

Piaget, J. e Inhelder, B. (2003). *A Psicologia da Criança*. São Paulo: Difel.

Polli, G. e Wachelke, J. (2013). Confirmação de Centralidade das Representações Sociais pela Análise Gráfica do Questionário de Caracterização. *Temas em Psicologia*, 21(1), 97-104.

Rajaraman, A. e Ullman, J. D. (2011). Data Mining. In: *Mining of Massive Datasets,* 1st ed. Cambridge: Cambridge University Press.

Reas, C. e Fry. B. (2007). *Processing: A Programming Handbook for Visual Designers and Artists*. Cambridge: MIT Press.

Resnick, M. & Ocko, S. (1991). LEGO/Logo: Learning Through and About Design in Harel, I. e Papert, S. (orgs). *Constructionism*. New Jersey: Ablex Publishing.

Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a General Theory of Planning. *Policy Sciences*, 4, 155–169.

Rocha, H. V. (1995). Ambientes de Programação com fins Educacionais: A proposta SLogo in Valente, J. (org). *O Professor no Ambiente Logo: Formação e Educação*. Campinas: NIED.

Rusk, N., Resnick, M., e Cooke, S. (2009). Origins and Guiding Principles of the Computer Clubhouse. In Kafai, Y., Peppler, K., and Chapman, R. (eds.), *The Computer Clubhouse: Constructionism and Creativity in Youth Communities*. Teachers College Press.

Rutkowska, J. (1990). Action, connectionism and enaction: A developmental perspective. *AI & Society*, 4(2), 96-114.

Sá, C. P. (1996). *Núcleo Central das Representações Sociais*. Rio de Janeiro: Vozes.

Saada-Robert, M. (1997). A construção microgenética de um esquema elementar. In: Inhelder, B. e Cellérier, G. (Orgs). *O percurso das descobertas das crianças*. Lisboa: Instituto Piaget.

Sant'Anna, H.C., Gatti, F., Carmo, J. C., Rocha, M. A., Rangel, S. O. e Neves, V. B. (2012a). Da Arte Generativa ao Pensamento Computacional - Uma análise comparativa das plataformas de aprendizagem. In A*nais do 11o Encontro Internacional de Arte e Tecnologia*. Brasília: Departamento de Artes Visuais/UnB.

Sant'Anna, H. C. (2012b). openEvoc: um programa de apoio à pesquisa em Representações Sociais. In: Avellar, L. Z., Ciscon-Evangelista, M. R., Nardi, M. B., Nascimento, A. S.,

Ribeiro Neto, P. M. (Orgs.). *Psicologia Social: desafios contemporâneos*. Vitória: GM Gráfica e Editora.

Segaran, T. (2008). *Programando a inteligência coletiva*. Rio de Janeiro: Alta Books.

Shapiro, L. (2011). *Embodied Cognition*. New York: Routledge.

Siegler, R. S. e Crowley, K. (1991). The Microgenetic Method: A Direct Means for Studying Cognitive Development. *American Psychologist*, 46(6), 606-620.

Singéry, J. (2001). Representaciones sociales y proyecto de cambio tecnológico en empresa. In: Abric, J. (Org). *Prácticas sociales y representaciones*. D. F. México: Ediciones Coyoacán.

Sociedade Brasileira de Computação – SBC (2011). *Computação Brasil*, abril/maio/junho de 2011. Porto Alegre: Giornale Comunicação Empresarial.

Spink, M. J. (2003). Desvendando as teorias implícitas: uma metodologia de análise das Representações Sociais. In: Guareschi, P., Jovchelovitch, S. (orgs*). Textos em Representações Sociais*. Petrópolis: Vozes.

Sternberg, R. J. (2010). *Psicologia Cognitiva*. São Paulo: Cengage Learning.

Strohecker, C. (1996). Understanding topological relationships through comparisons of similar knots. *AI&Society: Learning with Artifacts* 10:1, 58-69.

Thomson, E. e Stapleton, M. (2009). Making Sense of Sense-Making: Reflections on Enactive and Extended Mind Theories. *Topoi*, 28(1), 23-30.

Trindade, Z. A., Santos, M. F, S e Almeida, A. M. (2011). Ancoragem: notas sobre consensos e dissensos. In: Almeida, A. M. O., Santos, M. F. S e Trindade, Z. A. (Orgs). *Teoria das Representações Sociais 50 anos*. Brasília: TechnoPolitik.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, v. 59, n. 236.

Vala, J. (1997). Representações sociais e percepções intergrupais. *Análise Social*, 32 (140), 7-29.

