UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

ÉRICO MARCELO HOFF DO AMARAL

PROCESSO DE ENSINO E APRENDIZAGEM DE ALGORITMOS INTEGRANDO AMBIENTES IMERSIVOS E O PARADIGMA DE BLOCOS DE PROGRAMAÇÃO VISUAL

Porto Alegre

2015

ÉRICO MARCELO HOFF DO AMARAL

PROCESSO DE ENSINO E APRENDIZAGEM DE ALGORITMOS INTEGRANDO AMBIENTES IMERSIVOS E O PARADIGMA DE BLOCOS DE PROGRAMAÇÃO VISUAL

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Informática na Educação

Profa. Dra. Liane Margarida Rockenbach Tarouco
Orientadora

Profa. Dra. Roseclea Duarte Medina
Coorientadora

Porto Alegre

2015

CIP - Catalogação na Publicação

Hoff do Amaral, Érico Marcelo
PROCESSO DE ENSINO E APRENDIZAGEM DE ALGORITMOS INTEGRANDO AMBIENTES IMERSIVOS E O PARADIGMA DE BLOCOS DE PROGRAMAÇÃO VISUAL / Érico Marcelo Hoff do Amaral. -- 2015.
255 f.

Orientadora: Liane Margarida Rockenbach Tarouco.
Coorientadora: Roseclea Duarte Medina.

Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Centro de Estudos Interdisciplinares em Novas Tecnologias na Educação, Programa de Pós-Graduação em Informática na Educação, Porto Alegre, BR-RS, 2015.

1. Algoritmos. 2. Programação. 3. Ambientes Imersivos. I. Rockenbach Tarouco, Liane Margarida , orient. II. Duarte Medina, Roseclea, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Porto Alegre

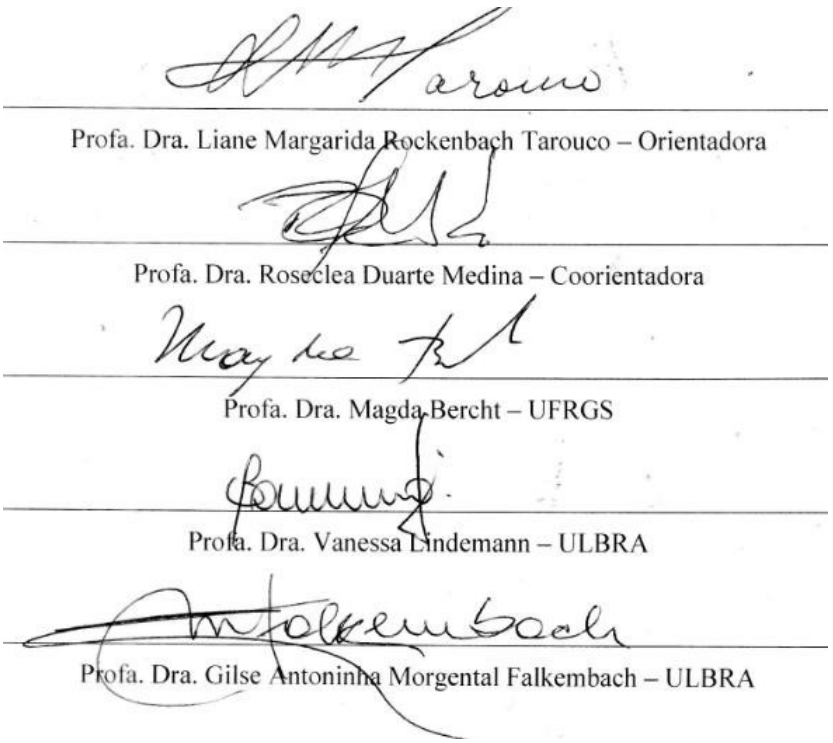
2015

Érico Marcelo Hoff do Amaral

PROCESSO DE ENSINO E APRENDIZAGEM DE ALGORITMOS INTEGRANDO AMBIENTES IMERSIVOS E O PARADIGMA DE BLOCOS DE PROGRAMAÇÃO VISUAL

Tese apresentada ao Programa de Pós-Graduação em Informática na Educação de Educação do Centro Interdisciplinar de Novas Tecnologias na Educação da Universidade Federal do Rio Grande do Sul como requisito para obtenção do título de Doutor em Informática na Educação.

Aprovada em: 20 de agosto de 2015.



Profa. Dra. Liane Margarida Rockenbach Tarouco – Orientadora

Profa. Dra. Roseclea Duarte Medina – Coorientadora

Profa. Dra. Magda Bercht – UFRGS

Profa. Dra. Vanessa Lindemann – ULBRA

Profa. Dra. Gilse Antoninha Morgental Falkembach – ULBRA

RESUMO

A dificuldade dos estudantes em relação à aprendizagem de Algoritmos é um assunto que se renova, devido à grande demanda de mercado em relação a profissionais da área de Tecnologia da Informação. A complexidade na construção do raciocínio lógico e do Pensamento Computacional leva a altos índices de evasão e reprovação em disciplinas introdutórias de programação. A concepção tradicional no ensino de Algoritmos possui grandes desafios, da elaboração de materiais interessantes, que estimulem o aluno a resolver problemas, até o uso de novas tecnologias, a fim de motivar o discente em um ambiente dinâmico. Entendendo essas circunstâncias, esta tese tem como foco propor uma concepção diferenciada para o processo de ensino de Algoritmos, caracterizada por uma prática pedagógica baseada na integração de ambientes imersivos a paradigmas de programação, diferentes dos comumente utilizados em sala de aula. Os resultados alcançados com esta proposta mostraram que as características de interação e imersão, proporcionadas pelo mundo virtual, aliadas a ferramentas de programação em blocos visuais, motivam e estimulam a atenção dos alunos, tornando-os protagonistas no desenvolvimento de seus saberes sobre algoritmos e programação. Estas afirmações estão calcadas na observação dos experimentos realizados, com base na metodologia de ensino proposta, consolidados a partir de uma inferência estatística apurada sobre os resultados dos alunos na disciplina de Algoritmos e programação nos três semestres avaliados e, por uma análise detalhada sobre a evolução destes estudantes durante todas as atividades da última etapa do estudo. Estas análises mostraram efetivamente a evolução dos estudantes, tanto na construção do raciocínio lógico, quanto na melhoria do seu desempenho acadêmico.

Palavras-Chave: algoritmos, programação, ambientes imersivos

AMARAL, Érico M. H. Processo de ensino e aprendizagem de algoritmos integrando ambientes imersivos e o paradigma de blocos de programação visual. Porto Alegre, 2015. 264 f. Tese (Doutorado em Informática na Educação) Programa de Pós-graduação em Informática na Educação, Centro Interdisciplinar de Novas Tecnologias na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

ABSTRACT

Difficulties of students towards learning algorithms are a topic that is being renewed constantly, due to the large market demand in relation to professionals in the field of Information Technology. The complexity in the construction of logical reasoning and Computational Thinking leads to high rates of evasion and failure in introductory programming courses. Traditional concept in teaching algorithms has great challenges, since the development of interesting materials that encourage students to solve problems, until to the use of new technologies in order to motivate the students in a dynamic environment. Understanding these circumstances, this thesis focuses on proposing a different design to the process of teaching algorithms, characterized by a pedagogical practice based on the integration of immersive environments with programming paradigms, different from those commonly used in the classroom. The results obtained with this proposal showed that the interaction and immersion features offered by the virtual world, combined with programming tools in visual blocks, motivate and stimulate student's attention, making them protagonists in developing their knowledge about algorithms and programming. These statements are justified by the observation of experiments conducted, based on the proposed teaching methodology, consolidated from a statistical inference calculated on the results of the students in the discipline of algorithms and programming in three semesters evaluated, and, for a detailed analysis of the evolution of these students during all activities of the last stage of the study. These analyzes showed effectively the progress of students, both in the construction of logical reasoning and in improving their academic performance.

Keywords: algorithms, programming, immersive environments

AMARAL, Érico M. H. Processo de ensino e aprendizagem de algoritmos integrando ambientes imersivos e o paradigma de blocos de programação visual. Porto Alegre, 2015. 264 f. Tese (Doutorado em Informática na Educação) Programa de Pós-graduação em Informática na Educação, Centro Interdisciplinar de Novas Tecnologias na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

LISTA DE ABREVIATURAS E SIGLAS

A4	Ambiente de Aprendizagem Adaptado para Algoritmos
AIA	*App Inventor for Android*
AVA	Ambiente Virtual de Aprendizagem
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CSCL	*Computer Supported Collaborative Learning*
CSCW	*Computer Supported Cooperative Work*
DC	Dimensão de Competências
DCN	Diretrizes Curriculares Nacionais
DH	Dimensão de Habilidades
EaD	Educação a Distância
IES	Instituição de Ensino Superior
IGUAL	*Innovation for Equality in Latin American Universities*
JS	Java Script
JSP	*Java Server Pages*
LSL	*Linden Script Language*
MOJO	Módulo de Integração com Juízes Online
MOODLE	*Modular Object-Oriented Dynamic Learning*
MV	Mundo Virtual
NUI	*Nature User Interface*
OpenSim	*Open Simulator*
PBD	Pesquisa Baseada em *Design*
PC	Pensamento Computacional
PDI	Plano de Desenvolvimento Institucional
PPC	Projeto Pedagógico de Curso
PPI	Plano Pedagógico Institucional
RA	Realidade Aumentada
RBC	Raciocínio Baseado em Casos
RV	Realidade Virtual
SBC	Sociedade Brasileira de Computação
SL	Second Life
SLOODLE	*Simulation Linked Object Oriented Dynamic Learning*
SPE	Sistema Personalizado de Ensino
TAPC	Taxonomia para Avaliação do Pensamento Computacional
TCC	Trabalho de Conclusão de Curso
TCC	Teoria da Carga Cognitiva
UFAL	Universidade Federal de Alagoas
UFAM	Universidade Federal do Amazonas
UFPA	Universidade Federal do Pará

UFPB	Universidade Federal da Paraíba
UFRA	Universidade Federal Rural do Amazonas
UFSC	Universidade Federal de Santa Catarina
UFSM	Universidade Federal de Santa Maria
UNICAMP	Universidade Estadual de Campinas
UNIPAMPA	Universidade Federal do Pampa

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE EQUAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Vive-se atualmente em um mundo de constantes mudanças sociais, culturais e tecnológicas, as quais impelem diferentes desafios à humanidade e, através destes, o indivíduo se desenvolve enquanto ser, em um ambiente adverso. Contudo, esta é uma tarefa que exige, a cada dia, uma maior capacidade de adaptação ao meio, denotando novas demandas para o cidadão do século XXI, o qual deve estar apto a reconhecer e lidar com estas mudanças. O uso das tecnologias é um dos grandes alicerces do novo cenário, onde as pessoas estão imersas em um ambiente no qual a computação é cada vez mais ubíqua. Utilizar estes recursos tecnológicos para tomar decisões apropriadas no dia a dia exige do cidadão a faculdade de saber manipular diferentes sistemas, aplicando conceitos básicos de lógica de programação.

Ao apresentar esta nova concepção, percebe-se a importância do ensino e aprendizagem da Lógica de Programação e Algoritmos em todos os meios, principalmente em ambientes de ensino superior. É comum encontrar pelo menos uma disciplina relacionada com a computação, no currículo de cursos de nível superior, de áreas que não a da computação. Ao adotar-se esta linha de ação, relacionada ao ensino da Lógica e Programação, é possível perceber a dificuldade inerente à abstração e construção do raciocínio lógico dos estudantes, especialmente no que tange à forma de compreensão de esquemas algorítmicos.

Na formação de profissionais que irão atuar no desenvolvimento dos sistemas usando TIC, em que é necessária a presença de disciplinas voltadas ao estudo da computação em si, constata-se que existe falta de profissionais de programação, visto que a evolução tecnológica provoca uma grande demanda de tal mão-de-obra. Esta formação esbarra em problemas que ocorrem já nas primeiras disciplinas do curso quando os estudantes começam a ter que trabalhar com Algoritmos e Introdução à Programação.

Para Santos e Costa (2006) o fato do estudante não conseguir, de forma simples, construir o raciocínio lógico pode ser considerado como um dos principais fatores que

corroboram para os altos índices de evasão nas disciplinas de programação e, também dos cursos superiores que possuem estas áreas como elementos curriculares.

Um estudo apresentado pelo Instituto Nacional de Estudos e Pesquisas (INEP) aponta para dados preocupantes relacionados ao desempenho e continuidade dos alunos nas instituições brasileiras de ensino superior (IES), em disciplinas que envolvam lógica de programação nos cursos na área das Engenharias, Ciências, Matemática e Computação, em que é identificado um índice médio nacional de 23% de reprovação e evasão (BRASIL, 2012). Corroborando com estes dados, o Governo Federal relata sua preocupação com o tema, visto que a evasão em cursos de nível superior, em algumas áreas, apresentam índices alarmantes (MEC, 2007).

Especificamente nas áreas de programação e algoritmos, alguns pontos são de extrema importância para que sejam identificadas as causas relacionadas às dificuldades de aprendizagem apresentadas pelos alunos, que por sua vez influenciam tais níveis de evasão.

A dificuldade na construção do Pensamento Computacional (PC) pode ser considerada uma delas, visto que Wing (2006) descreve o PC como um conjunto de competências e habilidades básicas e específicas, apresentadas pelo indivíduo, relacionadas à área de computação. Estas competências abrangem de forma geral a capacidade da utilização de recursos computacionais e estratégias algorítmicas para a resolução de problemas, por meio da abstração e decomposição lógica dos mesmos. Conforme defendido pela UNESCO (2009):

> Para viver, aprender e trabalhar bem em uma sociedade cada vez mais complexa, rica em informação e baseada em conhecimento, os alunos e professores devem usar a tecnologia de forma efetiva, pois em um ambiente educacional qualificado, a tecnologia pode permitir que os alunos se tornem: usuários qualificados das tecnologias da informação; pessoas que buscam, analisam e avaliam a informação; solucionadores de problemas e tomadores de decisões; usuários criativos e efetivos de ferramentas de produtividade; comunicadores, colaboradores, editores e produtores; cidadãos informados, responsáveis e que oferecem contribuições.

Portanto, embora Barcelos & Silveira (2012) estabeleçam que o Pensamento Computacional pode auxiliar o estudante no processo de aprendizagem de área específicas como programação, matemática, lógica entre outras, ele é necessário em todas as áreas do conhecimento, pois compreende o que os estudantes deveriam saber e ser capazes de fazer para aprender efetivamente e viver produtivamente em um mundo cada vez mais digital (NETS, 2007).

No caso específico dos estudantes de cursos de computação, foi constatada uma dificuldade relacionada diretamente ao raciocínio lógico e suas relações, pois estas podem ser

consideradas alicerces para a construção do conhecimento sobre a programação de computadores (DOS SANTOS & COSTA, 2006; GOMES *et al*. 2014).

Neste contexto, Ferrandin & Stephani (2005) definem a lógica como a arte de pensar corretamente e, visto que a forma mais complexa do pensamento é o raciocínio, a lógica estuda ou tem em vista a correção do raciocínio. Este conceito é observado no desenvolvimento de um algoritmo, pois esta ação envolve tal estrutura lógica e de raciocínio, tendo como descrição uma sequência de passos que deve ser seguida para a realização de uma tarefa a partir de um estado inicial, que após um período de tempo finito, produzirá um estado final previsível e bem definido (CAMPOS, 2003).

O nível de complexidade inserido na ação de descrever um algoritmo pode ser um fator relevante. Porém é importante considerar que a proficiência em programação pode ser analisada de uma forma que transcende o simples fato de escrever um conjunto de linhas de código, sobre um raciocínio lógico, em uma linguagem qualquer, ação que pode ser descrita como uma arte e uma ciência. Arte, pois exige criatividade e, ciência, por se basear em um conjunto de regras orientadoras, calcadas sobre um raciocínio lógico, alinhado a métodos rigorosos de programação, segundo Gomes *et al.* (2008).

Para Wirth (1976) a construção de um programa é realizada a partir de formulações concretas destes algoritmos, com base em representações e estruturas específicas, as quais, para atender a um problema, devem respeitar uma determinada codificação (KNUTH, 1968). A ordenação desta codificação pode ser considerada por Dijkstra (1976) como uma técnica de notação para programar, auxiliando assim o raciocínio algorítmico. Forbellone (2005) afirma, neste contexto, que o uso correto deste raciocínio e da simbolização formal na programação de um computador, a fim de disponibilizar soluções coerentes para problemas, é definido como lógica de programação.

No estudo sobre as causas que norteiam as dificuldades dos alunos na construção do conhecimento sobre algoritmos e programação, destacam-se alguns pontos relevantes. Inicia-se pelo simples fato que o estudo de algoritmos e programação não é uma tarefa fácil para alunos iniciantes (SLEEMAN, 1986). Contudo, para Dijkstra (1989), este é um processo moroso e gradual; O entendimento da complexidade relacionada à sintaxe, necessária para construção de códigos, com a linguagem de programação, é um dos principais fatores apontados pela pesquisa de Motil & Epstein (2000); Perkins (1989) relaciona a dificuldade na aprendizagem diretamente relacionada à necessidade do treinamento intensivo sobre

resolução de problemas, os quais exigem diferentes competências dos alunos; a falta de motivação do aluno também é pontuada como fator negativo para a aprendizagem (BEREITER & NG, 1991). Por fim, Jenkins (2002) descreve a incapacidade na resolução de problemas e processos de aprendizagem não adequados aos estilos cognitivos dos alunos. Borges (2000) aponta também outro conjunto de fatores, como: a falta de acompanhamento docente individualizado; dificuldade do estudante em desenvolver o raciocínio lógico, a ponto que se adaptaram a decorar conteúdos; falta de persistência mediante a complexidade e dificuldade imposta pela disciplina de algoritmos; baixa capacidade de abstração no início do curso superior; o "modelo tradicional" de ensino, em que os discentes são meros receptores, não participativos; ausência, em algumas instituições, de metodologias de ensino focadas nos conteúdos de lógica, as quais incentivem a utilização de aplicações de apoio à aprendizagem. Na mesma linha de pesquisa, Faria & Lima (2011) apresentam por meio de um quadro (Quadro 01) outro conjunto de dificuldades identificadas em disciplinas de programação, porém agora descrevendo propostas de soluções para cada problema, além de citar qual a teoria de aprendizagem adotada para cada proposta.

Quadro 1–Dificuldades e Soluções para a aprendizagem de Algoritmo/Lógica de programação.

Dificuldades/ Entraves	Autores	Soluções Propostas	Teoria de Aprendizagem
Desenvolvimento do Raciocínio Lógico	SILVA(2009) SOUTO(2009 VIEIRA(2008) FALCKEMBACH(2011)	• Estratégia Ascendente de solução de problemas • Jogos Educacionais	Construtivismo
Criação de Abstrações	SILVA(2009) MACHADO(2011) FALCKEMBACH(2011)	• Ilustração de Algoritmos • Software Especialista	Construtivismo
Inexperiência em Práticas Comerciais/Industriais	MATTOS(2008)	Simulação de Práticas Comerciais/Industriais em Jogos e Estudos de Caso	Sócio-Construtivismo
Impossibilidade de Acompanhamento Individualizado	OLIVEIRA(2008) PIMENTEL(2003)	Monitoria com alunos que dominem o conteúdo	Sócio-Construtivismo
Disparidade de Conhecimento e Ritmo de Aprendizagem	MACHADO(2012) PIMENTEL(2003) AUSUBEL (2003)	Aulas de Nivelamento, Objetos de Aprendizagem	Aprendizagem Significativa
Baixa Compreensão de enunciados dos algoritmos	PIMENTEL(2003) AUSUBEL (2003)	Aulas de Interpretação de Texto	Aprendizagem Significativa

Fonte: Adaptado de Faria & Lima, 2011

A adoção de uma metodologia de ensino, dita tradicional, pode ser considerada também como um dos fatores agravantes dos problemas descritos, pois este modelo muitas vezes não é

adequado para a função de motivar o estudante a se interessar pelo conteúdo, a ponto de compreender a sua importância (PETRY, 2005). A forma expositiva, não clara, na abordagem dos conteúdos, focada exclusivamente na resolução de problemas e exercícios, contribui para falta de motivação e baixo rendimento do aluno (ROCHA, 2006). Um conjunto de causas relacionadas com o baixo desempenho dos alunos é apresentado no Quadro 02, apontando características observadas em docentes e discentes.

Quadro 2–Causas para o baixo desempenho de alunos em algoritmos, relação professor e discente.

Causas relacionadas diretamente ao docente	Causas relacionadas diretamente ao discente
• Incapacidade de reconhecimento de habilidades dos alunos impedindo traçar estratégias pedagógicas apropriadas; • Falta de criatividade do docente para apresentar novas técnicas de solução de algoritmos; • Inabilidade em promover a cooperação e colaboração entre os alunos; • Formação pedagógica deficitária; • Visão instrumental sobre o ensino; • Uso de metodologia inadequada, gerando a desmotivação para aprendizagem de Algoritmo; • Planejamento e execução das aulas sem objetivos preestabelecidos ou mal dimensionados; • Ensino conjunto de algoritmo e linguagem de programação; • Ensino focado em determinada linguagem de programação.	• Deficiência no raciocínio operatório-formal, que é a base para o raciocínio lógico; • Hábito de decorar ao invés de compreender o conteúdo; • Dificuldade de formar abstrações; • Baixa capacidade de interpretação e compreensão de enunciados dos algoritmos e de textos em geral; • Falta de experiência com práticas comerciais e industriais; • Baixo nível de conhecimento em Matemática; • Ineficiência em armazenar e classificar mentalmente o conteúdo estudado.

Fonte: Faria, 2013

Algumas soluções para o ensino de algoritmos vislumbram a construção do Pensamento Computacional, já nos primeiros anos escolares, conforme é apresentado pelo Departamento de Educação Britânico em sua orientação estatutária a qual insere a computação no currículo nacional da Inglaterra, publicado em setembro de 2013 (UK, 2013). Esta proposta busca uma educação em computação de alto nível, fazendo com que o estudante utilize o Pensamento Computacional para resolver problemas de forma criativa, disseminando a cultura digital e dotando o aluno atual de competências necessárias para se desenvolver e, por meio da tecnologia da informação, construir programas, se tornando assim participante ativo do mundo virtual. A estrutura deste currículo visa, entre tantos objetivos, fazer com que o estudante seja capaz de compreender como interpretar uma situação ou problema e abstrair a mesma, por meio da lógica de programação e algoritmos, construindo soluções

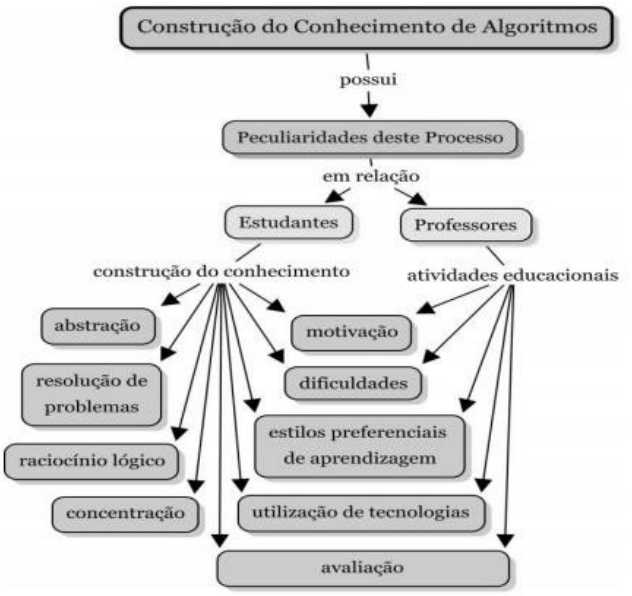
computacionais para tais problemas. O projeto é baseado em quatro estágios chave, cada um com objetivos específicos, dentre os quais são descritos aqui apenas os relacionados à área de programação:

- No estágio um, o aluno deve entender o que são algoritmos, usar o raciocínio lógico para depurar problemas;
- No segundo estágio desenvolver o *design* de soluções e construir programas com objetivos específicos, detectar e corrigir erros;
- No estágio três, o aluno é motivado a entender algoritmos que reflitam o Pensamento Computacional, que permitam o entendimento da lógica, por meio de algoritmos de pesquisa, ordenação e classificação e;
- O quarto estágio é voltado para que o discente construa o conhecimento sobre diferentes aspectos de programação, ciência da computação e tecnologias de informação com profundidade, a fim de que possam aplicar suas habilidades de Pensamento Computacional.

Para validação e efetividade deste programa de estudos, um guia para professores primários foi implementado a fim de desmistificar o uso do novo currículo e, também preparar o docente para desenvolver suas competências em relação à adoção das tecnologias em sala de aula (NAACE, 2014).

Uma estrutura pertinente para o ensino de algoritmos é proposta por Barcelos (2012), em que o autor apresenta a construção do conhecimento com base em dois cenários distintos, o perfil de aluno e de professor, conforme a Figura 01.

Figura 1–Características do ensino de algoritmos



Fonte: Barcelos, 2012

Em ambos os cenários tanto o estudante, quanto o professor são motivados. No perfil do aluno tem-se como alvo o incentivo de suas competências, enquanto o professor é incentivado a elaborar processos de ensino e de aprendizagem diferenciados, focando no processo de construção do conhecimento sobre programação.

1.1. MOTIVAÇÃO

O estudo proposto busca investigar estratégias mais adequadas para estimular os processos de ensino e de aprendizagem dos estudantes com vistas ao desenvolvimento de algoritmos e programação. A realização desta pesquisa foi motivada por diversos fatores:

O reconhecimento da falta de mão-de-obra especializada na área de desenvolvimento de *software* no mercado nacional, conforme apresentado pela sociedade Softex[1], Organização de Sociedade Civil de interesse Público, responsável pela gestão do programa governamental para a Promoção da Excelência de *Software* Brasileiro, que publicou por meio de seu

[1] A Associação para Promoção da Excelência do *Software* Brasileiro (Softex) executa, desde 1996, iniciativas de apoio, desenvolvimento, promoção e fomento para impulsionar a Indústria Brasileira de *Software* e Serviços de TI, uma das maiores em todo o mundo, conhecida por sua criatividade, competência e fonte de talentos. Disponível em: http://www.softex.br/a-softex/

observatório uma pesquisa sobre vagas não preenchidas para profissionais da área de TI, na qual destacaram-se os postos de trabalho para desenvolvedores/programadores com 86,1% de vagas não preenchidas (SOFTEX, 2009). Ainda em pesquisa publicada pelo IDC (2014), a demanda de profissionais, no Brasil, será 32% maior que a oferta de mão-de-obra disponível para o ano de 2015. Segundo a Hays (2014), empresa especializada em recrutamento internacional, o perfil de profissionais em programação foi um dos mais procurados em 2013, contudo a demanda ainda está muito acima da oferta de desenvolvedores disponível no mercado.

Outro fator pertinente é a carência de métodos ou modelos didáticos complementares, com maior nível de efetividade, que possam ser adotados para auxiliar o educando na tarefa de construir o conhecimento sobre a lógica e programação.

E por fim, a percepção da necessidade de construção do Pensamento Computacional, aspecto discutido pelo grupo de trabalho 6 *"Advancing computational thinking in 21st century learning"* durante a EDUSummit´2013, encabeçado por Chris Dede (Havard University), Punya Mishra (Michigan State University) e Joke Voogt (University of Twente) os quais apontam tal pensamento como uma das principais competências necessárias ao estudante do século XXI, a fim de preparar este para o futuro e para as rápidas mudanças mundiais, desta maneira devendo compor o currículo em séries iniciais, conforme reflexão de profissionais de diferentes nações como Inglaterra, EUA e Países Baixos (EDUSUMMIT, 2013 e UNESCO, 2012).