Vala, J. e Monteiro, M. B. (2004). *Psicologia Social 6ª ed*. Lisboa: Fundação Calouste Gulbenkian.

Varela, F. (1994). *Conhecer: as Ciências Cognitivas, tendências e perspectivas*. Lisboa: Instituto Piaget.

Varela, F., Thompson, E. e Rosch, E. (2002). *The Embodied Mind*. Cambridge: MIT Press.

Vèrges, P. (2002). *Manuel Evoc2000 – Ensemble de Programmes Permettant L'analyse des Evocations*. Disponível em http://tinyurl.com/manualevoc. Acesso em 04 de setembro de 2011.

Venturelli, S. e Burgos, F. (1999). *Arte Computacional*. Texto apresentado no evento DigitalArte, ICBA, Salvador, Bahia – 19 de março de 1999. Disponível em http://tinyurl.com/venturelli1. Acesso em 11 de setembro de 2013.

Vygotsky, L. (2007). *A formação social da mente*. São Paulo: Martins Fontes.

Vygotsky, L. (2008). *Pensamento e Linguagem*. São Paulo: Martins Fontes.

Wachelke, J. (2009). Índice de Centralidade de Representações Sociais a partir de Evocações (INCEV): Exemplo de Aplicação no Estudo da Representação Social sobre Envelhecimento. *Psicologia: Reflexão e Crítica*, 22(1), 102-110.

Wachelke, J. (2012). Representations and social knowledge: An integrative effort through a normative structural perspective. *New Ideas in Psychology*, 30(2012), 259-269.

Wachelke, J. (2012). Social Representations: a Review of Theory and Research from the Structural Approach. *Universitas Psychologica*, 11(3), 729-741.

Wachelke, J. e Wolter, R. (2011). Critérios de Construção e Relato da Análise Prototípica para Representações Sociais. Psicologia: Teoria e Pesquisa, 27(4), 521-526.

Wagner, W., Duveen, G., Farr, R., Jovchelovitch, S., Lorenzi-Cioldi, F., Marková, I., & Rose, D. (1999). Theory and method of social representations. *Asian Journal of Social Psychology*, 2, 95-125.

Wertsch, J. (1985). *Vygotsky and the Social Formation of Mind*. Cambrige: Harvard University Press.

Wing, J. M. (2006) Computational Thinking. *Communications of the ACM*, 49(3), March 2006.

Wing, J. M. (2008). Computational Thinking and Thinking About Computing, *Philosophical Transactions of the Royal Society*, 366, 3717-3725.

Wing, J. M. (2011). *Research Notebook: Computational Thinking – What and Why?* Disponível em http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why. Acesso em 11 de setembro de 2013.

Winograd, T. (1996). *Bringing Design to Software. Boston*: Addison-Wesley.

Yin, R. K. (2001). *Estudo de Caso: Planejamento e Métodos*. Porto Alegre: Bookman.

Bitmap Forma de armazenamento de pontos em uma imagem retangular indexada por duas coordenadas (*x* e *y*). Originalmente o formato permitia que cada ponto fosse registrado em duas cores (ex: preto ou branco), mas com o avanço das capacidades de armazenamento e processamento dos computadores o número de cores foi ampliado significativamente até os atuais 2^{64} de possibilidades por ponto.

Conectividade A conectividade refere-se às conexões entre computadores, dispositivos móveis (celulares e *tablets*) e redes de computadores à Internet, permitindo que os usuários acessem serviços *online* tais como a *World Wide Web* e *e-mails*. Há várias formas de oferta de conectividade, sendo a mais comum via provedores de Internet e operadoras de telefonia fixa ou móvel.

Dispositivo de entrada e saída Também denominados *periféricos*, permitem a interação do computador com o usuário, possibilitando a entrada e saída de dados. Teclados, mouses, CD-ROMs, DVD-ROMs, *scanners* e *webcams* são exemplos de dispositivos de entrada, enquanto monitores, caixas acústicas e impressoras são dispositivos de saída.

Plotter Impressora de computador que imprime gráficos baseados em primitivas geométricas – pontos, linhas, curvas e polígonos. Uma *plotter* imprime cópias em alta definição utilizando canetas que riscam diretamente papel, sendo muito utilizada na reprodução de plantas arquitetônicas e desenhos técnicos em diversas áreas.

Sistema Operacional Coleção de programas que gerencia os recursos dos dispositivos (*hardware*) do computador e oferece serviços de acesso comum aos programas (*software*). Sistemas operacionais modernos incluem o Microsoft Windows 7, Apple Mac OS X e Linux.

Programa de Pós-Graduação em Psicologia – PPGP/UFES

Doutorado em Psicologia – Aluno: Hugo Cristo Sant'Anna

Esta pesquisa faz parte de um dos estudos que compõem a tese de doutorado do Prof. Hugo Cristo no Programa de Pós-Graduação em Psicologia da Universidade Federal do Espírito Santo, sob orientação da Profª. Drª. Maria Cristina Smith Menandro. A pesquisa investiga questões cognitivas relacionadas ao aprendizado de fundamentos da Computação por Designers em geral e estudantes de design em particular.

A participação é voluntária e os dados não serão divulgados ou utilizados por terceiros. O e-mail é solicitado exclusivamente para que alguns participantes sejam convidados aleatoriamente para a segunda etapa da pesquisa. Por favor leia atentamente as questões e responda conforme solicitado.

Dúvidas e informações sobre a pesquisa

Prof. Hugo Cristo - Departamento de Desenho Industrial
hugocristo@gmail.com / (27) 4009 2584

1. Sobre você

1.1 Sexo	**1.2 Idade**	**1.3 Ano de entrada na IES**	**1.4 Você sabe programar em alguma linguagem?**
[ ] M [ ] F			[ ] Sim [ ] Não
1.5 Com qual idade aproximada você começou a utilizar computadores?			
1.6 Cite atividades que você mais realiza utilizando computadores, em ordem de importância.			

2. Se você marcou *Sim* no item 1.4 responda as questões abaixo. Se marcou *Não* siga para o item 3.

2.1 Cite a linguagem de programação na qual você é mais fluente.		
2.2 Cite programas, projetos ou outras aplicações que desenvolve nesta linguagem.		
2.3 Como você aprendeu a programar nesta linguagem?		
2.4 Quantos anos você tinha aproximadamente quando aprendeu a programar (em qualquer linguagem)?		
2.5 Marque na lista a seguir quais pessoas próximas a você também sabem programar.		
[ ] Pais	[ ] Irmãos ou irmãs	[ ] Colegas da faculdade
[ ] Colegas do trabalho	[ ] Parentes próximos (tios, primos)	[ ] Amigos
[ ] Outros:		
2.6 Explique o que despertou em você o interesse para aprender a programar. Cite situações, eventos ou objetivos pessoais, acadêmicos ou profissionais que podem estar relacionados.		
2.7 Saber programar já fez diferença na sua vida pessoal, acadêmica ou profissional? Por quê?		

3. Indique três palavras, conceitos ou ideias que vêm à sua cabeça quando pensa nos termos a seguir. Informe as palavras em ordem de importância, da *mais* importante para a *menos* importante.

Termo	**Palavras**		
	1ª mais importante	**2ª mais importante**	**3ª mais importante**
Computação			
Comunicação			
Coordenação			
Armazenagem			
Automação			
Avaliação			
Projeto			
Abstração			
Algoritmo			

4. Explique, com suas palavras, como você entende a relação dos termos a seguir com a Computação:

4.1 Computador
4.2 Comunicação
4.3 Coordenação
4.4 Armazenagem
4.5 Automação
4.6 Avaliação
4.7 Projeto
4.8 Abstração
4.9 Algoritmo

5. Como a *computação* lhe ajuda na solução dos problemas de Design, sejam eles acadêmicos, profissionais ou do seu cotidiano?

6. Como os *computadores* lhe ajudam na solução dos problemas de Design, sejam eles acadêmicos, profissionais ou do seu cotidiano?

7. Informe seu endereço de e-mail caso deseje participar da segunda fase desta pesquisa.

Obrigado!

PROGRAMA DE PÓS-GRADUAÇÃO EM PSICOLOGIA PPGP/UFES – DOUTORADO EM PSICOLOGIA
ALUNO: HUGO CRISTO / ORIENTADORA: MARIA CRISTINA SMITH MENANDRO
PESQUISA COMPUTAÇÃO E DESIGN

Este questionário é a segunda parte de um levantamento sobre os usos da Computação por estudantes de Design. Por favor presponda as questões a seguir considerando seu ponto de vista. Sua identidade será mantida em sigilo e as respostas serão utilizadas exclusivamente em conformidade com os objetivos do trabalho. Informações sobre a pesquisa podem ser obtidas pelo e-mail hugocristo@gmail.com

SOBRE VOCÊ

Idade: __	Ano de ingresso na UFES: ___	Sabe programar? [ ] Sim [ ] Não	Se sim, desde qual idade? ___
Se sabe programar, em qual linguagem você é mais fluente? _____			

PARTE I

1) Avalie os termos abaixo de acordo com a sua compreensão sobre a importância de cada um deles para o que você entende por **Computação**.