O estímulo deste projeto pode ser compreendido ao entender que a tarefa de auxiliar na construção do Pensamento Computacional e por consequência do ensino de programação, apresenta um alto nível de complexidade, visto que para um estudante chegar ao ponto de implementar um *software* por meio de uma linguagem, o mesmo deve, segundo Farias *et al.* (2012) e Gerhard & Brusilovsky (2001), apresentar algumas competências, como: possuir a capacidade de abstração e raciocínio lógico para propor uma solução; reconhecer a sintaxe e semântica de uma linguagem de programação; entender a forma correta de utilizar um ambiente de programação; reconhecer o processo de compilação, teste e depuração do código fonte e; por fim desenvolver a habilidade de solucionar problemas.

1.2. PROBLEMA DE PESQUISA

É clara a demanda imposta ao cidadão do século XXI em construir o Pensamento Computacional, a fim de que este consiga de forma satisfatória entender e lidar com as tecnologias de informação e programação atualmente disponíveis. Como fator crítico para a construção deste conhecimento citam-se a ineficiência das metodologias e práticas de ensino, as quais são adotadas como padrão para os processos de ensino e de aprendizagem de algoritmos. Ao identificar o contexto apresentado, este estudo almeja disponibilizar subsídios para responder a seguinte problemática: Como estimular a construção do Pensamento Computacional, por meio de uma prática pedagógica para o ensino de algoritmos, pautada na integração do uso de linguagens de programação com blocos visuais, ambientes imersivos e uma linguagem de programação estruturada tradicional?

1.3. OBJETIVO GERAL

Este projeto contempla a realização de um estudo sobre as dificuldades de aprendizagem em algoritmos e programação, propondo uma abordagem alternativa para estimular os processos de ensino e de aprendizagem, através de uma prática diferenciada, utilizando como instrumentos: um ambiente de blocos de programação visual, para introdução à lógica e programação; uma linguagem de programação para mundos virtuais, a fim de motivar o estudante na construção de soluções que podem ser observadas em um ambiente realístico em 3D e; uma linguagem de programação tradicional, linguagem C, fazendo com que o estudante consiga codificar estruturas algorítmicas para solução de diferentes problemas.

O uso do ambiente Scratch para a programação em blocos visuais é utilizado para a capacitação dos alunos em relação aos conceitos e algoritmos clássicos. Posteriormente, um ambiente derivado do Scratch é utilizado para o projeto e desenvolvimento de elementos multimídia interativos para um ambiente imersivo 3D (OpenSim).

Para alcançar tais objetivos, os conceitos propostos por alguns autores são organizados e integrados, de forma a constituir experimentos válidos e calcados sobre fortes bases teóricas, dentre as quais se destacam: Ausubel (2003) com a teoria da aprendizagem significativa; Kolb

(1997), com sua proposta de ciclo da aprendizagem experiencial; o ensino multimídia conforme proposto por Mayer (2001); a teoria da carga cognitiva defendida por Sweller (2005); aplicação das teorias cognitivas, segundo Tarouco & Cunha (2006); Wiley (2001), e seu conceito de objetos de aprendizagem entre outros.

1.3.1. OBJETIVOS ESPECÍFICOS

Nesta subseção são descritos os objetivos específicos elencados para atingir a proposta desta pesquisa:

- Desenvolvimento do estudo e levantamento do referencial sobre o ensino e aprendizagem de algoritmos e programação, buscando os principais autores e obras que deem subsídios a este estudo;
- Definição de padrões metodológicos a serem adotados, a fim de nortear as ações de pesquisa e aprendizagem envolvidas no processo de busca de um modelo efetivo para o ensino de algoritmos e programação;
- Avaliação de temas, trabalhos científicos e correlatos, publicados em eventos, periódicos e revistas, os quais possam delimitar ou até mesmo situar aspectos importantes relacionados ao estudo nesta tese;
- Definir o modelo e experimentos a serem realizados, assim como a ordem e método de aplicação dos mesmos, a fim de identificar qual metodologia melhor atende às necessidades do aluno, focando na construção do conhecimento sobre algoritmos e programação;

1.4. MÉTODOS E PRÁTICAS

Esta pesquisa foca em uma proposta evolutiva para a abordagem do ensino tradicional de uma linguagem de programação, a qual se origina na ideia do desenvolvimento em sala de aula de uma aplicação, partindo da análise dos seus requisitos, modelagem dos dados e, por

fim, a implementação do código, tendo como guia os conceitos da linguagem. A este conjunto de ações serão agregados diferentes tecnologias e conceitos de programação, como: programação em blocos visuais e o desenvolvimento pautado sobre características de imersão. Todas estas ações estarão calcadas sobre os princípios da metodologia de Pesquisa Baseada em *Design* (PBD), defendida por Wang e Hannafin (2005) como um método eficiente para atender demandas de pesquisa relacionadas ao processo de ensino e de aprendizagem, permeadas pela utilização da TIC e de teorias educacionais, focadas na construção de intervenções em um ciclo composto pela análise, desenvolvimento, avaliação, *design* e reconhecimento dos problemas. A PBD integra o estudo, planejamento, desenvolvimento de experiências, ambientes, materiais ou práticas educativas em contextos reais de ensino e aprendizagem (BROWN, 1992 e COLLINS, 1992), sendo reconhecida também como Pesquisa Baseada em Projeto, é considerada uma opção efetiva para pesquisas que não produzem conhecimentos aplicáveis, como em estudos de Psicologia Educacional conforme defendido por Bell e Sandoval (2004).

Neste trabalho a PBD está alinhada com o conceito da Aprendizagem Baseada em *Design* (EDUSCRATCH, 2011), a qual se demonstra efetivamente adequada à computação criativa, com uma abordagem sobre a concepção de:

- Criar, não apenas utilizar ou reutilizar soluções prontas;
- Personalizar, definindo o foco no que realmente é importante ou relevante;
- Colaborar, compartilhando e tendo acesso a outras pessoas e criações;
- Refletir, por meio da análise, avaliação e conceitualização das práticas criativas de cada indivíduo.

Desta forma, o foco deste estudo não está pontuado no desenvolvimento de um simples artefato pedagógico, mas sim no que é pautado pela PBD, uma análise e implementação de processos, métodos de aprendizagem, currículos e teorias envolvidas no processo de ensino de algoritmos (KELLY, 2004; RAMOS, 2009), buscando o pensamento crítico com o uso das TIC, o que leva à construção do Pensamento Computacional.

1.5. HIPÓTESES

O desenvolvimento das ações deste estudo visa avaliar os processos de aprendizagem, voltado para a área de algoritmos e, a proposição de uma nova metodologia de ensino integrando a utilização de ambientes imersivos e o paradigma de blocos de programação visual em aulas iniciais de programação, buscando desta forma que o indivíduo compreenda os conteúdos e construa o conhecimento sobre algoritmos e programação. Assim a presente pesquisa buscou elementos para validar a seguinte hipótese:

- A estratégia de aprendizagem de algoritmos integrando ambientes imersivos e o paradigma de blocos de programação visual promove melhorias na aprendizagem e construção do conhecimento sobre algoritmos e programação.

1.6. ESTRUTURA DO TEXTO

Para alcançar os objetivos propostos por esta tese, inicialmente partiu-se da construção de um referencial teórico inerentes às áreas de interesse que norteiam o tema de estudo em questão, os quais estão apresentados no Capítulo 2, com a fundamentação teórica, base para investigação.

No Capítulo 3 descreve o planejamento da pesquisa, abordando os aspectos metodológicos envolvidos, incluindo a etapa da revisão bibliográfica, o tipo de pesquisa adotado, os experimentos realizados, etapas e procedimentos.

O Capitulo 4 contém a descrição do planejamento e execução de todas etapas relacionadas ao projeto piloto, assim como a análise dos resultados obtidos com tais experimentos.

No Capítulo 5 é apresentado o experimento final, detalhando todos os casos implementados e resultados obtidos. Também é descrita a avaliação do desempenho acadêmico dos alunos, por meio de uma inferência estatística apurada, além da análise da construção do Pensamento Computacional por parte dos alunos participantes desta etapa do estudo.

O Capítulo 6 contém a descrição das considerações finais e, apresenta um parecer sobre a execução deste projeto de pesquisa e os resultados alcançados.

2. REFERENCIAL TEÓRICO

Este Capítulo apresenta a base teórica para a pesquisa, buscando enaltecer os aspectos, fundamentos e teorias relacionadas à aprendizagem de algoritmos, práticas pedagógicas, o ensino de algoritmos por meio de jogos, novas tecnologias de apoio ao ensino de programação e, por fim, ambientes educacionais ricos em tecnologia, os quais podem ser considerados elementos fundamentais para este estudo.

2.1. PRÁTICAS PEDAGÓGICAS NO ENSINO DE ALGORITMOS

A prática pedagógica possui uma ampla concepção, podendo ser considerada como uma ação social, calcada sobre objetivos, finalidades e conhecimentos, inserida no contexto da prática social, segundo Veiga (1992). Ainda, para Estrela *et al.* (2002), a prática pedagógica deve: estar centrada na análise das situações reais do exercício profissional; orientar-se para o desenvolvimento das competências técnicas, científicas, éticas, pessoais e sociais; atender a maior variedade de contextos possíveis, e; proporcionar momentos distintos de observação, diálogo e trocas. O que desta forma, atende também o pressuposto de Galveias (2008), o qual a descreve como a construção do conhecimento profissional, baseado na interação constante entre a teoria e a prática.

Ao se tratar do ensino na área de tecnologia é importante observar que os sistemas computacionais estão em constante mudança, aperfeiçoamento, o que atinge diretamente o contexto das práticas pedagógicas. Moran (2001) já afirmava que as mudanças são constantes na sociedade, países e instituições. Este contexto pode ser nitidamente identificado no conteúdo desta pesquisa, em que diferentes práticas pedagógicas são adotadas por professores e experimentadas em estudos com o foco na aprendizagem de algoritmos, dentre as quais destacam-se: o uso de realidade virtual (BARBAL & LOPES, 2013), adoção de dispositivos

móveis como ferramenta de apoio (BARCELOS, 2013), prática com o uso de animações, simulações e *web* (BECK *et al.* 2014, HAAJANEN *et al.* 1997), intervenções complexas (BROWN, 1992), ambientes interativos – projeto Alice (CONWAY E PAUSCH, 1997), ensino de programação por demonstração (FERREIRA *et al.* 2010), robótica como instrumento de aprendizagem (FINKE *et al.* 1996), utilização de redes sociais com prática em algoritmos (MIRANDA, 2014), entre outras.

2.1.1. O QUE É ENSINADO EM ALGORITMOS

A organização para o ensino de algoritmos e lógica de programação no ensino superior deve respeitar o Projeto Pedagógico do Curso (PPC), um documento elaborado de forma participativa por docentes, alunos e demais envolvidos no processo de aprendizagem (RANALI & LOMBARDO, 2006), orientando obrigatoriamente as ações educativas voltadas para a construção do conhecimento e formação do estudante. O PPC, segundo o Sistema Nacional de Avaliação da Educação (SINAES, 2014) deve estar devidamente articulado ao Plano Pedagógico Institucional (PPI), ao Plano de Desenvolvimento Institucional (PDI) e, em consonância com Leis e Decretos Nacionais, a fim de promover uma adequada gestão acadêmica, pedagógica e administrativa para os cursos de nível superior. Desta forma o PPC deverá conter um conjunto de orientações, conhecimentos e saberes focados no estímulo à construção das competências dos alunos, tendo como conteúdos a estrutura curricular de cada disciplina, ementas, bibliografias básicas e complementares, descrição de recursos, laboratórios e infraestrutura disponibilizada.

De aspecto amplo, assim como para o ensino de algoritmos e programação, os PPC dos diferentes cursos superiores, devem seguir as orientações das Diretrizes Curriculares Nacionais (DCN, 2014) e, pontualmente no caso deste estudo, as Diretrizes para os cursos de graduação em Computação (DCN-C, 2014), graduação em Engenharia da Computação (DCN-EC, 2014) e graduação Tecnológica (DCN-Tec, 2014). As DCN definem apenas os conteúdos básicos que devem nortear os currículos dos cursos, servindo como documento de orientação, possibilitando um determinado grau de liberdade para que as instituições possam adequar suas estruturas curriculares. Em relação à computação, alguns elementos são considerados comuns para as diferentes áreas, como as competências e habilidades gerais dos egressos, em que são definidas como habilidades comuns: identificar problemas que tenham

solução algorítmica e resolver problemas utilizando ambientes de programação. Em relação aos Projetos Pedagógicos de Cursos, também é comum a definição de conteúdos básicos que exijam o conhecimento de lógica e algoritmos (DCN-C, 2014).

Assim como as DCN, a Sociedade Brasileira de Computação (SBC) disponibiliza um documento de orientação denominado "*Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática*" (SBC, 2014), também denominado como Currículo de Referência (CR99.01) – aprovado em agosto de 2003, a fim de servir como um guia de referência para a implantação de novos currículos para os cursos de graduação na área de computação e informática. Neste guia as matérias estão organizadas em três núcleos: Fundamentos de Computação, Sistemas de Informação e Tecnologia da Computação. As disciplinas na área de programação, incluindo algoritmos, são enquadradas neste último núcleo, tendo como conteúdo básico:

- Desenvolvimento de algoritmos;
- Tipos de dados básicos e estruturados;
- Comandos de uma linguagem de programação;
- Metodologia de desenvolvimento de programas;
- Modularidade e abstração.

As diretrizes deste documento definem que os currículos devem contemplar as matérias de todos os núcleos, porém para cursos que possuem a computação como atividade-fim (Bacharelados em Ciência da Computação e Engenharia da Computação) é recomendado dispor de sessenta créditos, voltados para o núcleo de Tecnologia da Computação e, trinta créditos para os cursos que apresentam a computação como atividade-meio (Bacharelado em Sistemas de Informação).

Com base no entendimento de que as DCN e o guia da SBC são documentos apenas de referência, e que cada instituição pode elaborar de forma independente e autônoma os conteúdos curriculares de seus PPC, para a identificação do que é ministrado nas disciplinas de algoritmos e programação, em nível dos cursos de graduação no País, tornou-se necessário a pesquisa e análise de uma amostra de Projetos Pedagógicos, por meio da qual fosse possível apontar os temas comumente abordados nesta área. Com esta finalidade foram elencados e avaliados, por meio de um processo aleatório, um conjunto de vinte e cinco PPC, referentes a cursos de graduação de Instituições de Ensino Superiores (IES) no Brasil, conforme mostrado no Quadro 03. A amostra contempla cursos de computação dentro das cinco áreas de

formação sugeridas pelas Diretrizes Curriculares Nacionais e pelo Currículo de Referência da SBC, abrangendo cursos de bacharelado em ciência da computação, bacharelado em engenharia de computação, bacharelado em engenharia de *software*, Licenciatura em computação e Bacharelado em Sistemas de Informação (SBC, 2014).

Quadro 3–Instituições de ensino superior e Cursos

	Instituição de Ensino Superior	Curso		Instituição de Ensino Superior	Curso
1	UCB - Universidade Católica de Brasília	Ciência da Computação	14	IFF - Instituto Federal de Educação Ciência e Tecnologia Farroupilha, Campus Santo Augusto	Licenciatura em Computação
2	UFAL - Universidade Federal de Alagoas.	Ciência da Computação	15	IFTO - Instituto Federal de Educação Ciência e Tecnologia do Tocantins	Licenciatura em Computação
3	UFSJ - Universidade Federal de São João Del-Rei	Ciência da Computação	16	UFVJM - Universidade Federal dos Vales do Jequitinhonha e Mucuri	Sistemas de Informação
4	USP - Universidade de São Paulo.	Ciências de Computação	17	IFSE - Instituto Federal de Educação Ciência e Tecnologia de Sergipe	Sistemas de Informação
5	UFSM - Universidade Federal de Santa Maria	Ciência da Computação	18	UFPI - Universidade Federal do Piauí	Sistemas de Informação
6	UNIPAMPA - Universidade Federal do Pampa, Campus Bagé	Engenharia da Computação	19	UFRA - Universidade Federal Rural da Amazônia	Sistemas de Informação
7	UFPA - Universidade Federal do Pará	Engenharia da Computação	20	UFV - Universidade Federal de Viçosa	Sistemas de Informação
8	UFAM - Universidade Federal do Amazonas.	Engenharia da Computação	21	UFC - Universidade Federal do Ceará	Engenharia de Software
9	UFES - Universidade Federal do Espirito Santo	Engenharia de Computação	22	USS - Universidade Severino Sombra	Engenharia de Software
10	UFSC - Universidade Federal de Santa Catarina	Engenharia de Computação	23	UFG - Universidade Federal de Goiás	Engenharia de Software
11	UNICAMP - Universidade Estadual de Campinas	Engenharia de Computação	24	UNIPAMPA - Universidade Federal do Pampa, Campus Alegrete	Engenharia de Software
12	UFG - Universidade Federal de Goiás	Engenharia de Computação	25	UFRN - Universidade Federal do Rio Grande do Norte	Engenharia de Software
13	UFPB - Universidade Federal da Paraíba	Licenciatura em Computação			

Fonte: Próprio autor.

O levantamento da amostra de cursos, apresentada no Quadro 03, está elaborado de forma equânime, com o mesmo número de cursos de cada área, abordando todas as áreas de formação dos bacharelados e licenciatura em computação, distribuídos conforme o Quadro 04, com a apresentação das disciplinas avaliadas e suas respectivas cargas horárias, segundo (UCB, 2014; UFAL, 2014; UFSJ, 2014; USP, 2014; UFSM, 2014; UNIPAMPA-CB, 2014; UFPA, 2014; UFAM, 2014; UFES, 2014; UFSC, 2014; UNICAMP, 2014; UFG, 2014; UFPB, 2014; IFFarroupilha, 2014; IFTO, 2014; UFVJM, 2014; IFSE, 2014; UFPI, 2014; UFRA, 2014; UFV, 2014; UFC, 2014; USS, 2014; UFG-1, 2014; UNIPAMPA-CA, 2014; UFRN, 2014).

Quadro 4- Disciplinas observadas

	IES	Disciplina	Carga Horária		IES	Disciplina	Carga Horária
1	UCB	Algoritmos e Programação	120	12	UFG	Algoritmos e Programação I	60
2	UFAL	Programação I	60	13	UFPB	Introdução a Programação	60
		Laboratório de Programação I	60			Linguagem de Programação I	60
3	UFSJ	Algoritmos e Estrutura da Dados I	72	14	IFFarroupilha	Algoritmos	80
		Laboratório de Algoritmos e Estrutura de Dados I	72			Linguagem de Programação	80
4	USP	Introdução a Computação I	60	15	IFTO	Fundamentos de Lógica de Algoritmos	65
5	UFSM	Lógica e Algoritmos	60	16	UFVJM	Algoritmos e Estrutura de Dados I	75
6	UNIPAMPA-CB	Algoritmos e Programação	60	17	IFSE	Técnicas de Programação	90
7	UFPA	Programação I	60			Estrutura de Dados I	72
8	UFAM	Introdução a Computação	90	18	UFPI	Algoritmos e Programação I	75
		Algoritmos e Estrutura da Dados I	90	19	UFRA	Técnicas de Programação I	68
9	UFES	Programação I	60	20	UFV	Introdução a Lógica	60
		Programação II	60	21	UFC	Fundamentos de Programação	60
10	UFSC	Lógica de Programação	72	22	USS	Algoritmos	60
		Linguagem de Programação	72	23	UFG-1	Introdução a Programação	64
11	UNICAMP	Algoritmos e Programação de Computadores	60	24	UNIPAMPA-CA	Algoritmos e Programação	60
		Desafios de Programação I	60	25	UFRN	Introdução a Técnicas de Programação	90

Fonte: Próprio autor.

Em uma avaliação inicial, dos Quadros 03 e 04, é possível identificar:

- Que os cursos de Licenciatura em Computação distribuem uma maior carga horária média para as disciplinas com conteúdos de algoritmos (115 horas), em segundo lugar estão os bacharelados em Engenharia de Computação, com média horária de 106,28 horas, Sistemas de Informação estão na terceira posição, com média de 88 horas, os cursos de Ciência da Computação estão em quarto lugar, com uma média de 78 horas e, por último os bacharelados em Engenharia de *Software* com 66,80 horas para as mesmas disciplinas.
- O PPC de um curso de Engenharia de Computação apresenta a maior distribuição de carga horária para os conteúdos de algoritmos, com 180 horas, distribuídas em duas disciplinas. Valores muito próximos aos observados para os cursos de Sistemas de Informação e Licenciatura em Computação, com 162 horas e 160 horas, respectivamente e, em ambos também alocados em duas disciplinas;
- Em relação à distribuição dos conteúdos básicos de algoritmos, pelo número de disciplinas apresentadas nos PPC, houve nos cursos de Engenharia de Computação quatro incidências de conteúdos básicos distribuídos em duas disciplinas, o mesmo é identificado nas Licenciaturas em Computação, com duas ocorrências, e nos

cursos de Ciência da Computação e Sistemas de Informação apenas uma ocorrência para cada. Nas Engenharias de *software*, todas as ementas nos PPC apresentam uma relação de uma disciplina para todo conteúdo previsto.

Para a sistematização na análise das ementas/conteúdos dos Projetos Pedagógicos, os temas observados nos diferentes PPC foram agrupados por similaridade, em nove conjuntos diferentes, todos alinhados às resoluções das DCN e ao guia de referência SBC, conforme segue:

1. Resolução de problemas: conteúdos que abordam o fomento da capacidade do aluno de resolver problemas por meio do Pensamento Computacional;
2. Pseudocódigo e fluxograma: disciplinas que contemplem a utilização destas técnicas no estudo da lógica de programação;
3. Documentação e testes: disciplinas que contenham conteúdos voltados para o desenvolvimento da documentação e testes de algoritmos;
4. Conteúdos básicos: ementas que contemplem os conteúdos básicos, necessários para o desenvolvimento de algoritmos e programas simples, como: Conceito de algoritmo, partes do algoritmo, tipos de variáveis, atribuição e operações, entrada e saída, estruturas de condição, estruturas de repetição, vetores, matrizes. Subalgoritmos: procedimentos e funções;
5. Linguagem de Programação: disciplinas que possuam conteúdo voltado para o estudo de diferentes linguagens de programação;
6. Conceitos avançados: ementas que contemplem conteúdos avançados de programação, como: paradigmas de programação, linguagem de máquina, algoritmos de pesquisa e ordenação, algoritmos geométricos, manipulação de arquivos e conteúdos de memória, arquitetura do computador;
7. Ponteiros: conteúdos que abordem o estudo de ponteiros;
8. Recursividade: ementas que contenham o estudo de programas com diferentes níveis de recursividade;
9. História e teoria da computação: disciplinas que abordem o estudo sobre história de sistemas computacionais e teoria da computação.

A avaliação de cada curso é realizada a partir da análise e observação do conteúdo proposto em suas disciplinas de algoritmos, com os temas de cada um dos conjuntos. A igualdade mínima de um conteúdo é considerada suficiente para o seu enquadramento no conjunto. O quadro 05 demonstra o resultado para os vinte e cinco cursos observados.

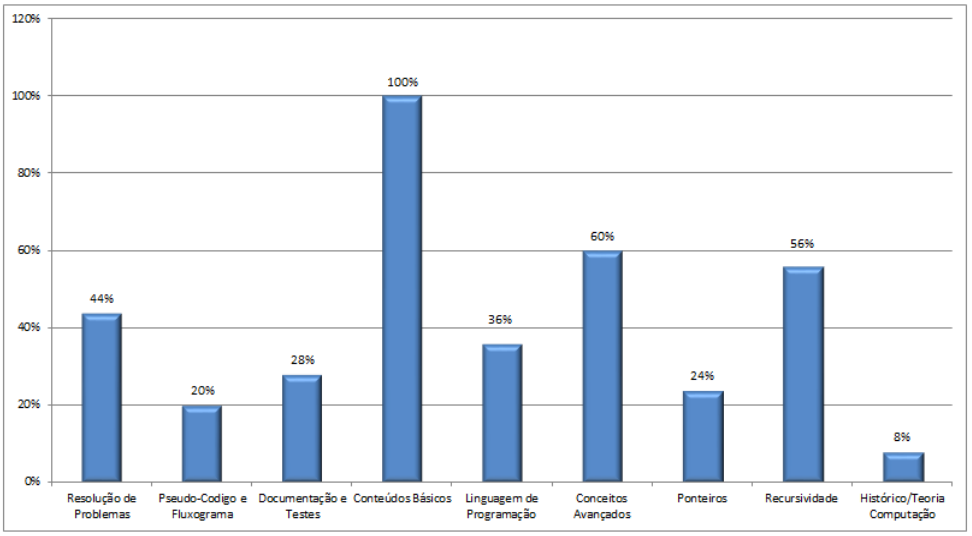
Quadro 5–Avaliação dos conteúdos das disciplinas de algoritmos e programação

		IES	1 Resolução de Problemas	2 Pseudo-Codigo e Fluxograma	3 Documentação e Testes	4 Conteúdos Básicos	5 Linguagem de Programação	6 Conceitos Avançados	7 Ponteiros	8 Recursividade	9 Histórico/ Teoria Computação
Ciência da Computação	1	UCB	1	1		1					
	2	UFAL	1		1	1		1		1	
	3	UFSJ			1	1		1			
	4	USP	1		1	1	1	1	1		1
	5	UFSM	1			1		1	1	1	
Engenharia de Computação	6	UNIPAMPA-CB	1	1		1					
	7	UFPA		1		1		1	1	1	
	8	UFAM	1		1	1		1	1	1	
	9	UFES				1	1			1	
	10	UFSC		1		1	1	1	1		
	11	UNICAMP	1	1	1	1		1		1	
	12	UFG				1				1	
Licenciatura em Computação	13	UFPB				1	1	1		1	1
	14	IFF				1	1			1	
	15	IFTO	1			1		1			
Sistemas de Informação	16	UFVJM	1			1					
	17	IFSE				1	1			1	
	18	UFPI	1		1	1					
	19	UFRA	1		1	1	1	1		1	
	20	UFV				1		1		1	
Engenharia de Software	21	UFC				1		1	1		
	22	USS				1		1			
	23	UFG-1				1	1			1	
	24	UNIPAMPA-CA				1					
	25	UFRN				1	1	1		1	

Fonte: Próprio autor.

O Gráfico 01, apresenta a comparação entre a análise percentual nas observações da relação dos conteúdos de cada PPC com os conjuntos definidos neste estudo. Destaca-se que todas as disciplinas (100%) apresentam os Conteúdos Básicos de algoritmos, (60%) contemplam Conteúdos Avançados, o tema Recursividade está presente em (56%) das ementas e a questão da resolução de problemas é descrito em menos da metade dos conteúdos (44%). De relevância para o ensino desta área e com menor índice apresentado, estão os conteúdos de Pseudocódigo e Fluxograma com apenas (20%) de ementas que apontam para estes assuntos.

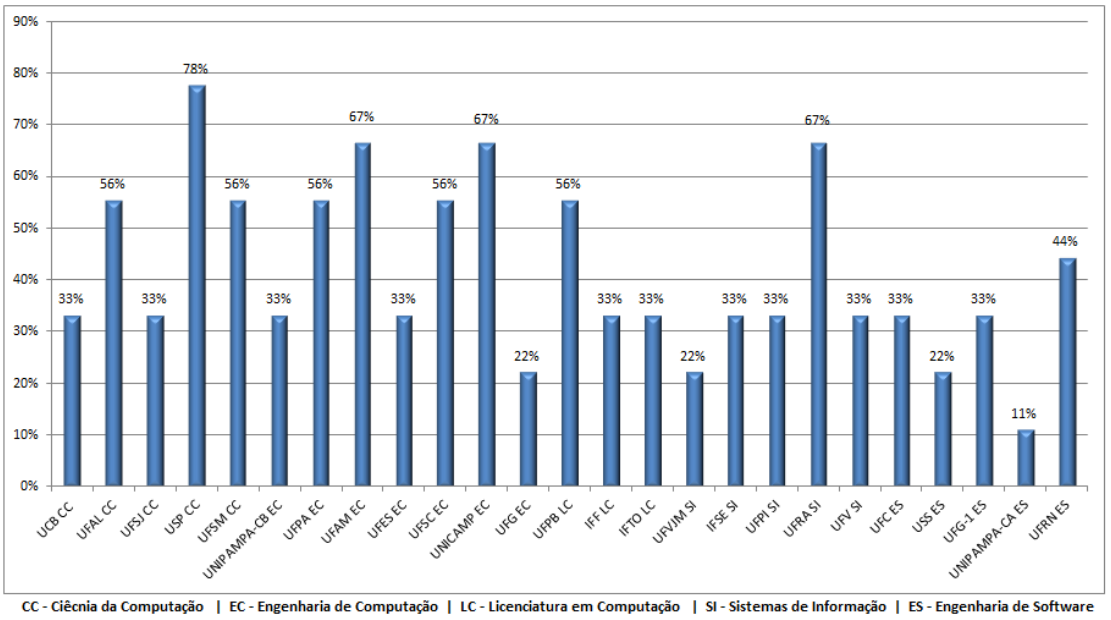
Gráfico 1–Avaliação de ementas dos PPC



Fonte: Próprio autor.

A abrangência dos conteúdos de cada ementa por curso, em relação ao conjunto de temas propostos, é apresentada no Gráfico 02, onde percebe-se que nenhum dos PPC atende à totalidade de conteúdos: o curso de Ciência da Computação da USP (USP, 2014) demonstra o melhor desempenho neste quesito (78%), ou seja, relacionando sete dos nove conjuntos; os cursos de Engenharia de Computação da UFAM e UNICAMP (UFAM, 2014; UNICAMP, 2014) e Sistemas de Informação da UFRA (UFRA, 2014) apresentam 67% de abrangência, distribuídos em seis conjuntos; com 56% (cinco conjuntos) estão classificados os bacharelados em Ciência da Computação da UFAL e UFSM (UFAL, 2014; UFSM, 2014), Engenharia de Computação da UFPA e UFSC (UFPA, 2014; UFSC, 2014) e Licenciatura em Computação da UFPB (UFPB, 2014); com a menor abrangência destaca-se o curso de Engenharia de *Software* da UNIPAMPA (UNIPAMPA-CA, 2014) abrangendo apenas um dos conjuntos (11%).

Gráfico 2–Abrangência em relação à distribuição nos conjuntos de conteúdos propostos



Fonte: Próprio autor.

Como um dos resultados deste levantamento bibliográfico sobre o que é ensinado nas disciplinas de algoritmos, cabe salientar que em todo universo de PPC analisados, nenhuma das ementas apontou de forma declarada a utilização de uma estrutura de programação específica, como a estruturada ou orientada a objetos. Além disso, apenas duas IES apontam

declaradamente para o uso de uma linguagem de programação, neste caso a Linguagem C (UFPB, 2014; UFRA, 2014).

Por fim, pode-se afirmar observando o Quadro 06, que as ementas dos cursos na área de computação, em grande parte contemplam os conjuntos que descrevem:

- Resolução de Problemas
- Conteúdos Básicos
    - Definição de algoritmos;
    - Identificação de valores de entrada e Saída;
    - Comandos, variáveis locais e globais;
    - Operadores lógicos e aritméticos;
    - Estruturas Condicionais;
    - Estruturas de Repetição;
    - Funções auxiliares e Manipulação de *Strings*;
    - Teste de Mesa;
    - Vetores;
    - Matrizes;
    - Funções, variáveis e passagem de parâmetros;
    - Procedimentos;
- Conteúdos Avançados
    - Algoritmos de ordenação;
    - Boas práticas de programação;
    - Manipulação de arquivos
- Recursividade

Quadro 6- Dedução dos principais assuntos abordados em disciplinas de algoritmos

			1	2	3	4	5	6	7	8	9
	IES		Resolução de Problemas	Pseudo-Codigo e Fluxograma	Documentação e Testes	Conteúdos Básicos	Linguagem de Programação	Conceitos Avançados	Ponteiros	Recursividade	Históricol Teoria Computação
Ciência da Computação	1	UCB	1	1		1					
	2	UFAL	1		1	1		1		1	
	3	UFSJ			1	1		1			
	4	USP	1		1	1	1	1	1		1
	5	UFSM	1			1		1	1	1	
Engenharia de Computação	6	UNIPAMPA-CB	1	1		1					
	7	UFPA		1		1		1	1	1	
	8	UFAM	1		1	1		1	1	1	
	9	UFES				1	1			1	
	10	UFSC		1		1	1	1	1		
	11	UNICAMP	1	1	1	1		1		1	
	12	UFG				1				1	
Licenciatura em Computação	13	UFPB				1	1	1		1	1
	14	IFF				1	1			1	
	15	IFTO	1			1		1			
Sistemas de Informação	16	UFVJM	1			1					
	17	IFSE				1	1			1	
	18	UFPI	1		1	1					
	19	UFRA	1		1	1	1	1		1	
	20	UFV				1		1		1	
Engenharia de Software	21	UFC				1		1	1		
	22	USS				1		1			
	23	UFG-1				1	1			1	
	24	UNIPAMPA-CA				1					
	25	UFRN				1	1	1		1	

Fonte: Próprio autor.

Ao analisar estudos relacionados ao ensino de programação nos cursos superiores de computação é possível afirmar que, normalmente, são adotadas práticas pedagógicas instrucionistas, ditas tradicionais, em que o professor apresenta os conceitos da linguagem de programação (estruturada ou orientada a objetos) e os alunos utilizam estas informações para resolver problemas (BORGES, 1998). Em contra partida, Júnior *et al.* (2004) descreve ainda três vertentes de práticas, diferenciadas, que visam à melhoria neste processo de aprendizagem: o uso de ferramentas computacionais, buscando facilitar os processos de ensino e de aprendizagem de programação; a adoção de metodologias, focadas em trabalhos que avaliam estratégias e competências; e por último, ferramentas e métodos, que relacionam atividades suportadas por ferramentas computacionais.

É importante salientar que os demais cursos de graduação, fora da área de computação, que possuem a disciplina de algoritmos em seu currículo, tem como fundamentação para este o fomento à construção do raciocínio lógico e de conhecimentos básicos em informática por parte do aluno, descrição observada nas DCN (2014).

2.1.2. O ENSINO TRADICIONAL DE ALGORITMOS

As práticas tradicionais adotadas para o ensino de algoritmos estão pautadas, na grande maioria das vezes, no atendimento de um conteúdo programático, com uma dinâmica

em sala de aula organizada visando basicamente atender o tempo disponível para a transmissão de determinado conteúdo para o aluno, desconsiderando assim as características individuais de cada estudante (ROCHA *et al.* 2010). Entretanto este é um modelo de ensino comumente presencial, o que permite um alto nível de interação entre os discentes e com o professor em sala de aula, garantindo desta maneira a propagação ampla das informações no ambiente (FERREIRA *et al.* 2013). Embora estes sejam métodos adequados para a apresentação de conceitos, este formato de ensino para Choi e Hannafin (1995) não é efetivo quando busca-se a internalização do conhecimento, relacionados a aplicações práticas. Costelloe (2004) expõe que o aluno, neste contexto, atua de forma passiva sendo um mero receptor de informações, baseadas em um conjunto de regras, restringindo desta forma as capacidades dos discentes.

Por sua vez, para Gondim *et al.* (2009) a apresentação dos conteúdos de algoritmos, tradicionalmente, para alunos iniciantes é muitas vezes uma ação de impacto, visto que o contato inicial com as teorias necessárias para a construção de programas se torna um obstáculo, pois o indivíduo deverá em pouco tempo desenvolver um raciocínio lógico e algorítmico, sendo este um conceito novo e totalmente abstrato, além da necessidade de reconhecer novas tecnologias e aprender uma linguagem de programação com suas nuances e, entendendo a complexidade envolvida na sua estrutura.

Outro fato relevante neste contexto caracteriza-se pelo ensino de programação estar calcado sobre conteúdos comuns das áreas das ciências exatas, visto que estes possuem uma forte relação com a matemática, o que permite a construção do raciocínio lógico e das competências necessárias para a resolução de problemas. Segundo Dos Santos e Costa (2006) e corroborado por outros autores que, apontam a necessidade do entendimento e domínio de habilidades matemáticas como fator de sucesso para a aprendizagem de programação (BORGES e FILHO, 2005).

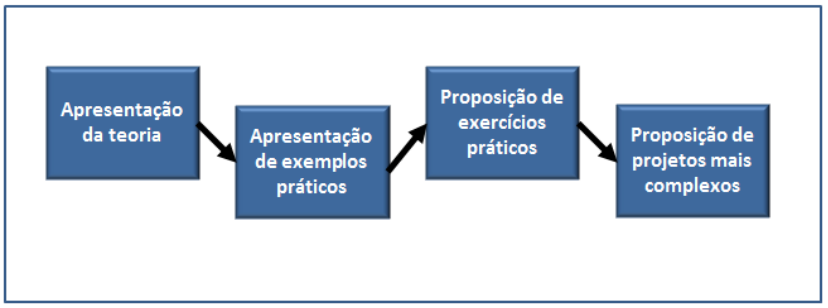
Analisando as práticas recorridas em sala de aula, na abordagem tradicional de ensino, é possível identificar um conjunto de técnicas específicas que visam fomentar a construção do conhecimento sobre programação. Algumas representações como o Portugol, Pseudocódigo, Diagrama de Chapin e Fluxograma (FARIA, 2013) são métodos utilizados com frequência para a elaboração e compreensão de estruturas algorítmicas básicas. Outra técnica baseia-se na utilização do conceito de Máquina de Turing, que segundo Barker-Plummer (2011) permite ao estudante identificar que a solução de um problema pode ser representada por um conjunto de instruções, as quais serão logicamente interpretadas pela máquina, sendo

considerada assim a formalização universal do conceito de algoritmo (FILHO, 2007), com isso, o entendimento sobre algoritmos passa a ser empregado como técnica para resolução de problemas específicos, o que limita o estudante a cenários pouco realistas, reduzindo a aprendizagem experiencial e por sua vez limitando a aplicação dos conceitos estudados (BATISTELLA *et al.* 2012).

A aprendizagem neste modelo inicia pelo uso da linguagem natural, por meio da qual o aluno deve identificar um problema e fragmentá-lo, normalmente de forma manuscrita e sua tradução para um pseudocódigo, para uma linguagem de programação qualquer, respeitando a complexidade, significado e rigidez da mesma (CAMPOS e ASCÊNCIO, 2003).

Borges (1998) descreve em sua pesquisa que no ensino de algoritmos o docente tradicionalmente aborda conceitos de programação estruturada, baseando-se em linguagens como Pascal, C e em casos menos comuns apenas o português estruturado. A Figura 02 apresenta o conjunto de ações comuns na abordagem de conteúdos de algoritmos, sequência também utilizada para o ensino de outras áreas da computação, na qual se inicia pela apresentação de informações teóricas e exemplos práticos (simples), proposição de exercícios que estimulem a conexão com os conteúdos estudados e finalizando com a execução de projetos mais complexos.

Figura 2– Sequência de ações para abordagem de um conteúdo de algoritmos



Fonte: Borges (2000)

Os elementos a serem reconhecidos durante o aprendizado de algoritmos envolvem os conceitos de: variáveis, comandos de entrada/saída, operadores aritméticos, estruturas condicionais e estruturas de repetição. Desta forma o aluno tem contato com todos os domínios de conhecimento necessários para a resolução de um problema e sua posterior implementação, fases que envolvem a sintaxe e escolha da linguagem, construtores de programa, ambiente de desenvolvimento e testes/debug.

Neste contexto, existem muitas ferramentas e metodologias empregadas para o ensino tradicional de programação que integram aulas expositivas a estruturas de laboratórios, caracterizando-se por um formato construtivista na construção do conhecimento. Várias linguagens podem servir de base para isso, contudo Harris (2000) em sua pesquisa, com 20 instituições de ensino, aponta as quatro linguagens de programação mais utilizadas em ambientes acadêmicos (disciplinas introdutórias), obtendo-se os seguintes resultados: linguagem C++ com 60% de utilização, seguida pela linguagem Java (25%), Pascal (10%) e por último C com 5%.

A organização do ensino descrita nesta seção denota uma maior ênfase dos procedimentos didáticos em algoritmos que pontuam na linguagem e não na definição de uma proposta de solução para o problema, o que comprovadamente gera um conjunto dificuldades e abstrações prematuras (SETTI, 2009).

2.1.3. OUTRAS PRÁTICAS PARA O ENSINO DE ALGORITMOS

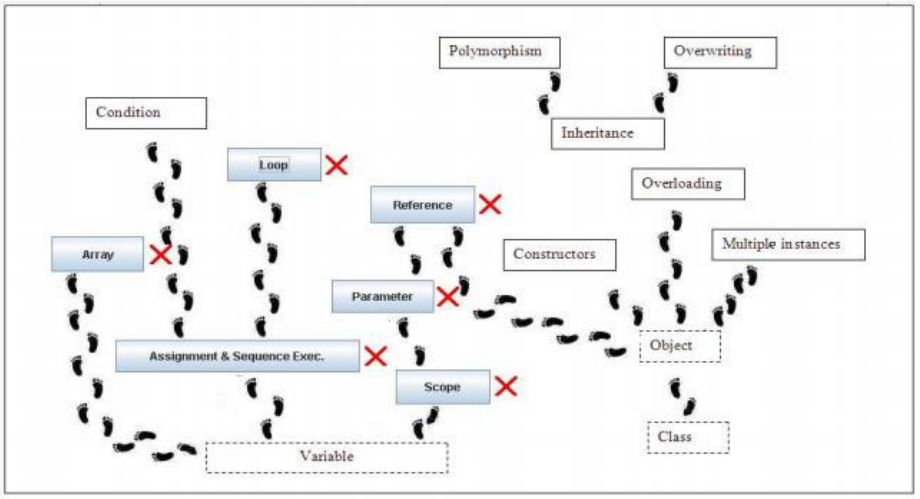
Duas práticas para o apoio ao ensino de programação, para alunos novatos em cursos de graduação, são apresentadas por Suo (2012), o qual descreve um experimento realizado com um universo de 30 alunos em uma disciplina de Programação I. Primeiramente é descrita uma abordagem para a construção do conhecimento sobre algoritmos e programação baseada em um método simplificado, em que um problema é repassado de forma macro ao estudante, em uma conferência geral e, posteriormente este mesmo problema é dividido em considerações menores (mini tarefas), as quais devem ser atendidas individualmente, em uma ordem pré-definida. A solução para cada tarefa respeitará uma sequência lógica de *design*, realização e depuração. As atividades não podem ser sobrepostas, sendo que a mini tarefa posterior, apenas pode ser resolvida, após o término de sua predecessora. A ideia básica desta proposta é não seguir a abordagem tradicional de ensino, a qual busca o desenvolvimento das habilidades de programação por meio da construção do raciocínio lógico, baseada em aspectos cognitivos dos alunos (HUNDHAUSEN, 2003). A segunda estratégia propõe a utilização de uma linguagem de programação gráfica, vislumbrando uma maior compreensão dos alunos sobre as técnicas de programação e, consequentemente um aumento na habilidade de resolução de problemas.

As avaliações destas estratégias, propostas por Suo (2012), ocorreram por meio do *feedback* fornecido pelos alunos, logo após as atividades. O destaque é atribuído ao caráter motivador das práticas, além do fato de instigar os estudantes a aprenderem novas técnicas de programação para a resolução dos problemas. O pesquisador explica ainda, que a experiência dos estudantes em segmentar uma tarefa possibilitou uma redução de erros nos códigos implementados. Em relação à linguagem de programação visual com a linguagem gráfica, os alunos manifestaram sentirem-se mais confortáveis com a utilização das estruturas disponibilizadas para a construção dos programas.

A identificação do baixo desempenho dos estudantes nas disciplinas de programação, nos primeiros anos da universidade, motivou Ma *et al.* (2009), a desenvolver uma pesquisa na Universidade de Strathclyde/Reino Unido, sobre este tema, publicada na *14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, a qual apresenta uma proposta de prática pedagógica para a aprendizagem de programação, baseada na tentativa de alcançar uma maior compreensão, por parte dos alunos iniciantes, de conceitos de programação por meio de uma abordagem de ensino que instiga o conflito cognitivo, a fim de desafiar os alunos, ativamente, estimulando suas ideias pré-existentes que possibilitem aos mesmos reconhecerem seus erros e, buscarem soluções viáveis e melhorias em seus modelos mentais. O intuito é disponibilizar ao aluno uma metodologia para a construção de modelos mentais apropriados, para o reconhecimento de técnicas e soluções de problemas de programação. Os pontos que fomentaram o estudo dos pesquisados estão relacionados à falta de capacidade para resolução de problemas computacionais, percebida por (BARNES *et al.* 1997) como uma das principais causas na deficiência da aprendizagem em programação, além da inadequada capacidade dos alunos na resolução de problemas. Como parte deste estudo, é proposto um ambiente *web*, baseado nos principais conceitos necessários para a construção de *software*, os quais motivam o estudante, por meio do conflito cognitivo, a encontrar seus erros de concepção. A ferramenta de visualização Jeliot (HAAJANEN *et al.* 1997) é adotada para a animação dos fragmentos de códigos construídos, através da qual possibilitou a identificação dos erros conceituais, auxiliando os iniciantes a compreenderem suas deficiências na atividade de programação. Um roteiro a ser seguido é apresentado na Figura 03, onde um conjunto de conceitos deve ser reconhecido pelo aluno, contudo as marcações em vermelho mostram conteúdos em que o aluno apresenta um maior nível de dificuldade, enquanto as marcações em verde demonstram conceitos já dominados. É importante destacar que para cada conceito no roteiro é proposto um conjunto de exercícios e uma questão final, elaborados

com a motivação de provocar um conflito cognitivo e identificar se ocorreu um entendimento correto do problema. As pegadas, ainda na Figura 03, indicam o caminho a ser percorrido.

Figura 3–Conceito de roteiro



Fonte: Ma *et al.* (2009)

Ao final dos experimentos, a observação dos resultados mostrou que muitos programadores iniciantes possuem uma baixa compreensão dos conceitos fundamentais de programação, além de provar que a proposta de integrar um modelo de conflito cognitivo e visualização é capaz de auxiliar o estudante a compreender inúmeros conceitos sobre programação.

Baseado nas considerações de (SADLER, 1998) sobre o apoio ao aluno no processo de compreensão dos conteúdos de uma disciplina de programação, que pode ser disponibilizado por meio do desenvolvimento de competências de auto avaliação, as quais vislumbram minimizar as lacunas entre o desempenho alcançado e o esperado para tais estudantes, Sirotheau *et al.* (2011) propõe, em sua pesquisa, uma prática pedagógica, com o intuito de incorporar ao ambiente virtual de aprendizagem Moodle recursos que possibilitem o *feedback* entre alunos, a fim de aprimorar as habilidades de avaliação destes. A motivação da pesquisa está no fato, comprovado por (RUST *et al.* 2003), da existência de uma alta relação entre a incapacidade dos alunos de compreender os requisitos das tarefas de programação e seu baixo desempenho nos conteúdos de algoritmos. A proposta de *feedback* aliada à avaliação entre pares é defendida neste estudo, por meio da disponibilização para o aluno, do visualizador JavaTool, através do qual o estudante iniciante pode interagir com a linguagem Java, de forma dinâmica, permitindo assim a observação dos benefícios da visualização na construção de programas. O experimento é calcado sobre a integração do JavaTool ao Moodle, além do uso,

também vinculado ao ambiente de aprendizagem, de um avaliador de código, com intuito de verificar a usabilidade e eficácia educacional da estratégia. Os resultados apresentados na pesquisa demonstram que 40% dos alunos utilizaram recursos para agilizar a atividade para obter o *feedback* do sistema; 95% dos alunos apontaram como positivo o uso do avaliador para a resolução de problemas incompletos e 49% manifestaram-se sobre a contribuição do avaliador na identificação de inconsistências nas situações anteriores. Os pesquisadores também declaram benéfico o uso do *feedback* colaborativo como instrumento para inserir o aluno no processo de avaliação das soluções propostas, motivando a auto avaliação e desta forma fortalecendo a internalização dos conteúdos e melhora no seu desempenho (SIROTHEAU *et al.* 2011).

Com o foco nas dificuldades do aluno, inerentes à capacidade de abstração e entendimento da estrutura e sintaxe das linguagens de programação, apontadas por (MOTA *et al.* 2009), a pesquisa desenvolvida por Ribeiro *et al.* (2012), busca avaliar o desempenho dos estudantes ao interagir com o iVProg, uma aplicação oriunda do projeto Alice (CONWAY e PAUSH, 1997), implementado com o intuito de ser uma ferramenta visual, para que o discente de forma interativa, construa soluções de *software*, tirando proveito das características de uma linguagem orientada a estruturas de controle, repetição e condição através da manipulação de blocos. O experimento conduzido pelo pesquisador elencou como amostra, turmas do curso de licenciatura em Matemática, os quais participaram durante dezoito semanas de atividades acadêmicas mistas (presenciais e semipresenciais) sobre programação em linguagem C, sendo divididos em 2 grupos (Curso II e Curso III) e, utilizado como amostra de controle alunos de um mesmo curso, porém ministrado de forma tradicional, denominado Curso I . Para tal, o iVProg foi integrado ao ambiente de aprendizagem Moodle através do *pluggin* iTarefa (RODRIGUES *et al.* 2010). Os resultados observados nesta prática apontaram um aumento no nível de frequência dos alunos nos cursos II e III. O impacto positivo alcançado com a utilização do iVProg ficou evidente devido ao crescimento da média dos alunos, a partir do contato com a aplicação, assim como a melhora identificada na capacidade de resolução de problemas, com a utilização da linguagem C.

O acompanhamento do professor ao aluno, em uma disciplina de algoritmos e programação, muitas vezes fica aquém do esperado pelo discente, a distância e demora no atendimento das necessidades do estudante, muitas vezes pode contribuir para problemas de evasão e reprovação. Esta temática é abordada por Chaves *et al.* (2013), ao propor em sua pesquisa uma ferramenta voltada para a automação de atividades práticas em um curso de

programação, possibilitando, desta forma, o docente acompanhar de forma efetiva o desempenho dos estudantes. A aplicação denominada MOJO (Módulo de Integração com Juízes Online) é uma ferramenta baseada nos sistemas utilizados, geralmente, em desafios de programação, com o intuito de analisar e avaliar códigos fonte (KURNIA, *et al.* 2001). O MOJO pode ser integrado ao Moodle, através do qual é realizado o acompanhamento e avaliação das atividades práticas de programação. Os pesquisadores descrevem que a ferramenta pode auxiliar o professor na tarefa mais complexa de uma disciplina de programação, a elaboração e correção de exercícios práticos. O MOJO como prática pedagógica vislumbra reduzir a sobrecarga de trabalho do docente, bem como agilizar o processo de correção e entrega dos resultados das atividades, motivando e aproximando o discente dos conteúdos propostos, ampliando a qualidade de ensino e, por fim disponibilizando ao professor tempo para atendimento.

2.2. USO DE NOVAS TECNOLOGIAS PARA APOIAR O ENSINO DE ALGORITMOS

Os avanços tecnológicos observados atualmente apontam para a necessidade de constantes mudanças nos processos de ensino e de aprendizagem, visto que as práticas atuais, predominantes no cenário de ensino de algoritmos são basicamente apoiadas por novas tecnologias (ambientes/plataformas), disponibilizando recursos multimídia (animações, interatividade, vídeos) ao estudante, conforme apresentado em diferentes pesquisas. Algumas destas tendências e métodos de ensino são apresentadas no BETT (2013), das quais se destacam:

1. A inserção da programação de *software* em todos os currículos;
2. O conceito da adoção de redes sociais (*social learning*) como forma de complemento da aprendizagem;
3. Estratégias de vídeos, que buscam o aspecto humano existente na explicação de um determinado conteúdo por um indivíduo;
4. Uso em larga escala de tecnologias móveis (*smartphones*), visando tirar proveito da concepção que a grande maioria dos estudantes portam seus celulares em sala de aula e, direcionar o uso destes dispositivos para atividades fins dos momentos acadêmicos.

Algumas destas tendências já são foco de experimentos ou práticas adotadas no ensino de programação, como exemplos da inserção de algoritmos no currículo de cursos de nível médio e superior, destacam-se: Pozzebon *et al.* (2013) com o projeto de ensino de programação para escolas, em nível médio, na região sul catarinense, com intuito de aproximar os alunos da universidade e dos cursos da área de computação; Marques *et al.* (2011), aponta em sua pesquisa a implementação de uma oficina de introdução à programação, para o ensino médio, com o uso da linguagem Python, tendo como fatores positivos relatados, o grande interesse dos alunos, o conceito oito apresentado por estes estudantes em relação à oficina e a motivação em seguir nas áreas de computação, mencionado por 56% dos participantes do experimento.

No ensino superior: Gonzáles e Tavares (2008) descrevem uma experiência, realizada durante três anos, sobre a aprendizagem de programação nos cursos de Engenharia de Petróleo, Engenharia Metalúrgica e Materiais e Engenharia Civil, a qual desenvolveu-se com base em uma metodologia organizada em três fases (resolução de problemas, formalização e construção algorítmica), utilizando as linguagens C e Pascal como soluções de desenvolvimento, tendo como resultado uma melhoria de 30% nos índices de aprovação. Identificando a importância da disciplina de algoritmos, Tavares *et al.* (2012) apresenta em sua pesquisa uma reflexão sobre a importância desta disciplina para o curso de Engenharia Elétrica, sua observação sobre a relação dos conteúdos de algoritmos com os Trabalhos de Conclusão de Curso (TCC), apontaram para um índice 100% de relação direta dos TCC com atividades de programação, ainda para esta amostra, as linguagens C e C++ foram elencadas como soluções de preferência para a efetivação de código.

Práticas pedagógicas de algoritmos voltadas para o *Social Learning* são descritas por Miranda (2014), em um estudo sobre a contribuição do uso do Facebook no processo de aprendizagem, de alunos de um curso de Sistemas de Informação. O pesquisador apoia-se no princípio de que as redes sociais podem auxiliar no armazenamento e difusão do conhecimento entre pessoas, permitir e instigar a conexão entre indivíduos e fomentar parcerias e colaborações mútuas, ideias corroboradas por Fuks (2011). Participaram da pesquisa 80 alunos iniciantes, os quais interagiram com um conjunto de recursos e atividades, organizadas na rede social a fim de fomentar o processo de aprendizagem, para isso construiu-se um grupo no facebook, por meio do qual o professor instigou a comunicação entre todos participantes. Denotou-se com esta prática um maior número de questionamentos por parte dos discentes, as quais culminaram em soluções colaborativas, visto que os colegas se

auxiliaram na solução dos problemas, um aspecto positivo, pois a comunicação no ambiente da disciplina tornou-se fluente.