	Muito importante	**Pouco importante**
Computador		
Programação		
Tecnologia		

2) Você conseguiria pensar em **Computação** sem lembrar de **Computador**? Por quê?

3) Você conseguiria pensar em **Computação** sem lembrar de **Tecnologia**? Por quê?

4) Você conseguiria pensar em **Computação** sem lembrar de **Programação**? Por quê?

PARTE II

5) Avalie os termos abaixo de acordo com a sua compreensão sobre a importância de cada um deles para o que você entende por **Algoritmo**.

	Muito importante	Pouco importante
Código		
Programação		
Matemática		
Números		

6) Você conseguiria pensar em **Algoritmos** sem lembrar de **Código**? Por quê?

7) Você conseguiria pensar em **Algoritmos** sem lembrar de **Programação**? Por quê?

8) Você conseguiria pensar em **Algoritmos** sem lembrar de **Matemática**? Por quê?

9) Você conseguiria pensar em **Algoritmos** sem lembrar de **Números**? Por quê?

PARTE III

10) Avalie os termos abaixo de acordo com a sua compreensão sobre a importância de cada um deles para o que você entende por **Abstração**.

	Muito importante	Pouco importante
Arte		
Conceito		
Criatividade		
Imaginação		
Pensamento		

11) Você conseguiria pensar em **Abstração** sem lembrar de **Arte**? Por quê?

12) Você conseguiria pensar em **Abstração** sem lembrar de **Conceito**? Por quê?

13) Você conseguiria pensar em **Abstração** sem lembrar de **Criatividade**? Por quê?

14) Você conseguiria pensar em **Abstração** sem lembrar de **Imaginação**? Por quê?

15) Você conseguiria pensar em **Abstração** sem lembrar de **Pensamento**? Por quê?

PARTE IV

16) Avalie os termos abaixo de acordo com a sua compreensão sobre a importância de cada um deles para o que você entende por **Automação**.

	Muito importante	**Pouco importante**
Automático		
Máquina		
Praticidade		
Robótica		
Tecnologia		

17) Você conseguiria pensar em **Automação** sem lembrar de **Automático**? Por quê?

18) Você conseguiria pensar em **Automação** sem lembrar de **Máquina**? Por quê?

19) Você conseguiria pensar em **Automação** sem lembrar de **Praticidade**? Por quê?

20) Você conseguiria pensar em **Automação** sem lembrar de **Robótica**? Por quê?

21) Você conseguiria pensar em **Automação** sem lembrar de **Tecnologia**? Por quê?

Obrigado por sua participação!
Hugo Cristo

Tabela 12.1 - Distâncias absolutas e médias entre termos que coocorrem – Auxílio da Computação na solução de problemas de Design

Termo	*f*	Co-ocorrência	Distâncias absolutas	DEM
ajud*	19	computação	4, 7, 2, 1, 1, 1, 1, 1	2,25
		tudo	3, 6	4,5
		faço	4, 5	4,5
		computador	5, 4	4,5
		pensar	2, 7	4,5
		possível	6, 3	4,5
		sei	1, 10	5,5
auxili*	5	computação	1, 1, 1, 6	2,25
computador	3	tudo	2, 1	1,5
		ajud*	5, 4	4,5
		computação	5, 5	5,5
computação	27	presente	1, 1	1
		permite	1, 1	1
		auxili*	1, 1, 1, 6	2,25
		ajud*	4, 7, 2, 1, 1, 1, 1, 1	2,25
		uso	2, 3, 2	2,33
		design	2, 3	2,5
		soluções	4, 6, 3	4,33
		possível	10, 1, 2	4,33
		sei	3, 8, 3	4,67
		problema*	7, 2, 3, 8	5
		softwares	3, 8, 5	5,33
		tudo	7, 4	5,5
		faço	8, 3	5,5
		pensar	6, 5	5,5
		pequenininha	1, 10	5,5
		computador	9, 2	5,5
		pesquis*	3, 9	6
criação	6	softwares	2, 1, 2	1,67
		programas	6, 3	4,5
		informações	7, 5	6
		dados	3, 7, 9	6,33
dados	21	organizar	1, 1	1
		impressos	2, 1, 2	1,67
		guardar	1, 3	2
		informações	2, 4	3
		variáveis	5, 4, 1	3,33
		trocar	3, 5	4
		banco	10, 1, 2, 5	4,5
		transformação	9, 2, 3, 6	5
		pessoas	4, 6	5
		exemplo	8, 3, 4, 7	5,5
		outras	5, 7	6
		simples	4, 7, 8	6,33
		criação	3, 7, 9	6,33
		tarefas	6, 5, 6, 9	6,5

		programas	3, 8, 9	6, 67
		scripts	2, 9, 10	7
		conectar	6, 8	7
		automatizem	5, 6, 7, 10	7
		processo	8, 10	9
desenvolv*	6	programas	8, 5	6.5
design[66]	3	computação	2, 3	2,5
		softwares	1, 5	3
facili_	11	uso	4, 2	3
forma	5	eficiente	2, 7, 3	4
		produzir	5, 4	4,5
		visível	8, 1	4,5
		conteúdo	6, 3	4,5
		permite	4, 5	4,5
		tornar	7, 2	4,5
		mais	1, 10, 8, 1	5
		imagens	1, 10	5,5
informações	5	dados	2, 4	3
		bom	5, 7	6
		processo	6, 6	6
		outras	9, 3	6
		vezes	4, 8	6
		criação	7, 5	6
		conectar	8, 4	6
		distrair	3, 9	6
		pessoas	10, 2	6
pesquis*	5	computação	3, 9	6
problema*	8	solucionar	1, 1	1
		resolução	1, 1	1
		soluções	3, 2	2,5
		pensar	4, 4	4
		computação	7, 2, 3, 8	5
		sei	8, 5	6,5
programas	13	utilização	1, 1	1
		softwares	1, 2	1,5
		uso	1, 10, 1, 1	3,25
		criação	6, 3	4,5
		projeto*	3, 8, 3	4,67
		internet	9, 1	5
		microsoft office	1, 10	5
		ideias	4, 7	5,5
		armazenar	3, 8	5,5
		desenvolv*	8, 5	6,5
		dados	3, 8, 9	6,67
programação	3	pensar	2, 2	2
		soluções	4, 3	3,5
projeto*	6	uso	2, 9, 2	4,33
		programas	3, 8, 3	4,67
		armazenar	6, 5	5,5
		briefings	9, 2	5,5
		ideias	7, 4	5,5
		microsoft office	4, 7	5,5
sei	6	parece	3, 4	3,5
		computação	3, 8, 3	4,67
		ajud*	1, 10	5,5
		maneira	4, 7	5,5
		problema*	8, 5	6,5

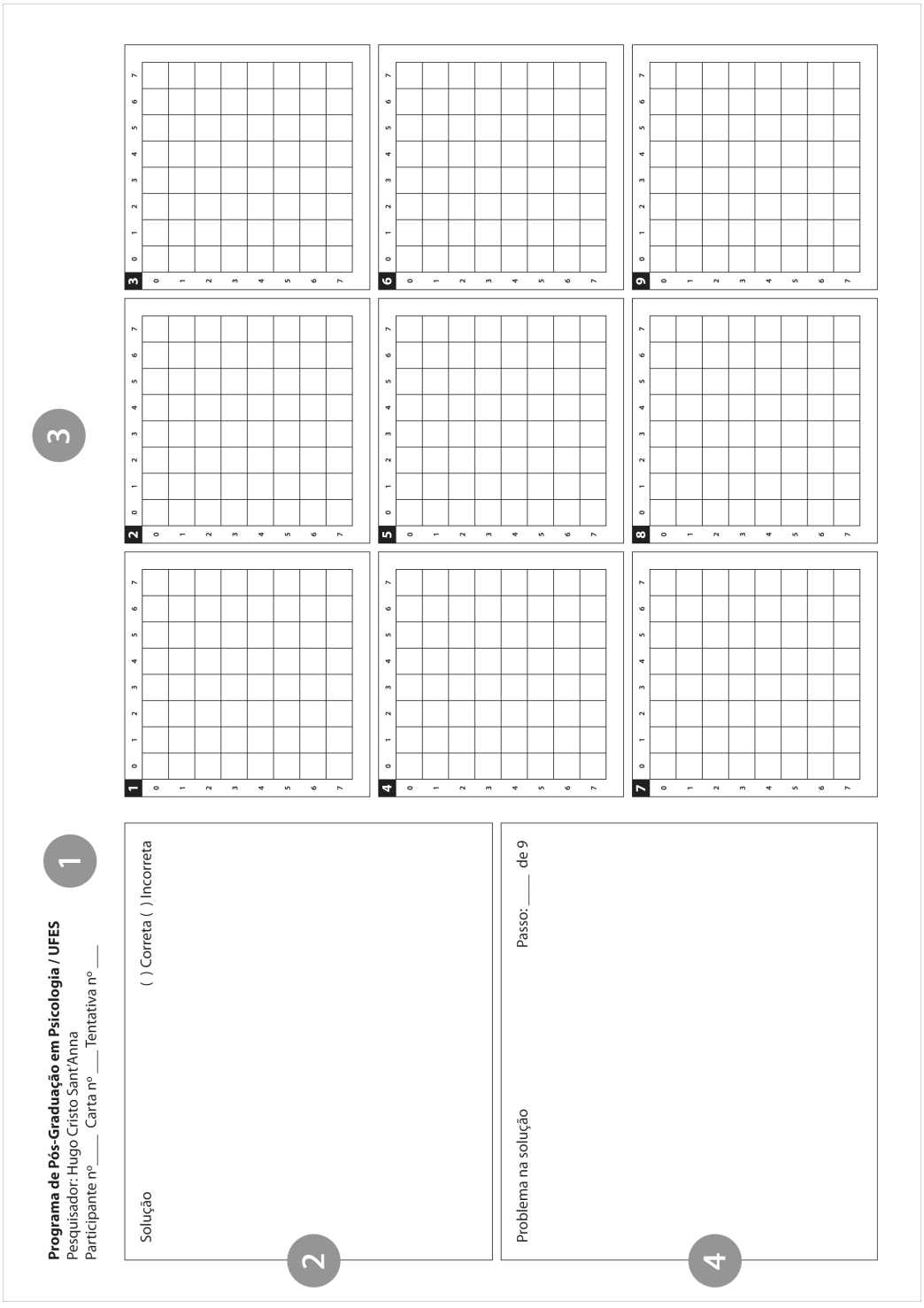
[66] Os termos design e programação foram inclusos na análise mesmo com baixa frequência de coocorrência pela relevância deles para a discussão.