A perceptível melhoria na adaptação aos conteúdos de algoritmos integrados à tecnologia de redes sociais é explicada pelas mesmas estarem diretamente ligadas aos seus interesses e necessidades pessoais. Ainda em relação ao uso de redes sociais, Camargo *et al.* (2013), apresenta a proposta e implementação de uma aplicação que permite a integração dos objetos de aprendizagem, disponibilizados pelo IGUAL - *Innovation for Equality in Latin American Universities* ao Facebook. O IGUAL é um projeto fomentado pela comunidade europeia que objetiva propor soluções, baseadas no uso de tecnologias de ensino inovadoras, para o apoio à aprendizagem de estudantes com dificuldades no entendimento de algoritmos, que frequentem universidades latino-americanas (IGUAL, 2014), buscando como resultado a disponibilização de forma centralizada, via *web*, de objetos de aprendizagem, segundo Ochoa *et al.* (2012). Neste contexto, Camargo *et al.* (2013), busca explorar um ambiente social para oferecer meios de interação do aluno com os assuntos relativos ao tema programação, através de discussões, comentários e construção de recursos. Para este fim é descrito o IGUALS, uma aplicação focada na integração das atividades e objetos de aprendizagem do IGUAL à rede social facebook, possibilitando ao estudante, por meio das opções “curtir”, “ver comentários”, “comentar”, “objeto” e “compartilhar”, divulgar suas ações de estudo, assim como aprender com as resoluções de seus colegas.

Como referência à terceira tendência de tecnologia voltada para o ensino (BETT, 2013), aplicada à área de algoritmos, cita-se o experimento sobre a utilização de Vídeo-Monitoria, em uma disciplina de programação, proposto por Kamei *et al.* (2010), o qual foi desenvolvido com o princípio de apoiar o aluno de programação por meio de uma comunicação rápida, eficiente e rica em conteúdo (CORREIA e CHAMBEL, 2004). O pesquisador propôs a construção de vídeo-aulas para serem utilizadas como práticas pedagógicas complementares às atividades de monitoria da disciplina. O experimento realizado contou com a integração de diferentes ações: aulas presenciais, identificação dos questionamentos comuns dos alunos, seleção de exemplos que atendessem tais questionamentos, gravação das vídeo-aulas, aplicação e observação dos resultados alcançados. Os índices de satisfação dos estudantes (100%), reconhecimento da importância das vídeo-aulas (100%) e o auxílio na compreensão dos conteúdos de matéria (100%), são conclusões positivas apontadas por este estudo.

Por fim, em relação ao uso de tecnologias móveis como ferramenta de ensino, vários trabalhos podem ser encontrados na literatura científica, dentre estes a pesquisa sobre o uso de *mobile learning* no ensino de algoritmos, apresentada por Barcelos *et al.* (2009), faz uma análise sobre as dificuldades encontradas por alunos nesta área do conhecimento e, a partir disto, realiza uma investigação sobre soluções para dispositivos móveis como apoio ao ensino. O estudo firmou-se em diferentes premissas: a disseminação do uso destes dispositivos pelos alunos; novos sistemas educacionais construídos especificamente para equipamentos portáteis; a possibilidade de integração de sistemas já existentes com tecnologias móveis; utilização flexível e otimizada do tempo por parte do aluno, ao interagir com este tipo de dispositivo, e; a ampliação do uso educacional destes recursos. O experimento focou na portabilidade de materiais em vídeos para celulares e a disponibilização como objetos de apoio à aprendizagem dos alunos. Alguns aspectos técnicos foram amplamente discutidos para a viabilização do projeto, como: conversão e compatibilidade dos arquivos de vídeo, diversidade de marcas e modelos dos dispositivos, capacidades e taxas de transferência, qualidade de áudio e vídeo, manuseio e interação com os equipamentos para construção de código. A capacidade em dispor diferentes níveis de envolvimento entre os alunos e entre os alunos e os professores, além da possibilidade de socialização das anotações realizadas pelo estudante, em seu celular, com determinados grupos, são descritos como fatores positivos na utilização destas tecnologias, para Barcelos *et al.* (2009) e corroborado por Fischer e Baird (2007).

O projeto baseado na utilização da *App Inventor for Android - AIA* (GOMES e MELO, 2013) é outro estudo sobre tecnologias móveis para o ensino de algoritmos e programação. Este descreve a uso de uma solução com blocos de programação visual, para dispositivos sobre a plataforma Android, com o intuito de promover situações mais atrativas para o processo de aprendizagem de algoritmos, em cursos iniciais de lógica de programação. A pesquisa busca alcançar a proposta de Cristovão (2008), que aponta a necessidade do estudante em dominar uma determinada linguagem de programação, a fim de produzir respostas algorítmicas satisfatórias para diferentes problemas. A escolha do *App Inventor for Android* é justificada por sua facilidade de entendimento e usabilidade, quando comparada a linguagens tradicionais de desenvolvimento, além do fato de sua concepção ser baseada nas linguagens Scratch e Logo, estimulando desta forma o processo criativo para a construção de programas para dispositivos móveis (ABELSON, 2010).

O universo de participantes da experiência foi composto de alunos do primeiro e segundo ano do ensino médio, de uma escola pública, além de estudantes de uma IES, os quais foram estimulados a desenvolver aplicações com o uso do AIA, para equipamentos portáteis, contemplando um conjunto de conhecimentos básicos de programação. Observações relevantes, descritas pelos pesquisadores, relatam o manifesto da grande maioria dos participantes (75%) em prosseguir seus estudos na área de informática em nível superior. Outros aspectos positivos também apresentados: a utilização do AIA como ambiente de aprendizagem significativa; a motivação do estudante, pela possibilidade de construir aplicações para seus próprios dispositivos (celulares), integrando características comuns nestes equipamentos (mobilidade, geolocalização, entre outros); a detecção intuitiva de erros; o *feedback* imediato, e; a exploração do caráter construcionista das atividades realizadas.

Além das tendências apontadas na BETT (2013), segue abaixo, um conjunto de estratégias utilizadas no ensino de algoritmos, as quais empregam tecnologias como instrumento de apoio:

- Educação a distância de programação para alunos iniciantes (FERNANDIN & STEPHANI, 2005), (NASCIMENTO *et al.* 2011);
- Construção de algoritmos através da mediação digital (BERTCH *et al.* 2005), (ROSA & RAPKIEWICZ, 2007);
- *Software* educacionais para o ensino de algoritmos por meio de grafos (SANTOS & COSTA, 2005), (SANTOS *et al.* 2008);
- Uso de brinquedos no ensino de algoritmos (CORREIA & DOMINGUES, 2006);
- Aprendizagem de lógica de programação utilizando programação por demonstração (FERREIRA *et al.* 2010), (SANTOS & COSTA, 2006);
- Experiência de ensino de algoritmos utilizando ambientes visuais de programação 3D (GONDIM *et al.* 2009), (BARCELOS, 2011);
- Ensino de programação através do Portugol (NOBRE, 2002), (MANSO *et al.* 2009);
- Análise dos estados afetivos dos alunos, em relação à frustração, na disciplina de algoritmos (IEPSEN *et al.* 2011; 2013);
- Utilização de *software* gráfico para apresentação de algoritmos e estrutura de dados para alunos iniciantes (SANTOS & COSTA, 2006), (GARCIA *et al.* 1997);
- Sistemas tutores inteligentes no ensino de programação (BOTELHO, 2010), (GIRAFFA & VICARI, 1999);

- Uso de ambientes virtuais colaborativos para o auxílio no processo de aprendizagem em algoritmos (REBELO *et al.* 2005), (REATEGUI *et al.* 2006);
- Propostas inovadoras no ensino de algoritmos utilizando linguagens visuais (Scratch) e ferramentas como Arduino (MÉLO *et al.* 2011), (NETO, 2013);
- Aprendizagem de algoritmos por meio da estratégia ascendente de resolução de problemas (FALKEMBACH, 2003), (DIM & ROCHA, 2011);

2.2.1. ENSINAR E APRENDER EM UM AMBIENTE RICO EM TECNOLOGIA

Ao analisar o contexto educacional atual é possível identificar, teoricamente e também na prática, um conjunto de recursos tecnológicos que podem ser utilizados como ferramentas para o apoio aos processos e ensino e de aprendizagem. Contudo, para alcançar a efetividade na ação de ensino é necessária a adoção de alguns conceitos importantes, pautados sobre estratégias específicas, dentre elas: teorias de cognição, construção cooperativa de saberes e aspectos multimídia (TAROUCO *et al.* 2009).

Sabe-se que ao utilizar recursos tecnológicos para construção de materiais didáticos, diversos elementos devem ser considerados, dentre os quais se destaca a Teoria da Carga Cognitiva (TCC), descrita por Sweller *et al.* (1998). Para o entendimento desta teoria é primordial o reconhecimento sobre o conceito da cognição humana (NUNES & GIRAFFA, 2003), que descreve a capacidade de um indivíduo em processar informações por meio de sua estrutura cognitiva, definida em: memória sensorial, memória de longa duração e memória de curta duração, segundo Tarouco & Santos (2007). Por sua vez a concepção de materiais educacionais deve ser estruturada através da composição sucessiva de objetos, disponibilizados de acordo com o seu nível de dificuldade, que segundo Wiley (2000), é a base para a chamada teoria da complexidade, a qual define que a aprendizagem deve ser organizada de forma hierárquica e crescente em relação a sua complexidade. Todavia, este fator pode ter um efeito negativo no processo de aprendizagem, levando a uma alta demanda no processo de interpretação dos conteúdos e, muitas vezes à distração do estudante, o que é descrito como Sobrecarga Cognitiva (TAROUCO, 2006).

Com o intuito de reduzir esta demanda, Lindermann (1983) e (TAROUCO, 2006) apontam alguns cuidados na implementação de materiais educacionais, a fim de minimizar os

impactos das diferentes informações contidas em um objeto de aprendizagem: simplicidade nos textos utilizados, padrões de cores atraentes e não cansativas formatações de textos e imagens de forma agradável, agrupamento das informações por similaridade e o uso de imagens e gráficos, para um modo visual com o foco na interpretação dos dados. Alinhado a estas considerações, tem-se o "número mágico de sete mais ou menos 2 (7+-2)", apresentado por Miller (1956), que pauta sobre a incapacidade do sistema cognitivo humano de assimilar mais que nove elementos de informação por vez. Miller (1956) defendeu que um indivíduo normal tem a capacidade de absorver, de forma satisfatória, de cinco a nove elementos simultaneamente, sendo esta uma das regras básicas que levaram aos estudos sobre a TCC. Para que um determinado material didático de estudo não gere uma sobrecarga cognitiva, Mayer (2001) e Sweller (2003) propõem em suas pesquisas um conjunto de princípios a serem seguidos: Princípio de Representação Múltipla, o qual descreve que estudantes apresentam um melhor entendimento ao combinar palavras e imagens; Princípio de Proximidade Espacial apontando para que o texto e imagens relacionadas apresentem uma proximidade espacial; Princípio da Não Divisão ou da Proximidade Temporal sugere que o texto e imagens sejam apresentados simultaneamente; Princípio das Diferenças individuais aponta para maior capacidade de determinados indivíduos no entendimento de certos conteúdos; Princípio da Coerência defende que o material didático deve ser simples e objetivo, apenas com informações pertinentes ao tema tratado.

A aprendizagem cooperativa é outra ação de extrema relevância na área do ensino, pois impele um grupo de indivíduos com objetivos em comum a desenvolverem soluções em conjunto. Ellis & Whalen (1990) descrevem que neste tipo de aprendizagem, a cooperação vislumbra de forma equânime a distribuição de ações, respeitando a interdependência entre os elementos, em busca de uma solução comum e válida para todos. A tecnologia pode ser considerada como um fator positivo na aprendizagem cooperativa, porém o sucesso deste tipo de prática repousa sobre uma ação pedagógica calcada sobre a colaboração, descobertas e cooperação, segundo Johnson *et al.* (1994). Em suma este modelo de aprendizagem pode ser definido como uma estrutura de estratégias de ensino, o qual tem como suporte a interação entre estudantes, sendo a característica primordial das tarefas de aprendizagem (REBELO *et al.* 2005). Ao integrar sistemas computacionais à prática cooperativa tem-se duas novas áreas *Computer Supported Collaborative Learning* (CSCL) e *Computer Supported Cooperative Work* (CSCW). A primeira, também denominada Aprendizagem Colaborativa Suportada por Computador, agrega esta tecnologia a interações entre estudantes em atividades de ensino

com objetivos em comum. No Trabalho Cooperativo Suportado por Computador o intuito é propor soluções para permitir o trabalho colaborativo, utilizando o computador como recurso, e extrair o máximo desta tecnologia (GOUVEIA, 2000). Vários aspectos podem ser considerados como positivos na aprendizagem cooperativa com o uso do computador, pois a computação permite um alto grau de interação entre os indivíduos desde o uso de diferentes aplicações de comunicação em rede, até o contato por meio de mundos virtuais, o que segundo Redondo *et al.* (2001), estimula o compartilhamento de informações entre os estudantes, promovendo o comprometimento destes, na realização de atividades cognitivas e metacognitivas em um ambiente educacional, sem restrições de tempo e espaço (MACEDO, 1999).

Ainda sobre a utilização de recursos e métodos para o apoio à aprendizagem, a multimídia pode ser adotada como uma estratégia de ensino válida, pois é apontada por diferentes autores, dentre eles Mayer & Moreno (2002), como uma tecnologia com alto nível de interatividade e traços motivacionais, sendo amplamente reconhecida no favorecimento do processo de aprendizagem do aluno. No contexto pedagógico a multimídia é caracterizada pela disponibilização de um conjunto de elementos: imagens, sons, textos, simulações e gráficos, os quais devem ser organizados adequadamente com a finalidade de proporcionar um efeito positivo na capacidade de entendimento do aluno (VALENTE, 1997) e (KAMPFF & DIAS, 2003). Para Mayer (2001), este efeito é reconhecido a partir da observação na melhoria da capacidade de entendimento dos estudantes, sobre diferentes áreas, quando utilizado materiais educacionais multimídia. A base para a construção destes deve seguir a TCC de Sweller (1998), pois o processo mental necessário para a construção do conhecimento através de telas, imagens, áudios e outras mídias, repousa sobre a capacidade cognitiva do aprendiz em extrair os conteúdos necessários (TAROUCO *et al.* 2009).

A construção de materiais educacionais para a área de algoritmos, com a integração dos aspectos tecnológicos providos pelo computador e fundamentados nos preceitos descritos até este momento, pode produzir resultados positivos no processo de aprendizagem, contudo alguns parâmetros devem ser observados nesta prática: concepção e adequação dos conteúdos segundo o público alvo, qualidade do material proposto, características motivacionais, modelo de *feedback*, reusabilidade, padrões de navegação e *layout* visual (MUSSOI *et al.* 2010). A adoção destes parâmetros não garante a qualidade e eficácia do objeto educacional, visto que outros aspectos como a aprendizagem significativa (AUSUBEL *et al.* 1978), ciclo de aprendizagem experiencial (KOLB, 1997) a taxonomia de Bloom (BLOOM, 1986),

(FERRAZ & BELHOT, 2010), o estilo de aprendizagem do aluno (FELDER & BRENT, 2005), a adoção de modos verbais e não verbais (FELDER & SILVERMAN, 1988) também devem ser observados.

Desta forma inferir a qualidade de um objeto de aprendizagem, a partir de diferentes óticas, é uma necessidade. Sobre este viés, Müller *et al.* (2013) propõe uma metodologia para a mensuração e classificação de OA, de acordo com os princípios de multimodalidade agregados por estes, sendo a técnica pautada sobre sete classes: (1) multimodalidade dispersiva, (2) não multimídia, (3) texto sonorizado, (4) texto sonorizado ilustrado, (5) redundância verbal, (6) multimídia monotônico e (7) multimídia efetiva. Outro método de classificação de objetos de aprendizagem é proposto por Battistella *et al.* (2009), que defende a organização dos OA considerando o aspecto educacional (construção do conhecimento) e as tecnologias utilizadas. As etapas para classificação com este método, preveem a divisão inicial dos objetos, em relação às tecnologias, em interativos (avaliativos, explorativos e colaborativos) e não interativos (texto e multimídia). A área educacional é categorizada de acordo com as ferramentas de autoria adotadas, considerando assim, as características de aprendizagem (VALENTE, 1999) por meio de memorização ou esquemas mentais.

A busca por uma abordagem efetiva no ensino de algoritmos aponta também para a adoção de teorias voltadas a construção do raciocínio lógico, as quais podem estar pautadas sobre uma aprendizagem significativa, proposta por Ausubel *et al.* (1978), a qual explora conceitos importantes como a construção de materiais didáticos significativos, que buscam de forma simples uma conexão entre os conteúdos apresentados com os elementos preexistentes na estrutura cognitiva do aluno, vislumbrando assim mudanças efetivas destas estruturas com base nas novas informações assimiladas. Esta teoria se destaca pelo foco no estudante, buscando que o aluno enriqueça suas bases cognitivas, através da identificação e ligação de conceitos maiores (subsunçores), já reconhecidos, com novos conceitos menores, ou seja, as informações recebidas recentemente se ligam à base de conhecimento do aluno, construindo assim um aprendizado significativo. Contudo para se alcançar este tipo de aprendizagem é necessário o engajamento do estudante, motivação e que o mesmo possua os subsunçores (conhecimentos prévios) necessários para a assimilação dos novos conteúdos, segundo Moreira (2006).

Uma das maneiras de alcançar a aprendizagem significativa, parte da utilização de um ciclo de aprendizagem experiencial, apresentado por Kolb (1984), o qual descreve o processo de concepção da aprendizagem humana em quatro etapas consecutivas, em que: as Vivências

Concretas estão relacionadas ao conhecimento prévio do aluno; as Observações e Reflexões são estimuladas a partir da apresentação de novos conceitos; Conceitos Abstratos e Generalizações encorajam os estudantes a relacionarem os novos conceitos a suas bases de conhecimento prévio e a identificação da aplicação dos mesmos em diferentes problematizações; os Testes, por último, fornece a experimentação ativa, permitindo o contato com novas situações e resultados, que o remetem novamente ao início do ciclo, a fim de ter acesso a novas vivências concretas e assim consecutivamente.

Por fim destaca-se que integração de *software* de autoria e métodos educacionais, utilizados com o intuito de apoiar o processo de ensino de programação, os quais seguem uma visão pedagógica, fortemente embasada nos conceitos e teorias de aprendizagem, procurando alcançar um alto nível de sinergia com as tecnologias atuais, adaptados à educação e estando alinhados aos conceitos de construtivismo de Piaget e Papert segundo Franco *et al.* (2006) e Zuffo (2001), independente da forma ou procedimento de classificação adotado.

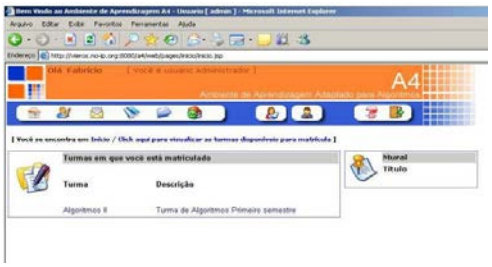
2.2.2. SOLUÇÕES APLICADAS AO ENSINO DE ALGORITMOS

De forma similar à evolução dos sistemas computacionais, a área de ensino e aprendizagem de algoritmos vem se adaptando de forma natural às novas tecnologias, a partir da utilização de métodos ou práticas que exploram, de diferentes formas, as soluções disponíveis. Para o uso efetivo dos elementos computacionais atuais é necessário o estudo e desenvolvimento de práticas pedagógicas que integrem os conceitos básicos de programação aos novos padrões de sistemas e recursos. Dentre estas novas práticas, destacam-se a utilização da Educação a Distância (EaD), ambientes virtuais de aprendizagem (AVA), laboratórios e mundos virtuais, dispositivos móveis, robótica, redes sociais, realidade aumentada, entre outros. Neste contexto, serão descritas algumas soluções para o ensino de algoritmos, utilizando diferentes recursos e métodos, publicados com intuito de aprimorar o processo de aprendizagem em programação.

Um estudo sobre as dificuldades no ensino de algoritmos culminou na proposta de FALKEMBACH (2003), denominado A4 Ambiente de Aprendizagem Adaptado para Algoritmos, uma ferramenta que integra inovações tecnológicas a ações pedagógicas, construído sobre um sistema de hipermídia interativo, utilizando uma estratégia ascendente de resolução de problemas, o qual tem por intuito a efetividade no processo de aprendizagem e

um ensino adaptado ao aluno (DE ARAÚJO *et al.* 2006). O termo adaptado se enquadra nas características desta ferramenta, pelo fato de propiciar a exploração dos conteúdos e instigar os alunos a interagir com cada módulo do sistema. Segundo De Araújo *et al.* (2006), para a construção do A4 utilizou-se o MySql (Base de Dados), *Java Server Pages* (JSP) e *Java Script* (JS) como Linguagem de Programação. O uso destas tecnologias permite ao A4 armazenar informações referentes às ações e preferências do estudante e se adaptar às necessidades destes.

Figura 4– (a) Tela de acesso ao ambiente, (b) Área de gerenciamento do professor



(a)

(b)

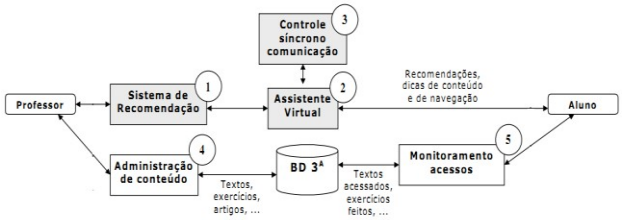
Fonte: De Araújo *et al.* (2006)

A estrutura de utilização do A4 respeita um roteiro guiado, que permite ao aluno transpor ações que dão margem a todos os conteúdos da disciplina, por meio de uma interface simples e atrativa, conforme mostrado na Figura 04(a), dispondo três modos de acesso: modo algoritmo, modo programa fonte e modo fluxograma. Este pode ser considerado uma aplicação dinâmica, o que é identificado por meio da Figura 04(a), a qual demonstra a área de gerenciamento do sistema, pelo professor. Nesta seção da aplicação o docente pode gerenciar: o Sistema de *login*, Fóruns de Discussão, Salas de discussão, Portfólio, Mural e Materiais de Apoio. Para (DE ARAÚJO *et al.* 2006) este ambiente possui amplas características de um objeto de aprendizagem, a disposição de docentes e discentes para o entendimento e resolução de problemas por meio de algoritmos.

Com o foco no suporte à comunicação e cooperação, a ferramenta 3A (Ambiente de Aprendizagem de Algoritmos), implementada por Reategui & Notare (2004), por meio de um conjunto de exercícios, busca o entendimento sobre os conteúdos básicos de programação. O sistema apresenta um conjunto de recursos pertinentes, e reconhecidos na literatura, a serem citados: um sistema de recomendação (REATEGUI *et al.* 2004), um assistente virtual propiciando uma interface personificada por meio de um tutor (CRAIG *et al.* 2002), um

módulo de sincronização das atividades dos alunos com o professor (OEIRAS *et al.* 2002), a capacidade de acesso cooperativo às respostas dos alunos, um módulo de administração de conteúdo e, por fim, um módulo para captura e armazenamento em banco de dados de todas as atividades dos alunos no ambiente. Este último tem a finalidade de prover informações válidas, para o sistema de recomendação. A tela de acesso ao 3A, assim como a estrutura deste sistema são apresentados nas Figuras 05 (a) e (b) respectivamente.

Figura 5–(a) Estrutura do ambiente, (b) Tela de acesso ao ambiente 3A



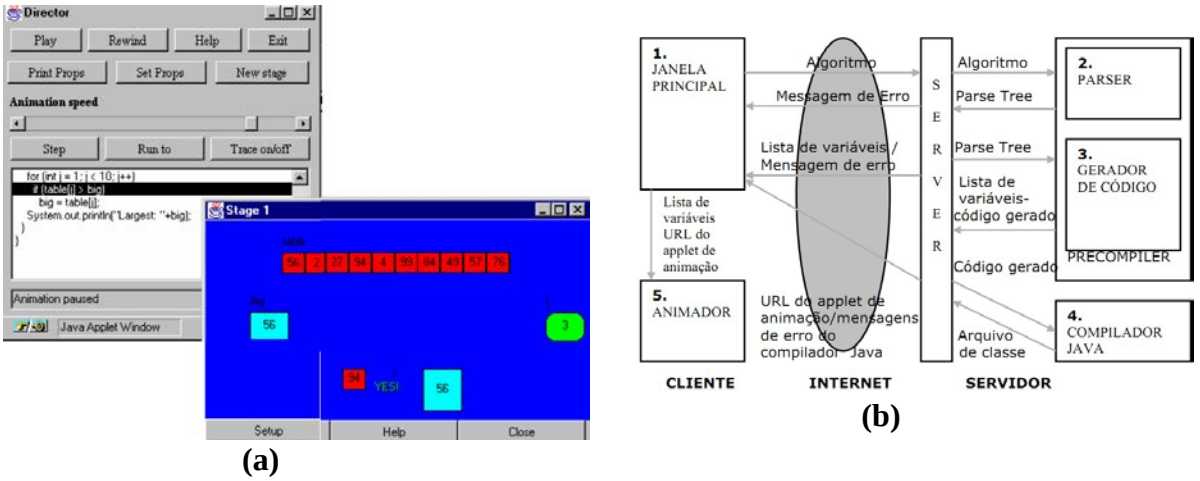
(a)

(b)

Fonte: Reategui & Notare (2004).

Dentre as tecnologias comumente utilizadas para a implementação de ferramentas de apoio, a Internet destaca-se como um meio para o intercâmbio de informações de extrema eficiência, rompendo os limites da sala de aula tradicional. O suporte à multimídia oferecido pela rede possibilita a adoção de recursos de visualização atrativos e dinâmicos, permitindo aos alunos e docentes explorar, com profundidade, diferentes conteúdos. A partir destas possibilidades, (TORI & MAINENTE, 2011) propõem a utilização dos conceitos de: visualização e ilustração de algoritmos, utilizando o sistema Jeliot (HAAJANEN *et al.* 1997), uma ferramenta desenvolvida com tecnologia Java cliente/servidor, para a animação de códigos, com tipos primitivos de dados, matrizes, pilhas e filas, conforme é mostrado na Figura 06 (a).

Figura 6-(a) Código desenvolvido pelo estudante e sua visualização gráfica, (b) Arquitetura das fases do Jeliot.



(a) **(b)**

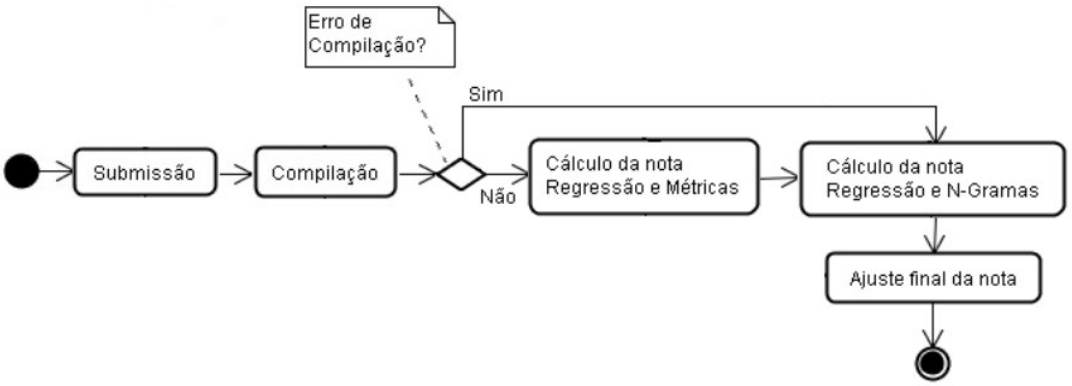
Fonte: Tori & Mainente (2004).

A estrutura do Jeliot, Figura 06 (b), mostra os passos da aplicação para a compilação de um código criado pelo aluno, segmentado em passos bem definidos: leitura do algoritmo, pré-compilação, geração do código e implementação das animações relacionadas ao código fonte. Esta última etapa, fornece um nível de abstração, com a apresentação de imagens, as quais possibilitam ao estudante visualizar o resultado do programa aplicado. A facilidade de conectar o ambiente de desenvolvimento com sítios *web*, contendo materiais e explicações clássicas sobre as práticas desenvolvidas, pode ser considerada um aspecto importante nesta solução.