		faço	5, 10	7,5
softwares	10	programas	1, 2	1,5
		utilização	2, 1	1,5
		uso	1, 2	1,5
		criação	2, 1, 2	1,67
		design	1, 5	3
		computação	3, 8, 5	5,33
soluções	5	problema*	3, 2	2,5
		programação	4, 3	3,5
		pensar	6, 1	3,5
		computação	4, 6, 3	4,33

Tabela 12.2 - Distâncias absolutas e médias entre termos que coocorrem – Auxílio dos Computadores na solução de problemas de Design

Termo	***f***	**Co-ocorrência**	**Distâncias absolutas**	**DEM**
acess*	5	facilidade	4, 1	2,5
computador*	31	uso	1, 1, 1	1
		computação	1, 2	1,5
		distinção	2, 3	2,5
		faço	3, 4	3,5
		conseguiria	2, 6	4
computação	7	computador*	1, 2	1,5
		projetos	3, 6	4,5
		faço	3, 4	3,5
		conseguiria	2, 6	4
dados	5	organizar	1, 1	1
		guardar	1, 3	2
		informações	2, 4	3
		trocar	3, 5	4
		pessoas	4, 6	5
		conectar	5, 7	6
		criação	6, 8	7
		processo	7, 9	8
ferramenta	14	utilizar	9, 2	5,5
informações	7	trabalhos	3, 3	3
		dados	2, 4	3
		ver	1, 1, 9	3,67
		conseguiria	6, 2	4
		referências	2, 8	5
		trocar	9, 1	5
		conectar	6, 3	4,5
		distrair	3, 6	4,5
		criação	5, 4	4,5
		pessoas	7, 2	4,5
		guardar	10, 1	5,5
		buscar	1, 10	5,5
permite	6	protótipos	2, 5	3,5
		simula	4, 3	3,5
		criação	1, 6	3,5

		situações	5, 2	3,5
		testes	8, 1	4,5
		demandam	9, 2	5,5
		dinheiro	10, 3	6,5
presente	5	projetos	1, 2	1,5
		madeira	2, 1	1,5
		estar	1, 4	2,5
		computação	2, 5, 5, 3	3,5
processo*	6	facilita	1, 1	1
		informações	4, 4	4
		dados	6, 8	7
programas	7	uso	1, 1, 1	1
		trabalhos	3, 2	2,5
		utilizar	1, 6	3,5
		adobe	1, 6	3,5
		internet	7, 6	6
projetos	5	presente	1, 2	1,5
		comunicação	3, 4	3,5
		computação	3, 4	3,5
		uso	8, 3	5,5
		servem	3, 9	6
softwares	6	uso	1, 2	1,5
		suporte	1, 3	2
		instalados	1, 5	3
uso	13	computador	1, 1, 1	1
		facilidade	1, 1	1
		programas	1, 1, 1	1
		softwares	1, 2	1,5
		projetos	8, 3	5,5
		coisas	9, 6	7,5



Modelo de Plano de Voo I – RocketSocket (Formato original: 210x297mm)