A avaliação automática pode ser considerada como uma característica relevante para um processo de ensino e aprendizagem efetivos. Os sistemas computacionais atuais permitem a utilização de recursos para prover um *feedback* automático ao estudante, disponibilizando desta maneira um método para a identificação imediata de erros e a construção de um raciocínio correto para os problemas propostos. Contudo a avaliação de um determinado código é uma atividade complexa, visto que, diferentes soluções podem ser elaboradas para um mesmo problema. Neste contexto, (MOREIRA & FAVERO, 2009) descrevem uma solução para avaliação de programas, utilizando como técnica de predição a regressão linear múltipla, defendida por Hearst (2000) e Lino *et al.* (2007). Como indicadores de medidas, utilizou-se a abordagem de n-gramas (SUKKARIEH *et al.* 2003) em comparação a métricas da área de engenharia de *software*. A validação da pesquisa ocorreu com a integração, por meio de um *WebService* do método de avaliação proposto a uma sala de aula no AVA

Moodle, sendo o módulo de integração e as fases para a resolução de códigos, observados no fluxograma da Figura 07. Como resultado, tem-se a avaliação positiva da proposta, comprovando o nível aceitável de precisão na correção da sintaxe dos algoritmos, em linguagem Java, desenvolvidos pelos alunos.

Figura 7- Módulo de resolução e *feedback* automático



Fonte: Moreira & Favero (2009).

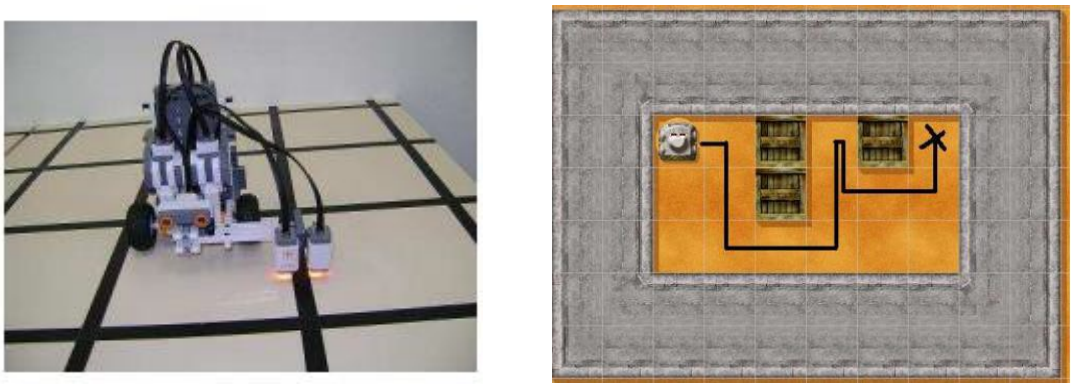
Na busca de novas tendências para o ensino de algoritmos e programação, a tecnologia de robótica educacional, apresentada por Maia *et al.* (2008) e Morelato & Borges (2008), salienta-se como uma possibilidade viável, devido às suas características relacionadas à construção e programação de robôs, geralmente estimulando aspectos intuitivos na resolução de problemas, integrando *hardware* e *software*. Nesta área (VAHLDICK *et al.* 2009) experimenta a utilização da robótica pedagógica, com robôs da Lego, no auxílio ao ensino de algoritmos em uma disciplina de programação no primeiro semestre de um curso de ciência da computação. A pesquisa abordou o uso do RoboMind, uma interface de desenvolvimento para a programação de movimentos de robôs, através da qual os alunos, de forma simples, deveriam propor soluções robóticas para determinados problemas, compreendendo a complexidade de suas propostas. Um exemplo de exercício contemplado no experimento:

> Dado um mundo de 6 colunas e 3 linhas, o Robô sempre nascendo na primeira coluna da primeira linha, e com caixas distribuídas aleatoriamente na primeira linha. Ainda, pode ser que exista uma caixa na segunda linha abaixo da caixa da primeira linha. Faça o robô andar até a última coluna e quando encontrar caixas ele deve contorná-las por baixo. Lembre-se que pode existir outra caixa abaixo da primeira. Porém, não existem duas caixas na sequência na mesma linha (VAHLDICK *et al.* 2009).

A partir deste tipo de desafio, o estudante deve propor uma solução que atenda aos requisitos do enunciado, além de realizar a previsão das demais situações possíveis. A Figura

08 (a) apresenta o robô sobre uma plataforma real, com um mapa para sua movimentação, enquanto na Figura 08 (b) é representado um problema de trajetória que o aluno deverá resolver por meio da lógica e programação com o RoboMind.

Figura 8- (a) Robô sobre um mapa real, (b) Desafio para implementação de rota para o robô.



(a) **(b)**

Fonte: Vahldick *et al.* (2009).

O interesse e motivação dos estudantes em desenvolver soluções programadas e testar estas com o robô, além da busca espontânea por falhas que reduzissem a eficiência das propostas, estimularam a competição entre os alunos durante a realização das atividades com o RoboMind e Lego. A observação dos resultados ao final da disciplina de programação, em relação aos semestres anteriores, demonstrou que os estudantes obtiveram uma melhoria na compreensão dos conceitos de programação, com ênfase nas estruturas de controle.

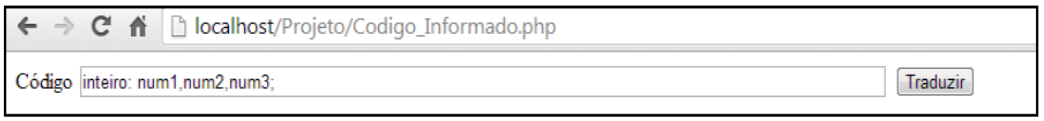
O resultado positivo, observado a partir da relação entre a lógica de programação e sua aplicação em um elemento robótico, permite uma reflexão sobre diferentes formas de explorar tais técnicas. Santos *et al.* (2007) descreve em um projeto denominado "*Utilização de robots no ensino de programação: O projecto droide*" a utilização da robótica com objetivo de validar esta técnica no apoio à aprendizagem de programação, aplicada a uma disciplina introdutória de informática, em um contexto de educação a distância. A base para este estudo é o projeto DROID (FAGIN *et al.* 2000), que vislumbra, por meio de experimentos simples e da materialização visível de teorias, estimular a capacidade criativa dos alunos na solução de problemas reais (FINKE *et al.* 1996). A inserção da educação a distância originou o projeto

DROID Virtual, com a finalidade de alcançar potenciais estudantes em regiões mais afastadas, buscando a interação virtual destes indivíduos com tecnologias de programação disponibilizadas através do projeto DROID. O Moodle foi adotado como ambiente virtual de aprendizagem para a disponibilização de materiais, recursos e comunicação entre os participantes do experimento. A teoria da Aprendizagem Situada norteou a pesquisa, com foco na implementação de uma Comunidade de Prática (LEAVE & WENGER, 1991; WENGER *et al.* 2002). A partir desta estrutura, o projeto pauta sobre a robótica como ferramenta de auxílio ao usuário na exploração de conceitos abstratos de programação, além de estimular a construção colaborativa de aplicações para robótica via *web*.

As tecnologias de simulação e animação apresentam um viés já incorporado às novas práticas de ensino de algoritmos, visto a importância de sistemas que adotem tais estruturas para a representação visual de algoritmos e estrutura de dados, no âmbito do ensino de programação. Santos & Costa (2006) defendem que as ferramentas educacionais voltadas para o desenvolvimento da prática permitem uma compreensão abstrata do aluno sobre diferentes estruturas. Beck *et al.* (2014) também vislumbram que a adoção de *software* de animação e simulação, em disciplinas de programação, pode auxiliar estudantes na experimentação e visualização de suas práticas, permitindo a percepção do processamento dos algoritmos, o que aumenta o nível de interação com a linguagem de programação e, para (FALKEMBACH & ARAÚJO, 2004) ocasiona uma melhora no processo da construção de conhecimento sobre esta área.

A pesquisa desenvolvida por Beck *et al.* (2014) discorre sobre a proposta de um ambiente de simulação e animação como ferramenta de auxílio ao ensino de programação, permitindo ao estudante acessar animações gráficas, previamente implementadas, que ilustram o funcionamento de diferentes algoritmos. O ambiente é provido de: uma descrição, para o entendimento da ferramenta; ambiente de programação, para edição do código, detecção de erros e identação; módulo usuário e especificação de estrutura de dados, composto de uma estrutura incompleta de código, para que o usuário complete o programa; arquivo de dados de teste, com as interfaces para a visualização dos códigos. Com o uso deste ambiente, o aluno desenvolve o algoritmo em pseudocódigo e a ferramenta faz a tradução do mesmo para a linguagem C, de forma dinâmica e visual, possibilitando o estudante visualizar o resultado de sua lógica em um código fonte, conforme é mostrado nas Figuras 09 (a) e (b).

Figura 9-(a) Comando descrito em Pseudocódigo, (b) Código traduzido para linguagem C.



(a)

(b)

Fonte: Beck *et al.* (2014).

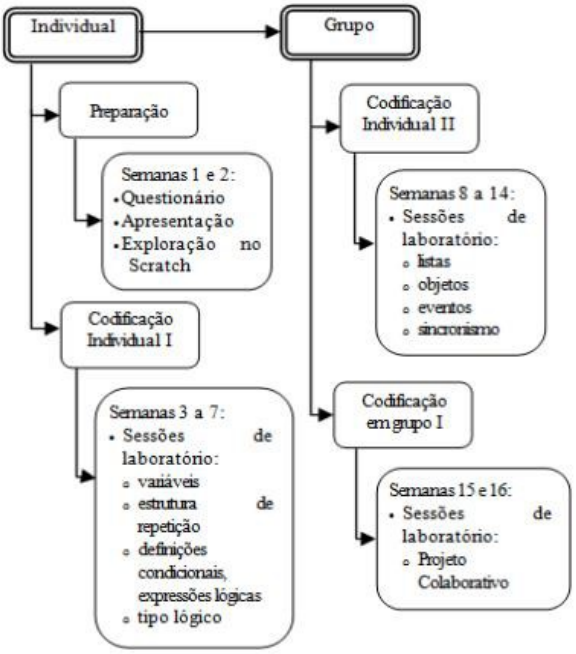
Como resultado de sua pesquisa, Beck *et al.* (2014) salienta que a construção de ferramentas que explorem sentidos audiovisuais, maximiza a capacidade de aprendizagem do estudante sobre a lógica de programação, por meio da captação e compreensão efetiva da lógica envolvida na construção das linhas de código de um programa, além de facilitar a ação do professor na explicação dos conteúdos.

Em uma concepção mais avançada, no uso da tecnologia para o ensino de algoritmos, tem-se a Realidade Virtual (RV) como uma técnica para estimular a cognição do estudante por meio da participação em outra realidade, propiciando desta forma a sensação de realmente vivenciar um mundo virtual, como real (LATTA & OBERG, 1994 e PIMENTEL & TEIXEIRA, 1995). Por sua vez, o mundo virtual é conceituado como um ambiente eletrônico que mimetiza visualmente espaços físicos complexos, onde pessoas, representadas por personagens, interagem com objetos e entre si, segundo Bainbridge (2007). A RV possui como características de suas interfaces, cinco fatores que estimulam as vias sensoriais do usuário, segundo (BRAGA, 2001): imersiva, intensiva, interativa, ilustrativa e informativa. A ideia básica desta tecnologia está baseada em três princípios: imersão, por meio de um estímulo sensorial completo; interação, descrita pela capacidade do ambiente de se modificar/adaptar de acordo com as ações do usuário; e o envolvimento, característica pautada sobre o nível de motivação do indivíduo em utilizar o sistema. A aplicação da RV na educação permite uma prospecção além-sala de aula, com a aprendizagem através do estudo e interação com o mundo virtual, sendo este um local adaptado para vivências múltiplas.

Como evolução da RV tem-se os Ambientes Imersivos ou Metaversos, sistemas gráficos 3D, que promovem a ampla interação com a aplicação. Estes ambientes se mostram efetivos no engajamento e motivação do usuário em experiências realísticas, se caracterizando como soluções válidas para ferramentas educacionais (AMARAL *et al.* 2012). Schmidt & Tarouco (2008) definem estes ambientes como espaços virtuais por onde os indivíduos navegam, interagem e vivenciam situações, utilizando uma representação personificada denominada avatar, termo utilizado para identificar o conceito de indivíduo no mundo virtual (SCHLEMMER & TREIN, 2008). Aplicados ao ensino de algoritmos, podem ser elencados alguns trabalhos científicos que exploram os ambientes imersivos Open Simulator (OpenSim) e Second Life (SL) com práticas para o auxílio e motivação dos discentes no estudo de programação.

Salgado *et al.* (2013) apresentam em sua pesquisa sobre o ensino de algoritmos e programação, uma abordagem diferenciada, com base na aprendizagem colaborativa em um ambiente imersivo, com a utilização do AVA Moodle, do Metaverso OpenSim, visualizador Imprudence e a linguagem visual Scratch, aplicado em uma disciplina de introdução à programação. A estratégia adotada baseou-se em (CASTRO & FUCKS, 2011) por meio de um esquema progressivo para aprendizagem, com foco em três elementos: indivíduo, grupo e time. As atividades relacionadas ao estudo ocorreram em fases: preparação, codificação individual I e II e codificação em grupo I, conforme a Figura 10.

Figura 10-Esquema de planejamento para o projeto



Fonte: Salgado *et al.* (2013).

Em cada fase integrante do projeto, os alunos interagiram com questionários, responderam a desafios com o uso da linguagem Scratch, os quais possuíam níveis de dificuldades variados, inicialmente de forma individual. Na etapa seguinte os estudantes trabalhavam de forma colaborativa, utilizando a ferramenta ColabWeb, na revisão das respostas aos desafios. Por fim os estudantes passaram a interagir com o mundo virtual, construindo estruturas e testando códigos aplicados a diferentes objetos. Os pesquisadores apresentaram como resultados alcançados a observação sobre a familiaridade demonstrada com o OpenSim e declarada pelos discentes, por meio de um questionário. Outra consideração relevante aponta para o interesse dos participantes em desenvolver projetos de implementação mais complexos com o uso do mundo virtual, por fim, destaca-se o caráter de motivação e aspectos positivos declarados, tais como: visualização, interação, facilidade e solução, termos que descrevem de forma sucinta um ambiente imersivo. Como aspectos negativos, foram observadas questões técnicas para a utilização do ambiente.

Um experimento de programação, com a Linguagem LSL (*Linden Script Language*), aplicada no metaverso Second Life é outro exemplo de pesquisa com o uso de ambientes imersivos no contexto do ensino de programação. Esteves *et al.* (2007), busca em sua pesquisa uma forma alternativa de aprendizagem, apostando na utilização de recursos

inteligentes como o SL, para tornar possível a contextualização do processo de ensino, permitindo assim a construção do conhecimento pelo próprio aluno, de forma independente, através do contato com ambientes ricos (virtuais) os quais dificilmente teriam contato em sala de aula. O experimento contou com a participação de nove estudantes, cinco da disciplina Laboratório I (L1) e 4 de Laboratório III (L3), do curso de Licenciatura em Informática. Dois desafios foram propostos aos alunos: os estudantes de L1 deveriam criar um cão que respondesse as ordens de seu dono, utilizando conceitos de variável, estrutura de repetição e condição, objetos, comunicação e eventos; alunos de L3 deveriam criar um robô e um comboio, utilizando os mesmos conceitos de LI, além de técnicas de comunicação (*mail*), manipulação de listas, sensores e temporizadores. Ambos os grupos possuíam liberdade de acesso e construção das soluções no metaverso, porém, deveriam reunir-se semanalmente com professores no próprio SL. As principais dificuldades descritas pelos estudantes foram relacionadas à ligação de objetos, criação de cópias e alinhamento destes elementos no mundo. Os alunos de L1 demonstraram compreensão e facilidade na construção da solução para o problema proposto. Estudantes de L3 também não apresentaram dificuldades na implementação dos códigos em LSL, contudo a manipulação de funções caracterizou o maior obstáculo para este grupo. Em relação à característica inerente à correção de erros (ESTEVES, 2004; GOMES, 2000), o grupo L3 apresentou melhor desempenho, pois conseguiram de forma autônoma avaliar e aprimorar seus programas, enquanto alunos do L1, na grande maioria das vezes não conseguiram solucionar os erros gerados.

Por fim, para encerrar esta seção sobre novas tecnologias aplicadas ao processo de aprendizagem de algoritmos, destaca-se a proposta futurista de Barbas & Lopes (2013), sobre a utilização das soluções promissoras de Realidade Aumentada (RA) (CAUDEL, 1990) e Interfaces Naturais (NUI), integradas em um protótipo, a fim de facilitar a aquisição, por parte dos estudantes, de competências básicas para programação. A tecnologia de RA, segundo Azuma (1997) e NMC (2011:16), permite que um determinado conteúdo, gerado por computador, complemente o mundo real através de uma camada contextual de informações agregadas a espaços tridimensionais em tempo real. Por sua vez as NUI permite que seus usuários participem de momentos virtuais, com movimentos próximos aos realizados no mundo real e desta forma manipulem conteúdos intuitivamente. O sistema multimídia que deverá resultar deste estudo, caracteriza-se como uma solução *e-learning*, projetada para rodar sobre plataformas multitoque, com grandes dimensões, onde os estudantes deverão se autenticar através de um método específico (cartões, *tokens* ou biometria). O contato com a

programação deverá ocorrer por peças de *puzzle* reais, representando estruturas de condição e repetição. A aproximação destes *puzzles* a plataforma disponibilizará, virtualmente, operadores aritméticos, relacionais e lógicos, os quais deverão ser organizados pelo próprio estudante, a fim de resolver um determinado problema. Para a mensuração dos resultados, os autores pretendem utilizar técnicas mistas de coleta de dados, quantitativa e qualitativamente. Como resultado, espera-se que esta seja uma solução viável e efetiva para seus utilizadores desenvolverem competências sobre programação e, especificar todos os diferentes aspectos visuais, físicos, entre outros do mundo sintético e dos objetos que o compõem.

2.2.3. ENSINO DE ALGORITMOS POR MEIO DE JOGOS

Para Barcelos *et al.* (2009) dentre as habilidades esperadas do aluno, para a construção de algoritmos e programação, podem ser citadas: concentração; capacidade de observação; desenvolvimento de estratégias para a resolução de problemas; raciocínio lógico, entre outros. Estes autores apontam ainda, a falta ou presença em menor escala destas características no estudante, como fatores diretamente associados a sua incapacidade na construção da lógica para o desenvolvimento de um programa. A partir deste ponto de vista, algumas pesquisas buscam identificar o impacto, pontos positivos e as possibilidades de melhoria que podem ser alcançadas com a utilização de jogos, o que se torna plausível de investigação, visto que o uso destes permite ao aluno, por meio da diversão, obter informações de maneira mais efetiva quando comparado a práticas pedagógicas tradicionais (MENDES, 2011).

O uso de jogos educacionais é utilizado por Souza *et al.* (2013) em uma pesquisa de cunho quantitativo, com intuito de enumerar e medir o desempenho de acadêmicos em uma disciplina de algoritmos, com o foco sobre o ensino de lógica. Este se caracterizou como um estudo pautado nas dificuldades apresentadas pelo estudante na assimilação de conteúdos. O pensamento destes pesquisadores está alinhado com a concepção de Grübel e Bez (2006), os quais defendem que a abordagem de ensino com o uso de jogos educacionais facilita o processo de aprendizagem, sendo um elemento positivo na construção do conhecimento. Como instrumento foi construído um jogo, para a resolução do problema de travessia de um rio, definido como um desafio lógico e relevante para o entendimento de algoritmos. A construção de uma sequência lógica para a solução da travessia é explorada com o intuito de estimular o raciocínio do estudante. A Figura 11 apresenta a tela de interação com o jogo

proposto, em que um fazendeiro deverá atravessar, em uma determinada ordem, de uma margem a outra do curso de água um conjunto de elementos (ovelha, alface e raposa).

Figura 11–Jogo do Fazendeiro

Fonte: Souza *et al.* (2013).

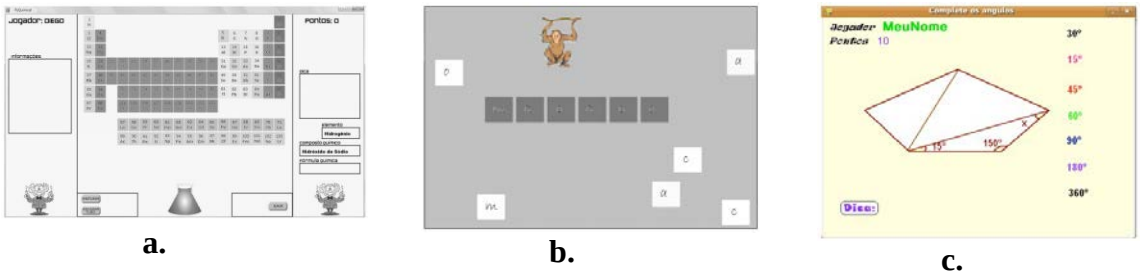
Através da possibilidade do aluno experimentar um conjunto de ações para a obtenção de uma solução viável para o problema, alguns pontos foram avaliados, dos quais destacaram-se que 75% apontaram dificuldades em entender a lógica envolvida, 100% afirmaram que a utilização do jogo é benéfica para a aprendizagem e, 50% destacaram um maior estímulo, para a resolução da atividade, com o uso do jogo. Os autores deste estudo mostraram que a metodologia proposta, lúdica, incentivou a evolução dos acadêmicos, por meio de "*um ambiente convidativo e desafiador*" (SOUZA *et al.* 2013), com resultados positivos, como o aumento no aprendizado individual e o maior envolvimento com os conceitos de programação. Contudo, destacaram que os jogos devem ser adotados como instrumentos de apoio, auxiliares e complementares, ao entendimento dos conteúdos no ensino de algoritmos.

Com o intuito de aumentar o interesse dos alunos de uma disciplina de programação orientada a objetos, em um curso de licenciatura em computação, Rebouças *et al.* (2010) explora em sua pesquisa alguns fatores motivacionais que podem ser alcançados com a adoção de jogos educacionais, utilizando como base a linguagem de programação Python. Este estudo teve como fomento inicial a resolução do Grupo de Licenciatura em Computação (GT-3), que durante o XXIV Congresso da SBC aprovou a proposta de incluir conteúdos relativos a informática, entre os quais destaca-se o tema programação, no ensino médio, com o intuito de fomentar o interesse e consequente aumento de mão-de-obra especializada nesta

área, no âmbito do País (PEREIRA JÚNIOR *et al.* 2005). O estudo objetivou a investigação e proposição de jogos, para apoiar o estudante no entendimento inicial dos conceitos de programação, aumentando desta forma sua afinidade com o tema, focando na redução da evasão e reprovação. A escolha do Python justifica-se por Ceder e Yergler (2003), que descrevem em seus estudos alguns fatores positivos no uso desta linguagem de programação, tais como: facilidade de compreensão da linguagem por iniciantes, a escrita de jogos pode ser considerada como uma prática inicial para o desenvolvimento de programas mais complexos, e a utilização da biblioteca PyGame, nativa do próprio Python possibilita o uso efetivo da técnica de jogos no ensino de algoritmos e programação.

Na primeira etapa do estudo os estudantes da disciplina foram motivados a desenvolver jogos educativos (PyQuimica, PyGonometria e PyLavra), a fim de comprovar a capacidade destes em construir soluções simples, com o uso do Python, conforme demonstrado na Figura 12.

Figura 12–Telas dos jogos a. PyQuimica, b. PyLavra, c.PyGonometria



a. **b.** **c.**

Fonte: Rebouças *et al.* (2010).

A demanda proposta aos estudantes para a construção dos algoritmos e programas, os quais pudessem ser aplicados no ensino médio, pode ser resumida em: para o PyQuimica identificar a nomenclatura de compostos químicos, componentes e fórmulas; para o PyLavra propor um método baseado em etapas para auxiliar na alfabetização de crianças, por meio de letras soltas a fim de associá-las na construção de palavras ou termos, e; o PyGonometria para o entendimento de trigonometria através da observação e definição de ângulos para um conjunto de figuras geométricas. Como resultados relevantes desta prática pedagógica, destacaram-se: que todos os grupos conseguiram desenvolver soluções para os problemas propostos; após a prática, os índices de reprovação e evasão, em comparação com semestres anteriores, sofreram quedas de 33,3% para 15,8% com a nova prática pedagógica, além do percentual de abandono de 20% para 10,5%; o índice de aprovações passou de 46,7% para 73,7%, o que comprova o maior comprometimento e motivação dos alunos; por fim, 81,3%

dos estudantes que finalizaram a disciplina afirmaram achar positiva a prática do uso de jogos educacionais como metodologia para o estudo de programação.

Outra prática pedagógica alinhada à concepção de jogos para o ensino de algoritmos e programação para iniciantes é apresentada por Scaico *et al.* (2011), o qual descreve a construção de um jogo sério com intuito de familiarizar estudantes a conceitos básicos de programação. Este estudo está pautado sobre os princípios da ausência de limites entre a aprendizagem-diversão e trabalho-jogo, buscando assim uma maior motivação e engajamento do aluno, conforme defendido por Koster (2004). Para Scaico *et al.* (2011) existe uma maior efetividade e motivação em disciplinas introdutórias de programação quando adotado o jogo como ferramenta para explorar a competição, os sentidos, e a capacidade de construção de estratégias sobre um conteúdo específico. Baseado nesta premissa, os autores apresentam um projeto para construção de um jogo sério, focado em: propor um ambiente estimulante e competitivo ao jogador, servir como solução de acompanhamento para do professor e, com a facilidade de ser utilizado em uma plataforma Android. Para o processo de ensino é utilizado o jogo Castelo dos Enigmas, o qual é focado inicialmente na apresentação de conceitos básicos sobre a sintaxe e estrutura de um algoritmo, podendo ser aplicado em um nível mais avançado, como no entendimento de estrutura de dados.

Figura 13–Jogo Castelo dos Enigmas: a. Sala do Castelo, b. Algoritmo da balança

Fonte: Scaico *et al.* (2011).

A Figura 13 apresenta dois momentos diferentes do jogo, em “a.” tem-se a sala inicial do ambiente, onde o usuário pode identificar as diferentes áreas com as quais pode interagir, enquanto a imagem “b.” demonstra um momento do *game* que o jogador deve construir uma expressão lógica a fim de complementar um determinado teste. Ao aumentar o tempo de

interação com o jogo, o estudante passa a perceber que a manipulação de objetos de forma lógica e estruturada permite a solução de alguns enigmas e que a analogia deste contexto se refere diretamente aos conhecimentos necessários para a construção de um algoritmo. Como resultados parciais os autores descrevem algumas dificuldades como a necessidade de um bom nível de criatividade para a construção dos desafios, o refinamento dos requisitos do jogo, a fim de atender padrões de usabilidade e jogabilidade, contudo os estudantes participantes dos experimentos com o jogo o classificaram como interessante e divertido, possibilitando o entendimento de significados básicos de programação.

2.2.4. ENSINO E APRENDIZAGEM DE ALGORITMOS POR MEIO DE PROGRAMAÇÃO DE BLOCOS VISUAIS E MUNDOS VIRTUAIS

Ao analisar os métodos de ensino e aprendizagem de algoritmos, uma questão importante a ser observada a capacidade do aluno em reter a informação com a qual tem contato, um conceito de inteligência. Contudo nesta área de estudo Gardner (1985) define a inteligência como uma relação intrínseca às origens biológicas da habilidade para resolução de problemas e, defende ainda que os indivíduos conseguem aprimorar qualquer área da inteligência (GARDNER, 1993). Gama (1998) classifica as inteligências em sete tipos: linguística, musical, lógico-matemática, espacial, sinestésica, interpessoal e intrapessoal.