Modelo de Plano de Voo II – Linguagem elaborada pelo participante (Formato original: 210x297mm)

Programa de Pós-Graduação em Psicologia / UFES
Pesquisador: Hugo Cristo Sant'Anna
Participante nº____ Carta nº ___ Tentativa nº ___

1

2

Comandos da sua linguagem e respectivas funções

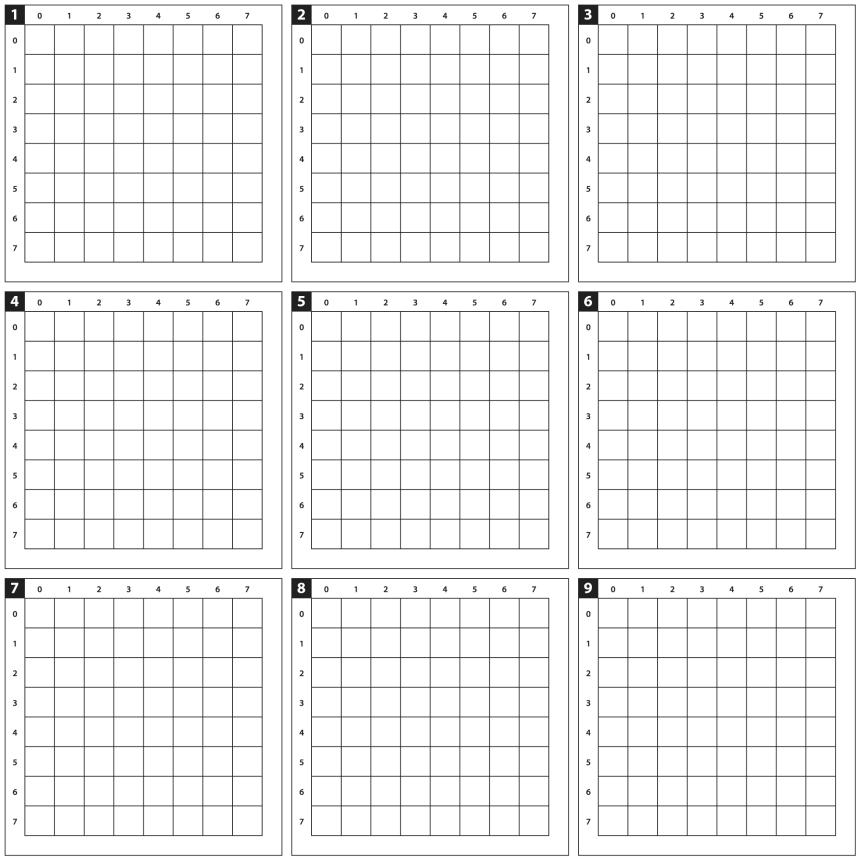
3

Solução ( ) Correta ( ) Incorreta

5

Problema na solução Passo: ____ de 9

4



Programa de Pós-Graduação em Psicologia – PPGP/UFES
Doutorado em Psicologia – Aluno: Hugo Cristo Sant'Anna
Roteiro de entrevista / estudo experimental

Participante nº _____

1. Conceitos gerais

1.1 Como você definiria:

a. O que é um comando em um computador? Cite exemplos.
b. O que é um programa de computador? Cite exemplos.
c. O que é uma linguagem de programação de computadores? Cite exemplos.

2. Conceitos, práticas e perspectivas do Pensamento Computacional

2.1 Sequências

a) Imagine que este tabuleiro é a tela de um computador de bordo de um foguete. Esse computador funciona exatamente como qualquer outro computador que você conhece e pode programá-lo para mover o foguete na tela e coletar estrelas ou desviar de asteroides usando comandos em uma linguagem de programação. Você acha possível a existência de comandos, programas e linguagens de programação fora dos computadores tradicionais? Explique sua resposta.

b) [**Mostrar configuração exemplo semelhante à carta 0, o bloco *mover* e explicar o funcionamento**] Imagine que este bloco é um comando que controla o movimento do foguete. Desenhe na parte superior desta folha [**mostrar mapa**] como você imagina que deveria usá-la para fazer o foguete coletar as estrelas que estão na tela do computador. Nos quadros ao lado você deverá indicar a situação do tabuleiro passo a passo durante a execução dos comandos.

c) [**Mostrar carta 0**] E como você faria para coletar estas estrelas?

Se houver diferença entre o resultado final desejado e o obtido, informar ao participante como marcar o problema no mapa: explicar o problema na caixa inferior, indicar qual passo da execução onde o problema se manifesta e tentar novamente.

d) [**Mostrar carta 1**] Você acha que seria possível coletar estas estrelas? Por quê?