Para a construção do conhecimento sobre programação o estudante precisa ter a capacidade de definir padrões e ordená-los de forma sistemática, reconhecer problemas e lidar com os mesmos a fim de resolvê-los, características estas pertinentes à descrição da inteligência lógico-matemática. Porém a habilidade do estudante em visualizar uma solução e mapeá-la no espaço e conceber o código relacionado o auxilia na compreensão de todos os aspectos associados à programação, o que está diretamente pautado sobre a teoria de inteligência espacial.

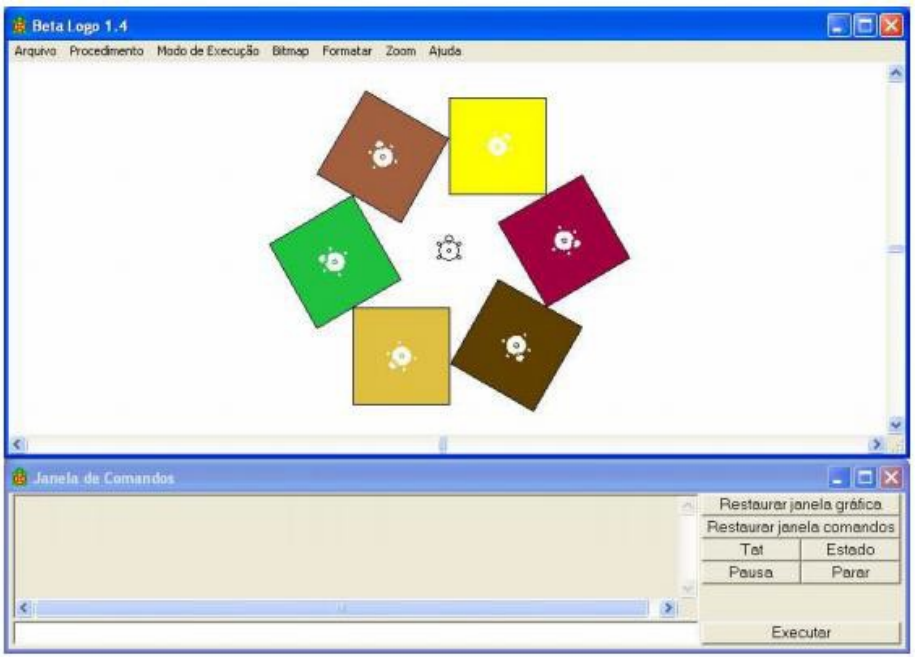
Percebendo que as interações de lógica e espaço podem auxiliar o estudante no processo de aprendizagem de algoritmos, é possível afirmar que a utilização de recursos visuais beneficie o aluno neste processo. Costelloe (2004) corrobora com esta afirmação, entretanto salienta que o material visual disponibilize um padrão adequado de interação entre o aluno e a visualização. Neste contexto, o uso de uma ferramenta de programação, com recursos visuais apurados, permite reduzir algumas das principais dificuldades dos estudantes, descritas na

literatura, relacionadas à complexidade e abstração envolvidas na implementação sintática de um código fonte, necessárias para entender uma determinada linguagem (MOTA *et al.* 2009).

Desde o início dos anos 60 desenvolvem-se pesquisas com o intuito de disponibilizar linguagens de programação e soluções de ambientes a fim de expandir o conhecimento sobre programação a um maior número de pessoas (KELLEHER & PAUSCH, 2005)

Uma das primeiras soluções visuais de programação foi a linguagem Logo (ABELSON & DISSESSA, 1981; ABELSON, 1993), uma ferramenta desenvolvida sobre a filosofia educacional construtivista, por Papert (1985) no *Massachussets Institute of Technology* (MIT). Esta foi projetada para ser uma linguagem computacional poderosa, contudo que permitisse principiantes e crianças expressarem seu raciocínio lógico de forma fácil por meio de recursos visuais. Esta ferramenta tem várias versões (proprietárias e abertas). Uma das versões gratuitas é a BetaLogo, disponível em http://www.fclar.unesp.br/betalogo. A Figura 14 apresenta a interface desta ferramenta.

Figura 14–Interface da ferramenta de programação BetaLogo 1.4



Fonte: Gregolin, 1994

A Logo, embora uma linguagem simples, na época de sua concepção não teve muito êxito na finalidade de ser uma forma fácil de programar, devido a vários motivos, conforme apresentado por Resnik *et al.* (2009): na grande maioria das vezes a introdução à programação com esta linguagem ocorre de forma não motivadora, complexa e não integrada aos interesses

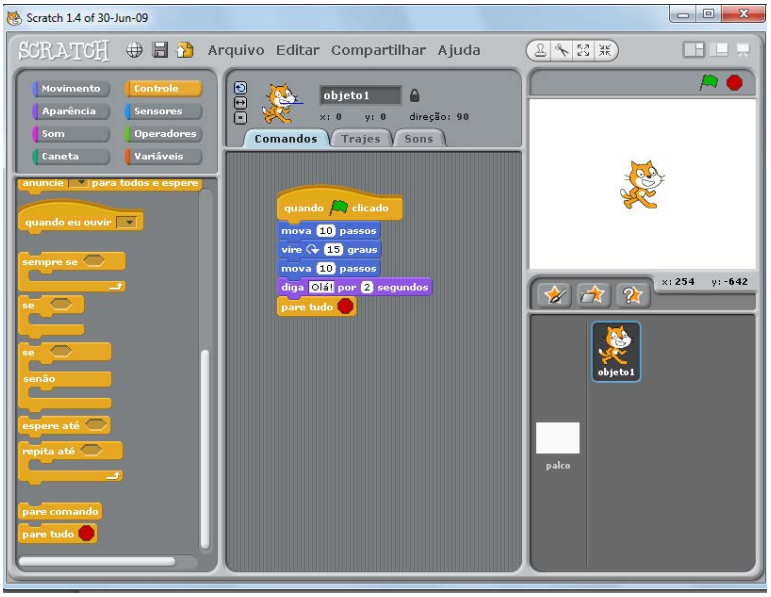
das crianças, o que gera a falta de interesse e desmotivação dos mesmos. Neste contexto Papert (1980) argumenta que o entendimento de programação deva ser fácil para começar (*low floor* ou piso baixo), que permita a construção de projetos de mais alto nível ao longo do tempo (*high ceiling* ou teto alto) e apoie diferentes projetos para indivíduos com diferentes interesses (*wide walls* ou paredes largas). Basicamente Papert defende para que uma linguagem de programação seja aceita ela deve possuir piso baixo, teto alto e paredes largas.

Seguindo estas características o *Lifelong Kindergarten Group* (LKG, 2014) desenvolveu um ambiente gráfico de programação, com uma interface baseada em ícones e blocos, denominado Scratch, com a finalidade de facilitar o processo de aprendizagem de conceitos de computação por meio do pensamento criativo, trabalho colaborativo e raciocínio sistemático (SCRATCH, 2014). Esta linguagem permite o estudante explorar diferentes conceitos, desde a construção de um algoritmo simples até a implementação de aplicações complexas com base e estruturas de repetição, decisão, laços, sequência, variáveis, tratamento de eventos (VOELCKER *et al.* 2008) e demais recursos comuns nas linguagens de programação tradicional como C e Pascal. A interface lúdica do Scratch o torna uma solução atrativa e de fácil manipulação para a construção de jogos interativos e outras soluções, que de forma visual e dinâmica, motivam o estudante a explorar suas capacidades de criação e raciocínio lógico.

Para Uludag *et al.* (2011) o Scratch permite o aprendizado de programação de computadores e algoritmos de maneira agradável, por meio de uma linguagem de programação gráfica do tipo arrastar-soltar, que permite o aluno identificar as estruturas dos códigos que estão construindo, além de perceber a dinâmica da relação dos mesmos. MIT (2014) define esta como uma linguagem de programação para as pessoas, a qual enseja construir histórias e soluções interativas, jogos, música e arte. Orzan *et al.* (2012) salienta ainda que o modelo de programação disponibilizado pelo Scratch torna agradável o processo de aprendizagem sobre algoritmos, sendo para Fal & Cagiltayc (2013) um dos modos mais fáceis de ensinar a lógica de programação e algoritmos para o estudante.

A partir da Figura 15 é possível perceber que o Scratch é uma linguagem simples, orientada a blocos de montar, permitindo assim um estilo de engajamento interativo e lúdico, o que possibilita ao aluno trabalhar de forma experimental, reavaliando suas decisões e buscando soluções diferenciadas (RESNICK & ROSENBAUM, 2013).

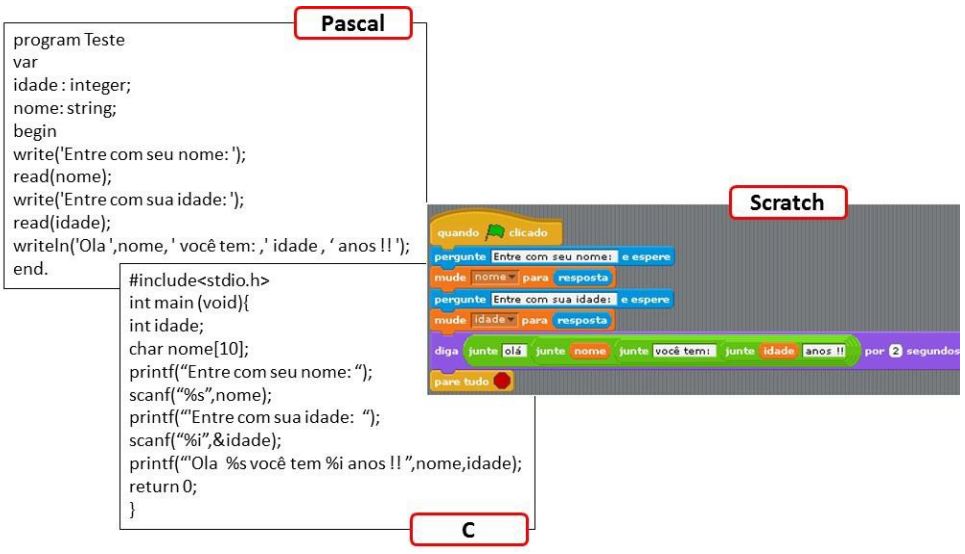
Figura 15–Interface de Programação do Scratch



Fonte: Próprio autor.

A Figura 16 apresenta três códigos, em linguagens diferentes (Pascal, C e Scratch), a fim de mostrar as diferenças de sintaxe e complexidade de implementação de cada aplicação. Neste caso específico os códigos possuem a mesma finalidade, receber nome e idade e apresentar uma mensagem na tela. A simplicidade da linguagem Scratch pode ser observada, assim como na pesquisa de Scaico *et al.* (2012).

Figura 16–Comparação do Scratch com outras linguagens de programação



Fonte: Adaptado de Scaico (2013).

Para Romeike (2007) a adoção do Scratch fornece a oportunidade ao aluno de desenvolver soluções individuais para problemas reais, sem a necessidade de um elevado conhecimento de técnicas de programação, identificando significados, construindo conexões e disponibilizando ao discente uma ferramenta que lhe permita um processo contínuo de expansão de habilidades/competências por meio da criação, experimentação e reflexão, segundo Resnick (2011), em uma espiral de pensamento criativo.

Percebe-se então, que a organização, estilo e recursos visuais do Scratch aguçam a perspicácia do aluno, motivando e estimulando sua capacidade de resolver problemas. Ao afirmar a existência desta relação é possível apontar que a integração de mais elementos neste contexto, além da simples visualização, possa contribuir de forma positiva para o processo de aprendizagem de algoritmos e programação. Com base nesta premissa, algumas pesquisas apontam o uso de mundos virtuais e ambientes imersivos como artifício de apoio ao ensino de programação.

Um mundo virtual pode ser considerado um espaço digital, tridimensional, que possibilita ao ser humano interagir por meio de avatares, trocando experiências em um espaço virtual. Este tipo de ambiente disponibiliza, através de suas características, uma interface atrativa para ações de aprendizagem, tendo como pontos relevantes por (SCHMITT & TAROUCO, 2008): interações que sempre ocorrem em primeira pessoa; ambiente basicamente gráfico/visual, onde o usuário realiza ações como no mundo real, embora possa utilizar recursos de comunicação como *chat*; O utilizador pode construir seu próprio avatar, o qual refletirá seu estilo no processo de interação com os demais usuários, tornando assim o sistema mais próximo da realidade; O metaverso apresenta-se semelhante aos *Multi User Dungeons* (MUDs) ou *Role Playing Games* (RPGs), contudo com enredos dinâmicos, construídos pelas ações dos usuários; por fim, permitem uma experiência de presença e imersão.

O uso destas tecnologias permite elevar o processo de aprendizagem a um patamar de interesse, fomentando a motivação e atenção do estudante, explorando assim os diferentes aspectos do ambiente como simulação e ações colaborativas com outros usuários. Reconhecendo este contexto é possível identificar a relação da interação entre os estudantes nos metaversos com a teoria sobre a ZDP (Zona de Desenvolvimento Proximal) de Vygotsky (1978), a qual descreve a distância entre o nível de desenvolvimento real (associado à capacidade de resolver problemas individualmente) e o nível de desenvolvimento potencial (processo de resolução de problemas por meio do contato/auxílio de um mediador, tutor ou

colega). Esta inter-relação ocorre de forma natural no mundo virtual, possibilitando que estudantes iniciantes, alunos mais experimentados (ou professores) e o próprio conteúdo interajam com o problema em busca de uma solução viável, desta maneira ampliando a capacidade de assimilação do discente e permitindo o atendimento à "janela de aprendizagem" de cada indivíduo (VYGOTSKY, 1978). Wagner (2012) apresenta, no âmbito da aplicação dos metaversos no apoio ao processo de ensino, vários projetos como: Kaneva [2], HIPIHI[3], IMVU[4], Metaplace[5], ClubPenguin[6], Second Life[7] e OpenSim[8] (GIRAFFA, 2009).

A adoção dos mundos virtuais como ferramenta de apoio aos processos de ensino e de aprendizagem de algoritmos combina todos os recursos desta tecnologia, a fim de permitir ao estudante implementar códigos de programação em um ambiente rico, com um alto nível de interação, colaboração e *feedback* automático (ESTEVES *et al.* 2007). Nos metaversos Second Life e OpenSim a linguagem de programação LSL é utilizada para a implementação de *scripts* que definem estados, eventos e comportamentos para os objetos dentro do ambiente. A sintaxe desta linguagem é muito próxima à das linguagens de programação tradicionais como C e Java, contudo um aspecto importante da construção de códigos no SL e OS permite a verificação instantânea dos códigos criados, motivando os alunos buscarem imediatamente soluções para os problemas encontrados, reduzindo a reação negativa dos alunos aos erros de compilação.

Ao se ambientar sobre as práticas e tecnologias, em estudo ou aplicadas, no ensino de algoritmos, cabe ressaltar que a correlação observada entre as informações levantadas neste referencial teórico subsidiaram a proposta desta tese, servindo como base para a modelagem e implementação do estudo.

O diferencial deste trabalho está calcado na concepção de que a construção do Pensamento Computacional e da lógica de programação, necessários para que o estudante consiga desenvolver soluções algorítmicas efetivas, pode ser alcançado por meio da integração de teorias educacionais aliadas ao uso, cronologicamente alinhado, de soluções de

---

[2] http://www.kaneva.com/

[3] http://www.hipihi.com

[4] http://www.imvu.com

[5] https://www.metaplace.com

[6] http://www.clubpenguin.com

[7] http://www.secondlife.com

[8] http://opensimulator.org

programação com blocos visuais, programação interativa com mundos virtuais e o uso de linguagens de programação tradicionais. Este modelo pedagógico sugerido vislumbra, de forma gradual, possibilitar ao aprendiz o entendimento dos conceitos básicos envolvidos nos processos de ensino e de aprendizagem de algoritmos, concebendo assim a noção necessária para a resolução lógica de problemas reais.

3. PROCEDIMENTOS METODOLÓGICOS

As dificuldades apresentadas por alunos nas disciplinas que tratam sobre os temas algoritmos e programação, são assuntos debatidos e avaliados de diferentes maneiras. Na literatura científica diversos trabalhos são publicados com alternativas viáveis para a redução da evasão e reprovação dos estudantes nestas disciplinas. Com a mesma temática, esta tese de doutorado, vislumbrou o desenvolvimento de um estudo pontual sobre as práticas pedagógicas adotadas para o ensino de programação para alunos iniciantes. Basicamente, o intuito foi prover a construção do Pensamento Computacional e da lógica de programação por meio da utilização de ambientes imersivos, linguagens de programação visuais e o paradigma de programação tradicional, integrados no conteúdo de uma disciplina de algoritmos e programação.

Para o sucesso nesta pesquisa foi importante a definição de um método científico, que propiciasse um conjunto de ações sistemáticas, a fim de conferir segurança aos resultados alcançados (LAKATOS, 2007). De acordo com os métodos descritos na literatura, esta tese baseou-se em uma proposta indutiva, a qual utilizou o registro de fatores singulares para a definição do conjunto de conclusões possíveis. A observação e identificação dos fenômenos, a partir dos experimentos pontualmente elaborados, concretizaram a caracterização deste estudo sobre uma metodologia indutiva científica (RUIZ, 1980).

A pesquisa, com base no objeto de estudo, foi desenvolvida de forma descritiva (BERVIAN *et al.* 2002), objetivando mapear as informações pertinentes ao processo de aprendizagem de algoritmos, considerando aspectos históricos, propondo um novo conceito para abordagem dos conteúdos e analisando os resultados alcançados.

Devido à concepção deste trabalho ser alinhada ao campo da educação, foi conveniente relacionar os procedimentos metodológicos enunciados, às práticas da metodologia de Pesquisa Baseada em *Design* (PBD), pois a mesma, segundo (RAMOS *et al.* 2009), está consolidada na integração da pesquisa ao desenvolvimento de interações

educativas em contextos de aprendizagem reais, sendo reconhecida internacionalmente por estes aspectos (EDELSON, 2002; BARAB & SQUIRE, 2004). As investigações alinhadas com as teorias da PBD vislumbram intervenções com o uso das tecnologias buscando a resolução para problemas nas áreas do ensino e da aprendizagem (WANG & HANNAFIN, 2005). Estes processos são apresentados de forma indissociável, pela Pesquisa Baseada em Design, conforme apontados pelos estudos de Collins *et al.* (2004), o que não ocorre em pesquisas tradicionais (RAMOS, 2010).

A adoção da PBD se fez necessária, uma vez que os problemas educativos observados nos sujeitos que participaram das práticas pedagógicas definiram o processo investigativo, aportando como elemento inicial para a pesquisa educacional (KOZMA, 2000). Visto o intuito deste estudo, em aprimorar uma prática educacional para o ensino da lógica e programação, o conjunto de ações como a análise, implementação e revisão sistemática, inerentes da Pesquisa Baseada em Design (WANG & HANNAFIN, 2004), se demonstram como métodos efetivos para a obtenção dos resultados esperados.

Para a implementação deste método realizou-se um processo de integração entre o universo participante da pesquisa, neste caso os alunos das disciplinas de algoritmos, com as intervenções propostas, a fim de garantir a evolução tanto das teorias quanto das práticas desenvolvidas durante todas as etapas desta Tese. Algumas das características da PBD segundo Wang & Hannafin (2005), as quais foram alinhadas com o estudo atual, são descritas a seguir:

a. Pragmática, característica focada na evolução positiva da teoria e prática, a qual é apresentada neste estudo por meio da evolução e constantes mudanças inseridas na construção e aplicação dos projetos piloto e experimento final;

b. Interativa, atributo que aponta para a relação entre pesquisador e demais participantes no processo de *design*, em um ciclo interativo relacionando as etapas de análise, projeto, desenvolvimento e avaliação, buscando adequar o método e mantê-lo efetivo. Este atributo é observado na aplicação e análise dos resultados dos diferentes experimentos realizados em sala de aula, como a utilização do mundo virtual e programação com diferentes paradigmas, onde os estudantes registravam suas opiniões sobre as atividades em um conjunto de instrumentos de pesquisa (questionários) ao longo dos semestres;

c. Fundamentada, condiz à adoção de uma base teórica pertinente, aplicável ao contexto do mundo real, para a realização do conjunto de experimentos propostos, no caso específico desta Tese, diferentes autores e conceitos reconhecidos nortearam a implementação das diferentes ações;

d. Contextual, as etapas que envolvem a pesquisa devem ser documentados, assim como os processos de configuração e *design*, os quais devem servir de orientação para a aplicação das propostas. Neste sentido, todas as atividades relacionadas a definição do projeto piloto e experimento final foram devidamente planejadas e executadas e, os resultados observados registrados, o que possibilitou a tomada de decisão em cada etapa deste trabalho;

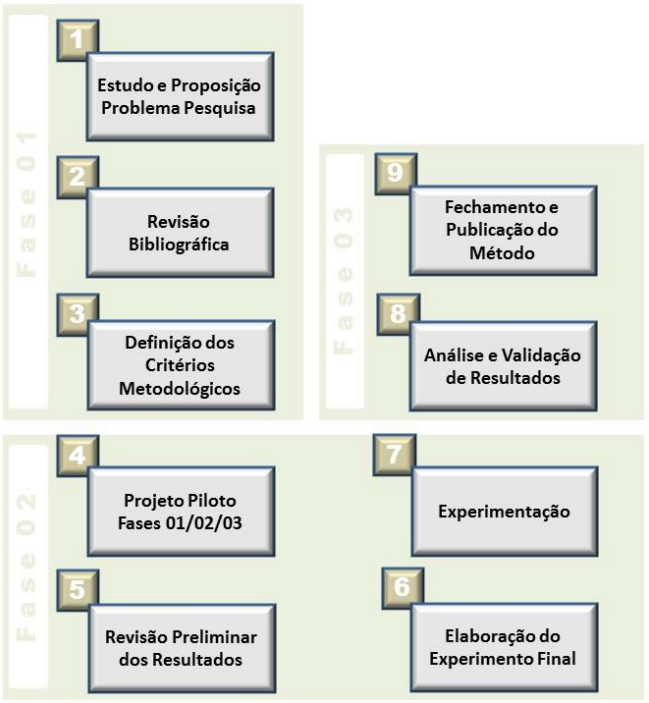
e. Integrativos, princípio que aponta o alinhamento de diferentes métodos de pesquisa em um estudo único, com intuito de abordar da melhor forma possível os diversos momentos da investigação e assim aumentar a confiança nos resultados alcançados. A metodologia de pesquisa indutiva aliada a PBD, assim como a adoção da técnica de casos múltiplos garantem que esta Tese esteja pontualmente alinhada a esta característica;

Além destes fatores, o compromisso na construção do conhecimento, almejado nesta tese, culminou não apenas em investigações, mas principalmente na construção de práticas educativas a partir dos resultados observados (KELLY, 2006). Tais intervenções incorporaram as observações realizadas sobre a relação entre ensinar e aprender, possibilitando associar a teoria, os elementos propostos e a prática no contexto da educação segundo DBRC (2003) e Struchiner (2006).

3.1. ETAPAS E PROCEDIMENTOS

Tendo a avaliação formativa durante o processo de aprendizagem como alvo primordial, foi possível dividir as práticas desta pesquisa em 09 etapas, claramente definidas, conforme mostrado na Figura 17.

Figura 17-Esquema de planejamento para a pesquisa



Fonte: Próprio autor.

Estas etapas contemplam um conjunto de ações previamente avaliadas e definidas de acordo com a importância de uma tese de doutorado, respeitando a metodologia científica adotada e parametrizando todas as ações do pesquisador. Um resumo de cada uma destas etapas é apresentado a seguir:

Etapa 01 (Estudo e proposição do problema de pesquisa): caracterizado como primeiro momento deste trabalho, a definição do problema de pesquisa ocorreu com base em estudos relacionados à aplicação da tecnologia no processo educativo. Como resultado desta ação, identificaram-se os principais problemas inerentes aos processos de ensino e de aprendizagem de algoritmos, elucidando as soluções atualmente utilizadas e tendências futuras. Investigaram-se teorias de aprendizagem capazes de proporcionar subsídio às decisões de delineamento de novas estratégias usando tecnologia, dentre as quais se destacaram o uso de ambientes imersivos como solução para o aprimoramento das práticas pedagógicas, dentro e fora de sala de aula. Reconhecendo o valor da utilização dos mundos virtuais, buscou-se um problema real para a sua aplicação. A vasta literatura e trabalhos científicos discorrendo sobre as dificuldades apresentadas por alunos na aprendizagem de algoritmos culminaram na definição da proposta desta tese, a qual vislumbra a integração de um metaverso às práticas pedagógicas tradicionais em uma disciplina de algoritmos, além da utilização de técnicas de

programação visual, orientada a blocos e a inserção de algumas teorias educacionais como o ciclo da aprendizagem experiencial e aprendizagem significativa.

Etapa 02 (Revisão bibliográfica): esta etapa contemplou o levantamento, estudo e análise crítica das produções acadêmicas relacionadas ao problema de pesquisa. A busca de material científico para o apoio na construção das proposições deste trabalho foram os objetivos principais desta fase e ao longo do projeto, com o intuito de construir um robusto referencial sobre os aspectos relacionados ao uso da realidade virtual, ambientes imersivos, laboratórios virtuais de aprendizagem, conteúdos e tecnologias utilizados no ensino de algoritmos, teorias de aprendizagem entre outros temas relevantes, foi realizado um extensivo estudo e mineração de informações em diferentes repositórios como o portal de periódicos da CAPES, bibliotecas digitais (IEEE, ACM, SCOPUS), anais dos principais eventos e periódicos na área de tecnologias aplicadas à educação.

Etapa 03 (Definição dos critérios metodológicos): visto a existência de um conjunto de diferentes teorias para a elaboração de trabalhos e pesquisas científicas (métodos dedutivos, indutivos, formais, vulgares ou científicos), esta etapa na elaboração da tese caracterizou-se pelo planejamento do estudo e definição das práticas que foram executadas, sobre uma base metodológica criticamente fundamentada e que atendiam as demandas inerentes ao tema e aos aspectos do estudo, as quais proveram uma estratégia clara para as ações do pesquisador na obtenção/análise/divulgação dos resultados alcançados.

Etapa 04 (Projeto piloto – Fases 01/02/03): o Projeto Piloto foi considerado um elemento experimental, utilizado para analisar diferentes aspectos deste estudo, através do qual foi possível avaliá-lo, identificando os processos envolvidos e validando sua aplicação. No contexto da tese, considerou-se esta etapa como a primeira fase experimental, contemplando três fases distintas, com o intuito de identificar a validade dos métodos, ideias, tecnologias e ferramentas elencadas para nortear os estudos, e desta forma apoiar nos processos decisórios necessários para a elaboração da pesquisa. O estudo do caso-piloto permitiu refinar os planos de coleta de dados, com relação ao conteúdo destes e aos procedimentos a serem elaborados (YIN, 2010). Uma breve descrição das fases do projeto piloto desenvolvido seguem abaixo:

- A primeira fase contemplou um experimento com alunos de uma disciplina de algoritmos e programação, em que foram adotados como prática pedagógica a utilização de um ambiente imersivo integrado ao Sloodle e a ferramenta de

desenvolvimento IDEOne. O objetivo deste consistiu em avaliar o desempenho dos estudantes na resolução de problemas de programação, interagindo com outros discentes em um mundo virtual. Este estudo vislumbrou também, coletar dados pertinentes para identificação das dificuldades e limitações no uso destas tecnologias.