Se o participante responder que sim, solicite a ele que use o mapa como nas questões anteriores. Assim que perceber a impossibilidade, peça a ele que reflita sobre o motivo e escreva suas impressões na parte inferior do mapa. Repetir essa situação até que a representação da tarefa mude.

e) O que você acha que falta para que seja possível fazer a coleta? Explique qual tipo de comando você imagina que seria necessário para concluir a tarefa.

f) [**Mostrar bloco girar 90º**] Este comando girará o foguete. Você acha que agora é possível realizar a coleta das estrelas [**da carta 1**]? Por quê?

Se o participante disser que continua **não sendo possível fazer a coleta**, mostrar uma variação da **carta 1** como exemplo. Se o participante **entender o princípio**, solicitar que refaça a **carta 1.**

g) Você se lembra das definições que me deu sobre o que seriam comandos, programas e linguagens de programação de computadores? Você acha que esses três conceitos estão presentes na tarefa que acabou de realizar **[carta 1]** ? Explique sua resposta.

2.2 Laços

a) Você já ouviu falar de laços ou loops na programação de computadores?

- Se sim, solicitar que o participante cite exemplos.
- Se não, explicar o conceito.

b) [**Mostrar tabuleiro na carta 0 com o código com e sem loops**] Observe como esse programa que você construiu pode ser refeito utilizando loops. Você consegue explicar como os dois programas atingem o mesmo resultado?

- Se sim, solicitar que o participante explique.
- Se não, explicar o que houve.

c) [**Mostrar tabuleiro na carta 2**] Você conseguiria programar o foguete para coletar estas estrelas utilizando loops?

3. Linguagens e Representações

3.1 Construção da linguagem própria

a) Considerando os códigos criados anteriormente, você conseguiria criar sua própria linguagem de programação para controlar o foguete?

- Se sim, solicitar que o participante indique quais comandos a linguagem teria.

b) Se não, perguntar o porquê e fornecer o exemplo do bloco **mover** como ilustração da possibilidade e solicitar que ele tente criar a linguagem.
c) Partindo dos comandos que você criou, como seria um programa para coletar as estrelas [**carta 3**]?
d) E da [**carta 4**]?

3.2 Abstrações e automações

a) Observe o tabuleiro da [**carta 5**]. Você consegue visualizar algum padrão que se repete? Qual?

Se o participante disser que não, solicitar que ele construa o código para resolver o trajeto e perguntar novamente até que ele perceba o padrão.

b) Imagine que você poderia criar um novo bloco para a linguagem ou um novo comando na sua linguagem que ajudasse na programação desse trajeto. O que esse bloco faria? Explique.

Solicitar que o participante utilize a opção gravar dos blocos para criar o comando nomeado como **a**. Na sequência exemplifique o uso da abstração na resolução do trajeto.

c) Tente fazer o mesmo processo de reconhecer um padrão e criar um novo bloco ou comando na [**carta 6**].
d) Tente fazer o mesmo processo de reconhecer um padrão e criar um novo bloco ou comando na [**carta 7**].
e) Você achou mais fácil ou mais difícil de resolver o trajeto com os novos blocos? Por quê?
f) Observando as outras cartas [**1 a 5**], tente reconhecer padrões e sugerir novos blocos.

Obrigado!

Apresentação Geral

Imagine que este tabuleiro é a tela de um computador de bordo de um foguete. Esse computador funciona exatamente como qualquer outro computador que você conhece e pode programá-lo para mover o foguete na tela e coletar estrelas ou desviar de asteroides usando comandos em uma linguagem de programação.

Elementos Primitivos e Meios de Combinação

Para programar o foguete, você utilizará peças como estas [exibir bloco de comando mover], que deverão ser encaixadas para criar sequências de movimentos [encaixar dois blocos mover e simular resultado no tabuleiro]. O trajeto a ser percorrido pelo foguete resultará da sequência dos comandos que você construir.

Para iniciar um programa, você deverá utilizar o bloco ligar o motor [exibir bloco], encaixar os comandos desejados na sequência que pretende executá-los [encaixar dois blocos de movimento seguidos como exemplo] e por fim desligar o motor [exibir bloco] para concluir o voo do foguete.

À medida em que você for avançando nos problemas, poderá utilizar novas peças que serão fornecidas.

Funcionamento das Cartas

Antes de construir o trajeto para o foguete, você receberá uma carta com as instruções daquela missão. Cada carta contém a posição das estrelas a serem coletadas e os asteroides a serem evitados, além dos blocos de comando permitidos na solução.