- Como fase dois do projeto piloto, foi proposta a modificação do plano de ensino da disciplina de algoritmos e programação, a fim de inserir na ementa do curso a utilização de ambientes imersivos e outros paradigmas de programação. O intuito principal deste experimento foi analisar o quanto a mudança no modelo de programação e os aspectos inerentes à adoção de mundos virtuais influenciariam no desempenho dos alunos em relação aos índices de reprovação, evasão e abandono. Aspectos como o aprimoramento na construção do Pensamento Computacional, do raciocínio lógico e da motivação na resolução de problemas também foram alvos de estudo e análise.
- A fase três seguiu o mesmo modelo proposto na segunda fase do projeto piloto, contudo os conteúdos abordados no plano de ensino tiveram sua ordem de exposição invertidas, a fim de identificar o nível de percepção dos alunos iniciando os estudos de programação pela linguagem C (segunda fase do projeto) e iniciando pelo Scratch (terceira fase). Este momento foi destinado a verificar sobre qual sistema os estudantes conseguiriam construir o conhecimento efetivo sobre a lógica de programação.

Etapa 05 (Revisão preliminar dos resultados): etapa destinada a avaliar, de forma minuciosa, os resultados alcançados até a fase dois do projeto piloto. Buscou-se nesta fase coletar evidências que demonstrassem a efetividade do modelo educacional proposto. Informações relacionadas ao universo de participantes dos experimentos, suas observações em relação às práticas em sala de aula, a contextualização da mudança de paradigma, os índices de aprovação, reprovação e evasão foram dissecados, a fim de se construir um parecer sobre as atividades realizadas. Dados históricos sobre a disciplina de algoritmos, de semestres anteriores no Campus Bagé da Unipampa, também foram apurados, com o intuito de realizar o cruzamento de informações, as quais serviram de base para a emissão de um parecer válido, sobre as proposições deste projeto, permitindo assim o desenvolvimento de linhas relevantes de questões e até mesmo proporcionando alguns esclarecimentos conceituais. As ações referentes à Revisão de resultados visaram analisar os dados obtidos com a terceira fase do

projeto piloto, além de correlacionar o resultado desta análise com as informações já observadas. O conhecimento obtido neste processo subsidiou os experimentos finais desta tese.

Etapa 06 (Elaboração do experimento final): esta etapa destinou-se à validação/análise de todos os resultados reunidos, observados na etapa 05 (Análise dos resultados obtidos em todas as fases do Projeto Piloto). A partir da verificação dos dados foi definido um escopo para o experimento final. As informações coletadas permitiram a definição de um protocolo personalizado, descrevendo os procedimentos formais que nortearam as ações de investigação realizadas. A adoção desta concepção possibilitou identificar diferentes fenômenos válidos, a partir de ângulos e abordagens distintas das adotadas no projeto piloto, as quais originaram o modelo e organização didática utilizada na fase de experimentação.

Etapa 07 (Experimentação): esta fase foi definida como última etapa dos experimentos, a qual se caracterizou pela organização, em forma de um projeto pedagógico, das práticas definidas e fundamentadas nas etapas anteriores. O manejo e estruturação do experimento, definição do universo participante (quantitativa e qualitativamente) foram os pontos cruciais para a obtenção dos resultados. Além disso, o tempo de aplicação, recursos utilizados e dados coletados foram devidamente documentados e utilizados para a fundamentação das conclusões apresentadas nesta tese.

Etapa 08 (Análise e validação dos resultados): a análise final e discussão sobre os resultados alcançados, com o projeto piloto e demais experimentos, foram realizados nesta etapa. O conjunto de ações realizadas até este momento permitiram a implementação de uma linha temporal, contendo todos os dados observados, do início ao encerramento dos experimentos. A partir da coleção de informações observadas e da utilização de técnicas estatísticas específicas (adequadas para a observação da correlação dos dados) foi possível elucidar dúvidas e indagações relacionadas às diferentes ações e experimentos desta pesquisa. O foco principal desta fase foi repousar sobre a avaliação da efetividade de investimento sobre a proposta defendida neste trabalho, exaltando ou desqualificando as ações e práticas executadas.

Etapa 09 (Fechamento e publicação do método): por fim, esta etapa consistiu na emissão de uma conclusão sobre a pesquisa desenvolvida, tendo como base os conceitos teóricos estudados e os resultados alcançados. A análise crítica das ações alinhada à avaliação dos dados observados permitiram a construção de uma resposta ao problema de pesquisa

elencado. Concomitantemente a todas as fases desta tese foram elaborados artigos científicos, os quais estratificaram as diferentes ações envolvidas no desenvolvimento desta tese.

Com este conjunto de etapas foi possível a proposição clara e terminantemente formal das ações desenvolvidas durante a elaboração da tese. O foco desta pesquisa pode ser determinado sobre a busca, promoção e aquisição, de maneira estruturada, de novos conhecimentos científicos para a solução do problema proposto.

3.2. ORGANIZAÇÃO DOS EXPERIMENTOS

A partir do referencial estudado e da estrutura metodológica proposta foram concebidos e planejados um conjunto de experimentos a fim de atender os requisitos desta pesquisa e, apontar um método efetivo para o ensino da lógica de programação. Foram realizados 4 experimentos (4 etapas diferentes), utilizando como grupo de estudos disciplinas de Algoritmos e Programação (60 horas), totalizando 240 horas de observações e coletas de dados, as quais contemplaram os 3 projetos piloto e o experimento final.

Para a validação das práticas e dos resultados observados foi adotado a metodologia de estudo de casos múltiplos, visto que esta é indicada para situações onde o pesquisador busca ampliar o conhecimento sobre um determinado tema, contudo não possui controle total sobre o fenômeno avaliado, conforme descrito por Yin (1994). Neste contexto, as 4 turmas elencadas para a participação da pesquisa foram selecionadas de maneira intencional, sendo relacionadas disciplinas com alunos de diferentes cursos superiores, incluindo estudantes de computação. A técnica de casos múltiplos permitiu ampliar a efetividade dos resultados alcançados, pois possibilitou a replicação dos casos no segundo e terceiro projeto piloto, também no experimento final, garantindo assim comparações deliberadas e contrastantes, reforçando os achados do estudo (YIN, 2002; DUARTE, 2008).

O primeiro caso, implementado no semestre 2013/1, teve como objetivo principal identificar a viabilidade da utilização do mundo virtual Open Simulator como ferramenta de apoio na disciplina de algoritmos. Este experimento ocorreu no intervalo entre a segunda e terceira avaliação da disciplina, onde um conjunto de atividades, com o uso do ambiente imersivo, foram propostas aos discentes: configurar seus avatares, interagir com colegas, conectar ao ambiente Sloodle, desenvolver aplicações com a linguagem C (através do software IDEOne) e trocar experiências com outros avatares. Vários aspectos foram avaliados

durante estas atividades, como: suporte tecnológico do ambiente para um número elevado de estudantes, infraestrutura de rede para o número de conexões, capacidade de interação dos discentes, além de analisar as características relacionadas à motivação do uso de ambientes imersivos no desenvolvimento das tarefas de programação. Os resultados deste experimento foram averiguados com base em um instrumento de pesquisa, respondido pelos alunos após as práticas realizadas e, também com uma análise estatística da evolução das suas notas nas avaliações posteriores a prática proposta. Ao término deste caso foi possível apontar a efetividade na utilização dos recursos do Open Simulator como apoio ao processo de ensino e aprendizagem de Algoritmos.

A partir do reconhecimento da viabilidade de utilização dos ambientes imersivos, realizou-se projeto, análise e implementação do segundo caso, durante o semestre de 2013/2. O desenvolvimento desta experiência esteve calcado na reorganização do plano de ensino da disciplina de Algoritmos e Programação, com o intuito de apresentar aos alunos, além dos conteúdos de lógica e linguagem C, outros paradigmas que pudessem auxiliar na tarefa da formação do Pensamento Computacional e da lógica de programação. Desta forma os conteúdos da disciplina tradicional foram condensados na primeira fase da disciplina e, novos temas como programação com a linguagem Scratch, Scratch4OS e LSL passaram a compor o final deste plano de ensino. A avaliação do caso ocorreu por meio da análise estatística, do desempenho acadêmico dos alunos (notas durante todo semestre) e, observação de instrumentos de pesquisa respondidos pelos mesmos durante diferentes momentos no semestre.

O terceiro caso baseou-se em uma réplica do segundo, durante o semestre de 2014/1, contudo neste experimento o plano de ensino da disciplina teve os conteúdos organizados de maneira inversa ao caso 2. Os paradigmas de blocos de programação visual e programação para mundos virtuais (LSL) ficaram dispostos no inicio do semestre e, os conteúdos de programação com a linguagem C no final do mesmo período. O projeto de alteração, na ordem dos temas apresentados em sala de aula, foi motivado pelo interesse em saber qual disposição de conteúdos obteria um melhor rendimento dos alunos. Realizou-se a análise dos resultados do caso 3 da mesma forma que no semestre anterior, a replicação teve como intuito garantir a obtenção de repostas fidedignas, validando a comparação das informações coletadas.

O caso 4, experimento final (2014/2), foi projetado a partir do reconhecimento da efetividade dos casos anteriores. Embora os resultados alcançados com o caso 2 e 3 tenham

ficado muito próximos, concluiu-se a partir da avaliação do desempenho acadêmico dos discentes, que a variação positiva nas médias dos alunos do terceiro caso possibilitou a tomada de decisão em relação a qual organização de conteúdo atendeu melhor a demanda de formação do Pensamento Computacional e capacidade de programação dos estudantes. Desta forma foi realizado o estudo e planejamento do experimento final, o qual foi constituído de um conjunto de atividades formativas e avaliativas, implementadas de maneira alinhada a uma adaptação da técnica do Raciocínio Baseado em Casos, com o intuito de observar a evolução no raciocínio lógico dos alunos, em relação ao contato com os novos conteúdos. Além dos resultados gerados por esta prática, foi proposto uma taxonomia, denominada Taxonomia para Avaliação do Pensamento Computacional, utilizada para identificar o desenvolvimento do PC dos discentes participantes do experimento. Por fim, realizou-se uma inferência estatística apurada sobre o desempenho acadêmico dos alunos participantes deste caso e, também a comparação das suas médias com as médias dos demais estudantes da disciplina de Algoritmos dos 4 semestres de duração desta pesquisa (2013/1, 2013/2, 2014/1 e 2014/2). A constatação, a partir dos resultados observados, viabilizou apresentar uma conclusão sobre a efetividade do modelo proposto, como metodologia válida para o apoio ao processo de ensino e de aprendizagem da lógica e programação.

4. EXPERIMENTO PILOTO

A meta do experimento piloto, no contexto deste estudo, foi garantir a identificação de métodos e procedimentos apropriados, os quais antecederam os estudos e procedimentos experimentais finais desta tese. Esta fase objetiva auxiliar a definição dos critérios operacionais a serem adotados e a aplicação dos casos finais no projeto de estudo, por meio do refinamento dos planos de execução da pesquisa. Seguindo a conceituação de Yin (2010), o experimento piloto caracterizou-se como um laboratório para o detalhamento de todas as ações a comporem o trabalho, fornecendo um valioso *feedback* para servir de subsídio a esta proposta.

O projeto piloto, especificamente nesta proposição, foi realizado em três momentos diferentes, sob óticas também diferenciadas, porém com objetivos muito próximos, os quais são descritos nas seções 4.1, 4.2 e 4.3.

4.1. PROJETO PILOTO – FASE 01

A primeira fase do projeto piloto contemplou a avaliação da aplicação do metaverso Open Simulator, integrado à tecnologia de um ambiente virtual de aprendizagem (Sloodle) e a ferramenta de desenvolvimento de código IDEOne como atividade didática, em alguns períodos da disciplina de Algoritmos e Programação na Universidade Federal do Pampa, Campus Bagé. O intuito desta experiência foi avaliar a percepção dos alunos ao utilizar um mundo virtual como sala aula, experimentando a sensação de imersão e da interação com outros estudantes na realização de práticas de programação, além de validar o uso de diferentes tecnologias agregadas ao mundo virtual.

A elaboração deste experimento seguiu um conjunto de etapas, conforme segue:

- Levantamento bibliográfico sobre a utilização de ambientes imersivos como prática pedagógica no processo de aprendizagem;
- Estudo sobre as principais características, conceitos e produções científicas da integração de novas tecnologias aos mundos virtuais, em ambientes educacionais;
- Avaliação sobre as tecnologias a serem utilizadas no experimento, contemplando: definição do mundo virtual, do visualizador, do ambiente virtual de aprendizagem e da ferramenta para a construção de programas;
- Definição do universo de estudantes a participarem da prática proposta;
- Proposição de conteúdos relacionados à disciplina de algoritmos, a serem estudados no mundo virtual;
- Estudo e implementação de um questionário, a fim de obter a percepção dos alunos sobre a experiência de participar de uma atividade acadêmica relacionada a disciplina de algoritmos em mundo virtual, interagindo com colegas e diferentes recursos;
- Analisar criticamente os resultados alcançados.

A fim de esclarecer alguns pontos enunciados nas etapas da pesquisa é pertinente apontar a escolha das tecnologias, práticas e universo da pesquisa:

- Em relação à definição do mundo virtual, vários metaversos foram avaliados (Second Life, Wonderland, FireStorm e Open Simulator), contudo elencou-se o Open Simulator (OPENSIM, 2013), como uma solução viável para este estudo, visto se caracterizar um recurso tecnológico pertinente para apoiar o processo de aprendizagem, devido a sua capacidade de simular um ambiente 3D com altos níveis de interação, possibilitando a programação do seu estado e seus objetos, vasta literatura disponível, além de ser uma solução livre, não proprietária. Como visualizador para acesso ao OpenSim, foram elencados o FireStorm e Imprudence, devido a facilidade de uso e estabilidade destas aplicações.
- A partir da definição do OpenSim para a escolha do ambiente virtual de aprendizagem, seguiu-se a premissa de compatibilidade com o mundo virtual, se destacando neste quesito o AVA Moodle (MOODLE, 2013), através do módulo Sloodle (SLOODLE, 2013), o qual fornece uma interface de integração do Moodle com o metaverso, possibilitando ao estudante ter acesso a diferentes recursos do AVA por meio de um objeto no próprio mundo, caracterizando uma maior sensação de imersão.

- Como ferramenta para o desenvolvimento de códigos foi elencado o IDEOne (IDEONE, 2013), um *software* construído para disponibilizar ao estudante um IDE para o desenvolvimento de algoritmos com suporte a diversas linguagens de programação. Como principais motivações para a escolha desta aplicação, destacam-se a simplicidade da interface, usabilidade e capacidade de *feedback* automático sobre os erros de programação cometidos pelo estudante.
- Os participantes da pesquisa foram alunos da disciplina de algoritmos e programação, ministrada pelo autor deste projeto, no curso de Engenharia de Computação durante o primeiro semestre de 2013. O universo foi composto de 14 alunos, com faixa etária média de 20 anos.
- Quanto aos conteúdos definidos para o experimento, elencou-se a linguagem de Programação C, devido a mesma ser a base para a disciplina de algoritmos e aos temas abordados referirem-se ao desenvolvimento de vetores e matrizes, seguindo a ementa da disciplina nos períodos de aplicação do experimento.

Como atividade inicial do experimento, os alunos foram orientados sobre a utilização do visualizador e convidados a configurar os seus avatares. Neste momento observou-se um aumento significativo na motivação e interesse dos alunos em relação à utilização do ambiente imersivo. Neste contexto, foi demonstrado como os estudantes poderiam se comunicar através de *chat*, permitindo um alto nível de interação entre todos os participantes.

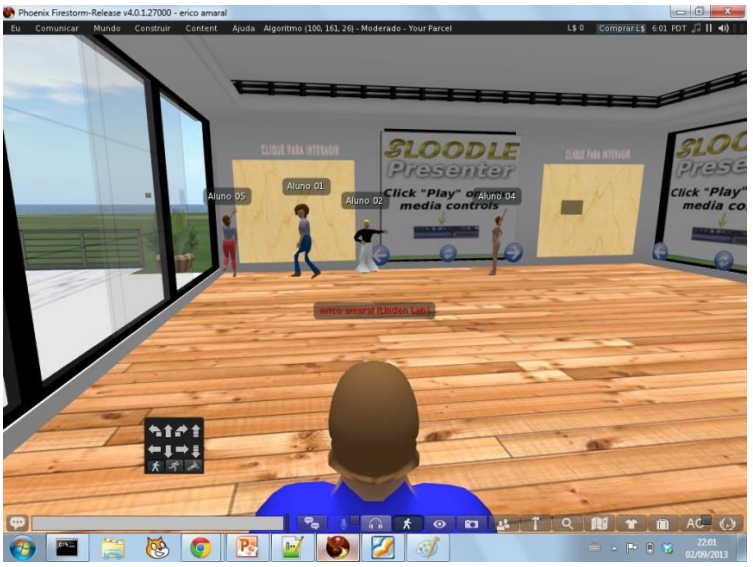
Na construção da sala de aula no mundo virtual foram contemplados um conjunto de recursos, vislumbrando para uma atividade acadêmica rica. O foco dos materiais disponibilizados se referia ao ensino de vetores e matrizes, utilizando como base o desenvolvimento de código em linguagem C, conforme mostrado na Figura18.

Figura 18-Sala de aula para o ensino de vetores e matrizes no OpenSim

Fonte: Próprio autor.

Em vários painéis no metaverso, por meio do Sloodle, foram disponibilizados materiais didáticos referentes aos conteúdos da aula. Os alunos deveriam estudar estes assuntos e ativar a aplicação IDEOne, por meio de painéis com acesso à *web*, para a digitação e teste dos algoritmos propostos. Com o intuito de atender um número elevado de estudantes, também foram habilitadas diferentes instâncias do IDEOne na sala de aula, como apresentado na Figura 19.

Figura 19-Interação dos alunos entre si e com recursos do metaverso



Fonte: Próprio autor.

Alguns pontos em relação às características técnicas, na utilização do OpenSim puderam ser observadas, servindo de base para os próximos estudos deste projeto: visto que o mundo virtual estava rodando sobre uma plataforma cliente/servidor (Grid), todos os alunos conseguiram de maneira satisfatória, acessar a sala de aula no metaverso; o sistema suportou mais de 20 conexões simultâneas, o que garante um grau de confiabilidade para a utilização do mesmo em práticas acadêmicas com este número de alunos; a comunicação entre os avatares também ocorreu de forma simultânea, sem atrasos, corroborando com a afirmação anterior; todos os alunos acessaram tanto o Sloodle, quanto o IDEOne, com atrasos aceitáveis na carga das aplicações.

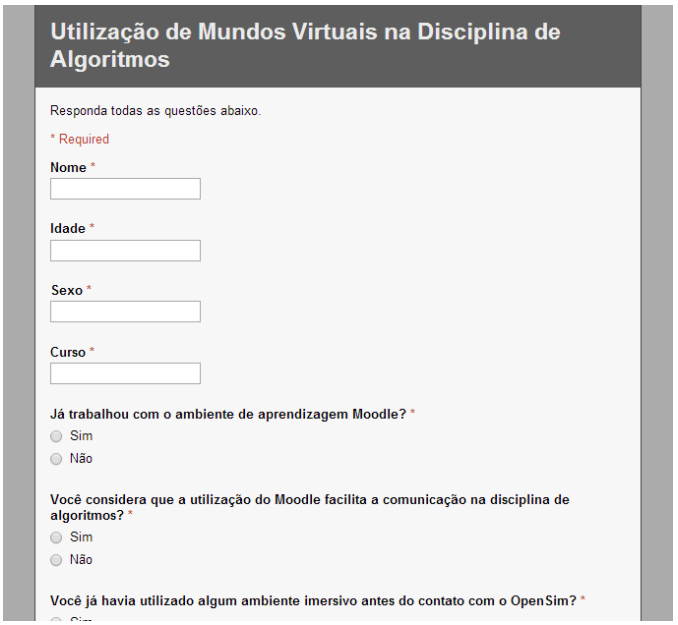
Com o intuito de identificar a percepção dos alunos em relação à utilização do ambiente imersivo, após a atividade foi disponibilizado um questionário para que os mesmos

respondessem manifestando suas considerações sobre a prática. Salienta-se que a análise de tais resultados serviu de base para a elaboração da segunda fase do experimento piloto.

4.1.1. ANÁLISE DE RESULTADOS – PROJETO PILOTO FASE 01

O instrumento de pesquisa proposto aos alunos estava constituído de 15 perguntas, abertas e objetivas, focadas especificamente em pontos pertinentes da utilização do mundo virtual e outras tecnologias integradas a este, em uma atividade acadêmica sobre algoritmos. A Figura 20 demonstra parte da tela do formulário, construído com a ferramenta DOCs do Google.

Figura 20-Instrumento de pesquisa disponibilizado aos alunos



Fonte: Próprio autor.

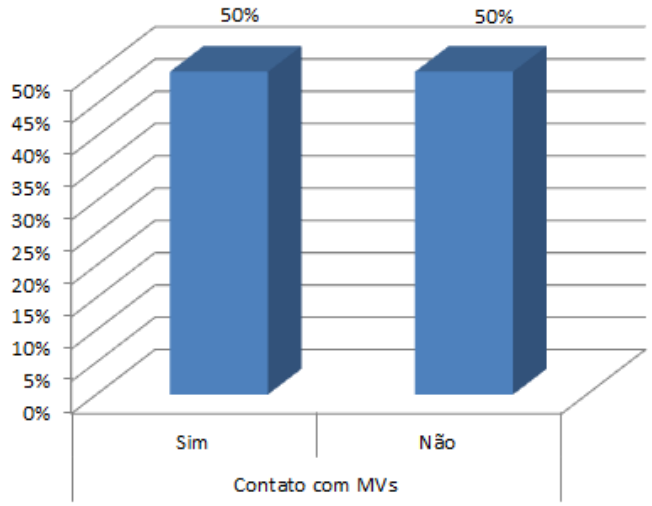
Um extrato das questões, com maior grau de relevância para este ponto da pesquisa é demonstrado no Quadro 07.

Quadro 7- Questões com maior relevância na avaliação do experimento

1. **Você já havia utilizado algum ambiente imersivo antes do contato com o OpenSim?**
2. **Entre uma aula tradicional e uma aula no OpenSim, qual você considera que prende mais a sua atenção?**
3. **Você conseguiu interagir com o ambiente, materiais, objetos e outros avatares?**
4. **Qual elemento, dentro do Mundo Virtual, chamou mais a sua atenção?**
5. **Sobre o conteúdo da disciplina de algoritmos (achou interessante o acesso via mundo virtual)?**

A análise dos resultados das perguntas do Quadro 07 permite a elaboração de algumas considerações. Em relação à primeira questão, conforme é mostrado no Gráfico 03, é possível identificar que 50% dos alunos afirmaram que já tiveram contato com algum ambiente imersivo. Esta afirmação é aceitável, visto que o universo de estudantes participantes da atividade faz parte do curso de Engenharia de Computação, devido a isso é plausível supor que tenham contato com vários tipos de tecnologias, constantemente.

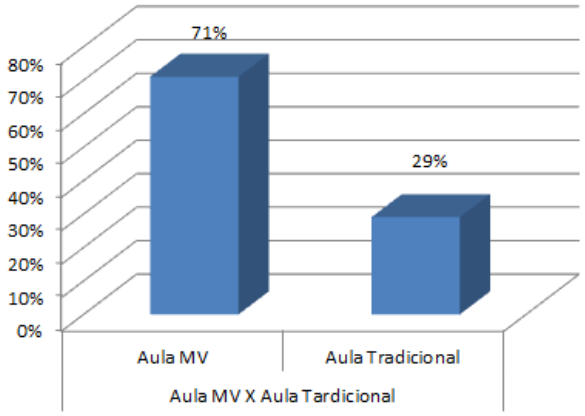
Gráfico 3–Contato com Ambientes Imersivos



Fonte: Próprio autor.

O gráfico 04 apresenta o resultado da segunda pergunta, que discorre sobre a capacidade do metaverso em estimular a atenção do aluno (questão 02), durante a atividade de ensino. Neste contexto 71% defendem ser mais interessante a utilização do MV, a ponto de ater sua atenção, diferentemente de aulas tradicionais, resultado corroborando com Braga (2001) em sua afirmação que este tipo de ambientes atrai o usuário e disponibiliza uma aprendizagem efetiva e ampla, pois se tratam de locais com uma vasta gama de possibilidades.

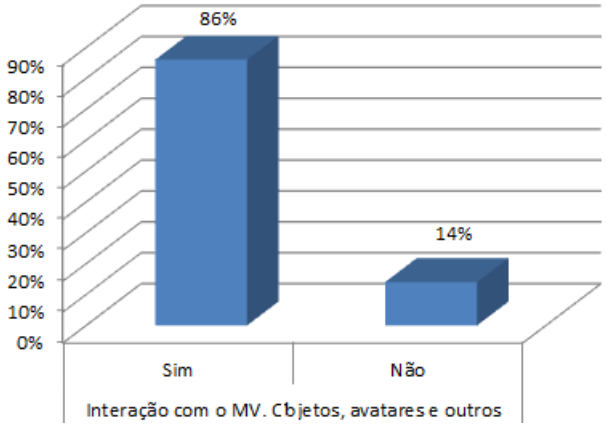
Gráfico 4–Qual prática didática prende mais atenção do aluno



Fonte: Próprio autor.

Quando questionados sobre a interação no âmbito do ambiente imersivo (questão 03) com avatares, objetos, materiais e outras tecnologias, a grande maioria (86%) demonstrou ter conseguido de forma satisfatória, interagir com todos os elementos no mundo virtual, segundo o Gráfico 05. Este resultado reflete a necessidade de se explorar o conceito de ZDP (VIGOTSKY, 1978), visto que o aluno tem a sua disposição no ambiente um conjunto de recursos, além de outros usuários (avatares), com os quais é possível a interação e troca de informações, desta forma tornando o metaverso um local propício para a construção do conhecimento.

Gráfico 5–Interação com os elementos do MV

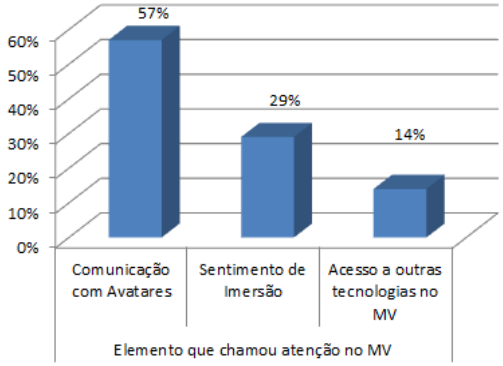


Fonte: Próprio autor.

Sobre o fato de interação com os elementos no OpenSim (Gráfico 06), 57% afirmaram que a comunicação com outros avatares foi o ponto que chamou mais atenção, durante o experimento. Neste ponto, 29% dos alunos descreveram que a situação de imersão prendeu

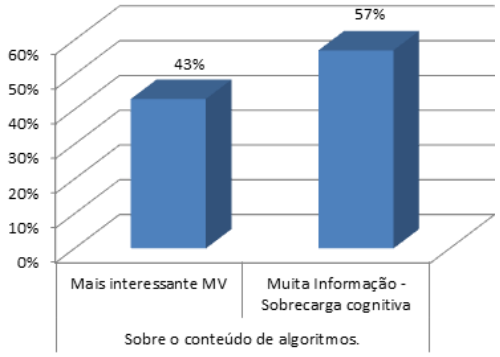
sua atenção, enquanto os 14% restantes, apontaram a interação com as tecnologias disponíveis no metaverso ser o fato mais interessante da atividade. As observações das informações coletadas nestas questões mostram que mudanças no processo de ensino, com o uso de tecnologias emergentes e inovadoras (mundos virtuais), se tornam atrativas para o estudante e atendem os requisitos educacionais definidos por Moran (2000), que aponta o uso das telemáticas, audiovisuais, textuais e lúdicas como métodos importantes a serem utilizados em sala de aula. Além disso, aspectos de colaboração denotados apontam para a possibilidade de uma abordagem educacional que permite aos alunos trabalhar em conjunto, na construção do conhecimento, estimulando assim a sua concreta participação nas ações do grupo em um ambiente imersivo, favorecendo o seu desenvolvimento cognitivo, conforme Harasim *et al.* 1997.

Gráfico 6–O que chamou atenção no MV



Fonte: Próprio autor.

Gráfico 7–Uso do MV em algoritmos



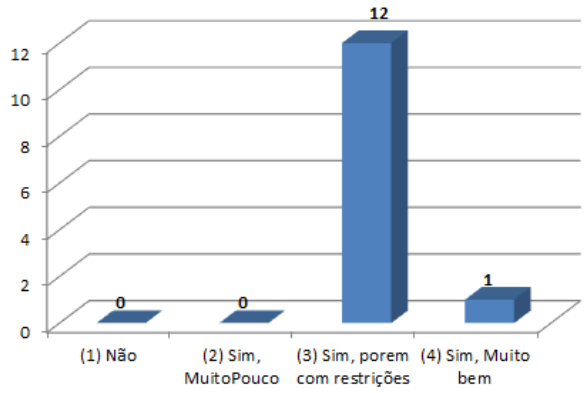
Fonte: Próprio autor.

Na última indagação, a grande maioria dos alunos (57%) afirmou que a utilização do mundo virtual leva à distração, devido à quantidade de informações disponibilizadas, enquanto que 43% apontaram ser mais interessante a utilização do recurso de mundos virtuais como prática na disciplina (Gráfico 07). O resultado observado neste questionamento possui relação direta com a teoria da sobrecarga cognitiva, visto que os alunos estão adaptados a atividades tradicionais e ao migrar para um ambiente imersivo, com uma carga acentuada de cores, tecnologias e outros recursos, a sobrecarga cognitiva pode ser observada.

Visto que este projeto piloto teria um impacto decisivo sobre o encaminhamento da tese, realizou-se, além da observação descritiva, uma análise estatística apurada sobre os dados do instrumento, a fim de formalizar os resultados verificados e garantir uma precisão estatística que apoiasse as afirmações observadas.

A observação dos resultados de duas questões do instrumento respondido pelos alunos, os quais apresentaram indícios positivos em relação a proposta de uso do OpenSim. A primeira das perguntas analisada, neste contexto foi: “Você conseguiu interagir e realizar as atividades disponibilizadas no OpenSim?”. Esta é uma questão com resposta objetiva, com uma escala Likert de 4 pontos, a qual vislumbra que os entrevistados apresentem seu grau de concordância ou discordância (BAKER, 1995), tendo como respostas possíveis os pesos: Sim muito bem (4); Sim com restrições (3); Sim muito pouco (2); Não (1). A partir de uma análise de frequência básica (Gráfico 08), pode-se definir que o grupo apontou, em sua grande maioria (12 estudantes) que interagiu e realizou as atividades de programação propostas durante o experimento, confirmado pelo cálculo da mediana da amostra, a qual apresentou como resultado um valor 3.

Gráfico 8-Efetividade na realização das atividades no Mundo Virtual



Fonte: Próprio autor.

Quadro 8-Resultado da análise de aleatoriedade

Testes de NPar

Teste de sequências

	OSFacilitouAlg
Valor de teste[a]	,62
Casos < Valor de Teste	5
Casos >= Valor de Teste	8
Total de casos	13
Número de Sequências	9
Z	,828
Significância Sig. (2 extremidades)	,408

Fonte: Próprio autor.

A segunda informação a ser avaliada, neste momento, referiu-se a utilização do OpenSim como ferramenta de apoio ao ensino de algoritmos, a qual questionou se as atividades com este recurso auxiliaram positivamente no entendimento dos conteúdos. Neste quesito, 8 alunos responderam “Sim” e 5 estudantes responderam “Não”, sendo estas respostas traduzidas como um critério dicotômico (0,1). A validação desta informação ocorreu por meio de uma análise não paramétrica dos dados, a fim de comprovar a não aleatoriedade do resultado observado (Quadro 08), o que indicou a existência de uma tendência do grupo em suas colocações. O teste de Wald-Wolfowitz (WALD & WOLFOWITZ, 1943), Equação 01, foi o escolhido para precisar tal informação, através da qual conjecturou-se duas hipóteses (H0: amostra aleatória e H1: amostra não aleatória).

(1)

$$\mu = \frac{2N_aN_b}{N} + 1,$$
$$\sigma = \sqrt{\frac{2N_aN_b(2N_aN_b - N)}{N^2(N-1)}}.$$

A realização do teste ocorreu através da ferramenta de análise estatística IBM SPSS[9], a qual apresentou como saída o Quadro 08. Definindo-se o nível de significância em 95% (α = 5%), e com um valor crítico de $Z_{0,05}$ = 1,96, ao observar o resultado de Sig 0,408 > 0,05, conclui-se que se deve rejeitar a hipótese nula, de que a amostra é aleatória. A partir desta observação e corroborando com o resultado do Gráfico 08, sugere-se que os participantes do experimento obtiveram certeza e uma tendência geral na afirmação de que o uso do mundo virtual e dos demais recursos é válido como instrumento no ensino de algoritmos.

A observação das respostas dos indivíduos participantes da pesquisa foi validada por meio do estudo de seu desempenho acadêmico, em avalições[10] imediatamente anteriores (A2)[9] ao experimento e imediatamente posteriores (A3)[9]. A relação destes resultados em comparação a média do grupo de controle também foi investigada, assim como a média geral da turma que participou do experimento e as demais turmas de algoritmos do mesmo semestre letivo. O intuito básico deste exame foi diagnosticar o desempenho da amostra de alunos em relação aos seus pares. O conjunto de variáveis utilizadas nesta inferência é apresentado no Quadro 09, onde tem-se o nome da variável, sua descrição, o número de elementos da amostra (N) e a indicação de normalidade da amostra. Um detalhe importante a ser frisado discorre sobre a diferença da amostra, referente aos participantes do experimento, utilizada nestes testes (12 alunos) em relação à avaliação anterior (13 alunos) – esta diferença é justificada pelo fato de que uma aluna que participou das atividades com o Mundo Virtual e respondeu ao questionário, não realizou as provas A2 e A3, sendo assim descartada desta fase de testes.

Quadro 9-Variáveis utilizadas na análise estatística

Variável	Descrição	N	Normalidade
A2Exp	Conjunto de notas da segunda avaliação semestral da disciplina de algoritmos (A2) – dos alunos participantes do experimento.	12	Normal

[9] http://www-01.ibm.com/software/analytics/spss/products/statistics/ - "*IBM SPSS Statistics is an integrated family of products that addresses the entire analytical process, from planning to data collection to analysis, reporting and deployment.*"

[10] A disciplina de Algoritmos e Programação, que serviu de base para o experimento, possui em seu Plano de Ensino quatro avaliações distintas, uma a cada momento do semestre (contendo conteúdos específicos). As avaliações são denominadas neste Plano como A1 (primeira avaliação), A2 (segunda avaliação), A3 (terceira avaliação) e AR (avaliação de recuperação).

A3Exp	Conjunto de notas da terceira avaliação semestral da disciplina de algoritmos (A3) – dos alunos participantes do experimento.	12	Normal
A3NExp	Conjunto de notas da terceira avaliação semestral da disciplina de algoritmos (A3) – Grupo de Controle.	21	Normal

Fonte: Próprio autor.

Para a exploração das amostras foi utilizado o Teste T de Student (BOX, 1987), Equação 02, um teste baseado em conceitos estatísticos, utilizado para a identificação de hipóteses válidas, o qual possui um alto grau de precisão para amostras com distribuição normal, especialmente se apresentarem dimensões menores que trinta (n<30).

(2)

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} \qquad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

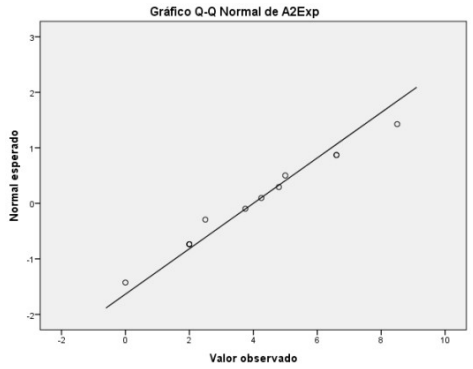
Comparou-se as médias A2Exp e A3Exp através do Teste T, contudo inicialmente foi validado a normalidade da amostra, por meio do teste de Shapiro-Wilk (SHAPIRO & WILK, 1965; ROYSTON, 1992; RAZALI & WAH, 2011; DOUFOUR *et al.* 1998), Equação 03, o qual comprovou a normalidade de ambas amostras, apresentando um Sig para A2Exp=0,848 e para A3Exp=0,189. Desta forma é possível afirmar que ambas variáveis apresentam Sig > 0,05, demonstrando o comportamento normal, para um nível de confiança maior que 0,05, validando a utilização do teste paramétrico proposto.

(3)

$$W = \frac{b^2}{\sum_{i=1}^{n}(x_{(i)} - \bar{x})^2} \qquad b = \begin{cases} \sum_{i=1}^{n/2} a_{n-i+1} \times (x_{(n-i+1)} - x_{(i)}) \text{ se n é par} \\ \sum_{i=1}^{(n+1)/2} a_{n-i+1} \times (x_{(n-i+1)} - x_{(i)} \text{ se n é ímpar} \end{cases}$$

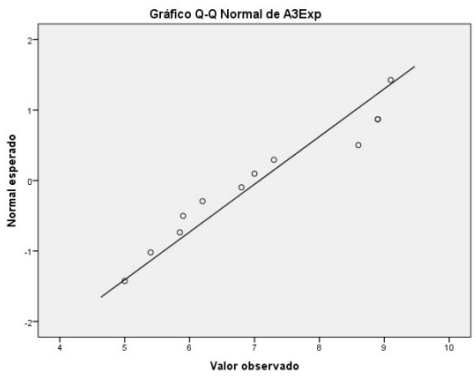
Os Gráficos 09 e 10, denominados Gráficos Quantil-Quantil ou QQ (RAZALI & WAH, 2011; BARBETTA *et al.* 2004), demonstram a validação da distribuição de probabilidade normal das amostras, visto que os pontos no gráfico se adequam a reta posta.

Gráfico 9-Teste de normalidade para A2Exp



Fonte: Próprio autor.

Gráfico 10-Teste de normalidade para A3Exp



Fonte: Próprio autor.

A média aritmética simples das amostras (A2Exp e A3Exp) já indicavam fortes indícios da existência de uma diferença válida entre as notas dos alunos na avaliação anterior e posterior ao experimento. A2Exp com média 4,00 e A3Exp com 7,07, apontaram para uma melhora significante no desempenho. A confirmação desta afirmação é realizada com a utilização do Teste T, para amostras pareadas, através do qual se definiu duas hipóteses, H0 onde a média da segunda avaliação é igual à média da terceira avaliação (H0: μA2Exp=μA3Exp) e, H1 em que a média da terceira avaliação é maior que a média da segunda (H1: μA3Exp>A2Exp). O resultado alcançado com esta observação (Quadro 10) comprovou a rejeição da hipótese nula, visto que o grau de significância demonstrado pelo método apresenta um valor de 0,002 nas duas extremidades. Desta forma assumiu-se a hipótese alternativa (H1: μA3Exp>A2Exp) como válida. Esta primeira intervenção demonstrou que o experimento alcançou resultados positivos, corroborando com a impressão demonstrada pelos alunos no instrumento de pesquisa. Contudo para consolidar tal afirmação foi necessário comparar as notas dos alunos do grupo de controle (A3NExp) com as notas dos alunos participantes do experimento (A3Exp).

Quadro 10-Teste T de Student (amostras pareadas) para A2Exp e A3Exp

Teste de amostras emparelhadas

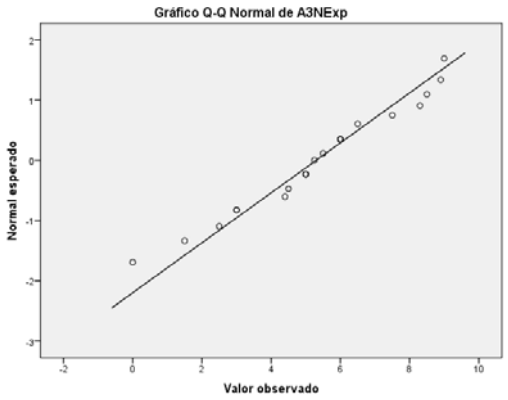
		Diferenças emparelhadas					t	df	Sig. (2 extremidades)
		Média	Desvio Padrão	Erro padrão da média	95% Intervalo de Confiança da Diferença				
					Inferior	Superior			
Par 1	A2Exp - A3Exp	-3,07917	2,56820	,74138	-4,71092	-1,44741	-4,153	11	,002

Fonte: Próprio autor.

Assumindo que os alunos participantes do experimento obtiveram uma melhora em suas avaliações (A2Exp < A3Exp), partiu-se para comparação das suas médias com as médias do grupo de controle (A3NExp). Para este caso definiu-se duas hipóteses, H0 que define a média de A3 do experimento igual a média de A3 do grupo de controle (H0: μA3Exp=μA3Nexp) e, H1 em que a média de A3 do experimento maior que A3 do grupo de controle (H1: μA3Exp>μA3NExp). Com o intuito de validar tais pressuposições foi realizado o Teste T para amostras independentes (baseado na diferença dos universos de A3Exp e A3NExp). Inicialmente também foi verificado, por meio de Shapiro-Wilk, o grau de normalidade de A3NExp, obtendo-se a confirmação de normalidade (0,590 > 0,05), conforme é apresentado no Gráfico 11.

A análise de médias, inicialmente, apontou para valores distintos entre os grupos (A3Exp=7,07 e A3NExp=5,30). Na etapa seguinte, avaliou-se o resultado para o Teste de Levene (LEVENE, 1960; SCHULTZ, 1985; LIM & LOH, 1996), que permite averiguar a homogeneidade das variâncias de ambos os grupos (A3Exp e A3NExp), mostrando que tais variâncias eram semelhantes, descrito por meio de uma significância associada ao teste, superior a 0,05 (0,216>0,05) – Quadro 11.

Gráfico 11-Teste de normalidade para A3NExp



Fonte: Próprio autor.

Quadro 11-Teste de homogeneidade de variâncias para grupos (A3Exp e A3NExp)

		Teste de Levene para igualdade de variâncias	
		Z	Sig.
NotasComp	Variâncias iguais assumidas	1,597	,216
	Variâncias iguais não assumidas		

Fonte: Próprio autor.

O resultado do Teste de Levene permitiu a observação do Teste T para as igualdades de média, através da análise de significância de 2 extremidades, avaliando a linha “Variância iguais assumidas”. O valor obtido, neste caso, foi de 0,028 com (0,028<0,05), demonstrando que existem evidências estatísticas significativas entre as médias dos grupos (Quadro 12), o que aponta para o descarte da hipótese nula (H0), comprovando que as notas de A3Exp foram

maiores (μA3Exp>μA3NExp), ou seja, é provado que os estudantes que participaram do experimento obtiveram um melhor rendimento em A3 do que os alunos do grupo de controle.

Quadro 12-Teste T de Student (amostras independentes) para A3Exp e A3NExp

Teste de amostras independentes

		teste-t para Igualdade de Médias						
		t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	95% Intervalo de Confiança da Diferença	
							Inferior	Superior
NotasComp	Variâncias iguais assumidas	2,308	31	,028	1,77679	,76993	,20651	3,34707
	Variâncias iguais não assumidas	2,624	30,772	,013	1,77679	,67726	,39510	3,15847

Fonte: Próprio autor.

Como última observação, realizou-se uma análise entre as médias finais das notas dos alunos de outras turmas de algoritmos (do mesmo período), aqui denominadas ABC (turma A, turma B e Turma C) em comparação às médias finais obtidas pelos alunos participantes do experimento (MedExp). Os dados amostrais estão apresentados no quadro 13.

Quadro 13-Variáveis utilizadas para comparação entre as médias dos grupos

Variável	Descrição	N	Média Notas
MedExp	Conjunto de notas referentes a média final dos alunos participantes do experimento.	12	6,87
ABC	Conjunto de notas dos alunos de outras turmas de algoritmos, representado por:	110	4,09

Turmas	Quantidade alunos	Média Final
A	**20**	**4,02**
B	**43**	**3,53**
C	**47**	**4,71**

Fonte: Próprio autor.

Para análise destas médias realizou-se o Teste T de Student (amostras independentes), em que foram definidas as hipóteses H0 com a média geral do experimento igual a média das turmas A,B e C (H0: $\mu MedExp=\mu ABC$) e, H1 onde a média do experimento é maior que a média das turmas A,B e C (*H1:* $\mu MedExp>\mu ABC$). O Teste de Levene para estes grupos apresenta a ausência de homogeneidade de variância (Quadro 14), com um Sig 0,008 (Sig < 0,05), assumindo a linha "Variâncias iguais não assumidas", que por sua vez apresenta um Sig 0,000 (Quadro 15), confirmando (0,000<0,05), ou seja, é possível estatisticamente afirmar que os alunos participantes do experimento alcançaram uma média maior que os demais (H1 valida), rejeitando-se a hipótese nula (H0).

Quadro 14-Teste de homogeneidade de variâncias para grupos (MedExp e ABC)

		Teste de Levene para igualdade de variâncias	
		Z	Sig.
Notas	Variâncias iguais assumidas	7,268	,008
	Variâncias iguais não assumidas		

Fonte: Próprio autor.

Quadro 15-Teste T de Student (amostras independentes) para MedExp e ABC

Teste de amostras independentes

		teste-t para Igualdade de Médias						
		t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	95% Intervalo de Confiança da Diferença	
							Inferior	Superior
Notas	Variâncias iguais assumidas	3,820	120	,000	2,74394	,71836	1,32163	4,16625
	Variâncias iguais não assumidas	5,609	18,159	,000	2,74394	,48918	1,71686	3,77102

Fonte: Próprio autor.

Neste ponto do estudo foi possível afirmar, com a gama de observações realizadas a partir das diferentes metodologias de análise sobre os resultados alcançados, que a mudança de paradigma nos processos e nas ferramentas utilizadas em uma disciplina de algoritmos e programação podem aprimorar a capacidade dos alunos em entender os conteúdos, além de estimular o estudante a resolver problemas de forma individual e com o auxílio de seus pares.

Ao final desta etapa do experimento piloto foi possível apontar conclusões pertinentes para a continuidade da pesquisa. A prática pedagógica adotada se baseou inicialmente na introdução ao Mundo Virtual, por meio da qual os alunos puderam identificar todas as funcionalidades do ambiente (manipulação/configuração dos avatares, dos objetos, comunicação via *chat* com outros participantes e reconhecimento da ilha no OpenSim). Após este momento, os estudantes foram motivados a se conectar ao Sloodle, a fim de identificar as tarefas a serem realizadas utilizando o IDEOne. Visando identificar características colaborativas de trabalho, a turma foi separada em duplas, as quais foram impelidas a trocar informações via *chat* com outras duplas e observar, nos painéis dentro do próprio Mundo Virtual, as resoluções enquanto construídas por seus colegas.

Os resultados obtidos demonstraram uma perceptível disponibilidade dos alunos em utilizar novos recursos tecnológicos durante as aulas de programação, além do destaque em relação aos aspectos motivacionais observados, os quais são sintomáticos às características de inovação, interação e imersão proporcionadas pelo mundo virtual. Na avaliação da tecnologia, tanto o metaverso quanto as demais aplicações utilizadas, foram suportadas pelos sistemas computacionais que as aportaram e, assim atenderam às demandas de processamento necessárias para a renderização e disponibilização do mundo virtual aos alunos. Todos estes aspectos enumerados ratificam a viabilidade de uso do OpenSim em práticas pedagógicas para o ensino de algoritmos e, desta forma, como instrumento base para as atividades e proposições deste projeto.

Por fim, destacou-se o alto grau de interação demonstrado entre os alunos durante o experimento, característica do aspecto colaborativo da atividade, o qual permitiu a participação dinâmica dos estudantes na resolução dos problemas propostos, o que ficou claro através do resultado apresentado no Gráfico 06, que apontou a comunicação no ambiente como um dos elementos mais atrativos da prática. As perspectivas dos alunos na resolução de um problema em comum, o envolvimento mútuo, partilha, iniciativa conjunta, estímulo cognitivo e a capacidade de interagirem dentro do metaverso corroboraram com o princípio da colaboração defendido por Harasin (1997) e Rogers (2000).

4.2. PROJETO PILOTO – FASE 02

A segunda etapa do projeto piloto caracterizou-se pela complexidade nas ações envolvidas para a elaboração dos experimentos e observações. O conhecimento construído no momento anterior (fase 01) permitiu a evolução dos estudos sobre novas práticas para o ensino de algoritmos, o que culminou na proposição de um ensaio com foco em uma estrutura inovadora, a ser aplicada em uma disciplina de algoritmos.

O objetivo principal com as ações planejadas e executadas nesta fase foi a busca da melhora dos índices de aprovação dos alunos na disciplina, por meio de uma reformulação do plano de ensino, práticas pedagógicas e instrumentos comumente adotados em uma aula tradicional. O foco de tais alterações esteve na adaptação dos conteúdos básicos de algoritmos para a inserção de uma mudança de paradigma no que é praticado atualmente, com a proposição da programação orientada a blocos visuais, *scripts* e integração ao mundo virtual (OpenSim).

Para alcançar o escopo deste trabalho, buscou-se um processo diferente do instrucionista, em que o computador é utilizado apenas como uma máquina de ensinar, adaptado ao ensino tradicional. Disponibilizou-se aqui um modelo construtivista, em que o estudante, a partir da interação com o meio, foi imbuído a construir o conhecimento sobre lógica e algoritmos, utilizando os sistemas de programação em blocos visuais e recursos de mundos virtuais.

O atendimento deste objetivo é realizado seguindo a linha de Papert (1986), visto que este autor remodelou os princípios do construtivismo de Piaget (1969), implementando um

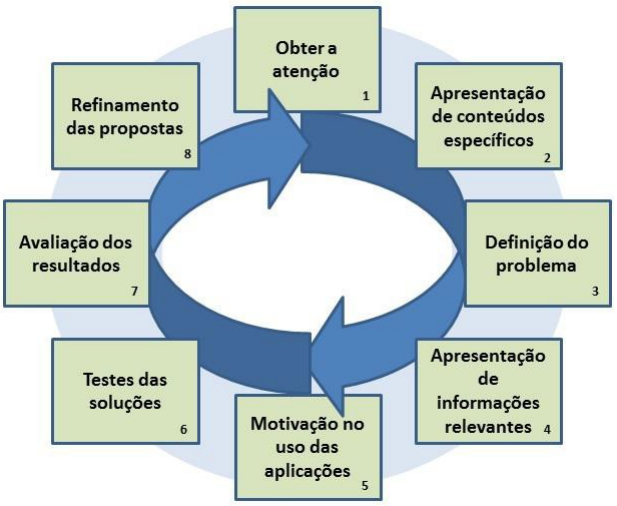
conjunto proposições para a utilização do computador como instrumento de apoio ao processo de ensino e de aprendizagem, o qual foi denominado construcionismo, que em síntese designa a modalidade em que um discente utiliza o computador como uma ferramenta com a qual constrói seu conhecimento, elaborando um objeto de seu interesse, como um relato de experiência ou um programa de computador.

Neste contexto ainda, é importante reconhecer que, para ser efetivo o modelo deve construir uma interpretação significativa (JONASSEN, 1992), visto que a utilização dos ambientes computacionais naturalmente induz a um progresso nas tecnologias de aprendizagem, permeando do crescimento cognitivo até a aprendizagem significativa (JONASSEN, 1993).

A proposta foi estimular o aprendiz a desenvolver soluções de seu interesse, focando nos níveis motivacionais que podem ser alcançados com o uso destes sistemas. A prática do construcionismo ocorreu com o fomento da capacidade do aluno em implementar programas, incentivados pelo seu interesse de interagir com as aplicações Scratch, Scratch4OS e OpenSim, buscando desta forma um alto nível de envolvimento e atingindo uma aprendizagem significativa, através da motivação e conexão dos novos conteúdos, adquiridos a cada etapa do estudo, com conhecimentos prévios (subsunçores) alcançando assim uma mudança na sua estrutura cognitiva relacionada a programação. A motivação necessária deu-se pela possibilidade de simulação aberta, inerente a estas aplicações, pois possibilitaram por meio da simulação o desenvolvimento de hipóteses, testes e ajustes dos conceitos observados, assim como proposto por Valente (1997).

Alcançou-se a abordagem pedagógica construcionista pela disponibilização aos alunos de um conjunto de conteúdos, materiais e estratégias, permitindo a estes estudantes integrar e correlacionar o conjunto de informações e, a partir disto propor soluções computacionais para os problemas apresentados durante o projeto piloto. Estas ações estão apresentadas em oito etapas distintas, conforme disposto na Figura 21.

Figura 21–Abordagem adotada

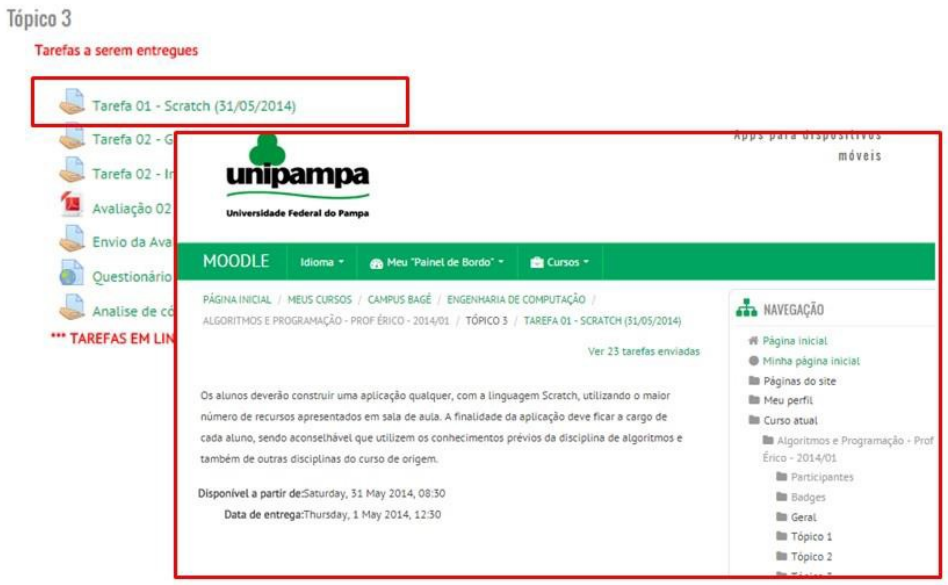


Fonte: Próprio autor.

A proposta desta organização foi abordar um conjunto de teorias de aprendizagem reconhecidas, como:

- O ciclo experiencial de Kolb (KOLB, 1984), contemplando a experiência concreta (etapas 1 a 3), observação reflexiva (etapa 4), conceitualização abstrata (etapa 5) e experimentação ativa (etapas 6 a 8);
- A aprendizagem significativa (AUSUBEL, 1963) também foi elencada neste modelo, nas etapas 1 e 2, onde ocorrem apreensão da atenção do aluno e apresentação de conteúdos específicos (menores), os quais devem estar relacionados à sua bagagem de conhecimento (subsunçores), fazendo com que o estudante aprenda significativamente, por meio da transformação dos conceitos e ideias de sua estrutura mental e, desta forma acessar, relacionar e aprimorar os novos conteúdos disponibilizados. Um exemplo deste tipo de ação é apresentado na Figura 22, uma tarefa realizada em sala de aula e disponibilizada no Moodle, onde os alunos deveriam implementar uma aplicação qualquer, utilizando o maior número possível de recursos da linguagem e os conhecimentos prévios de outras disciplinas dos seus cursos de origem;

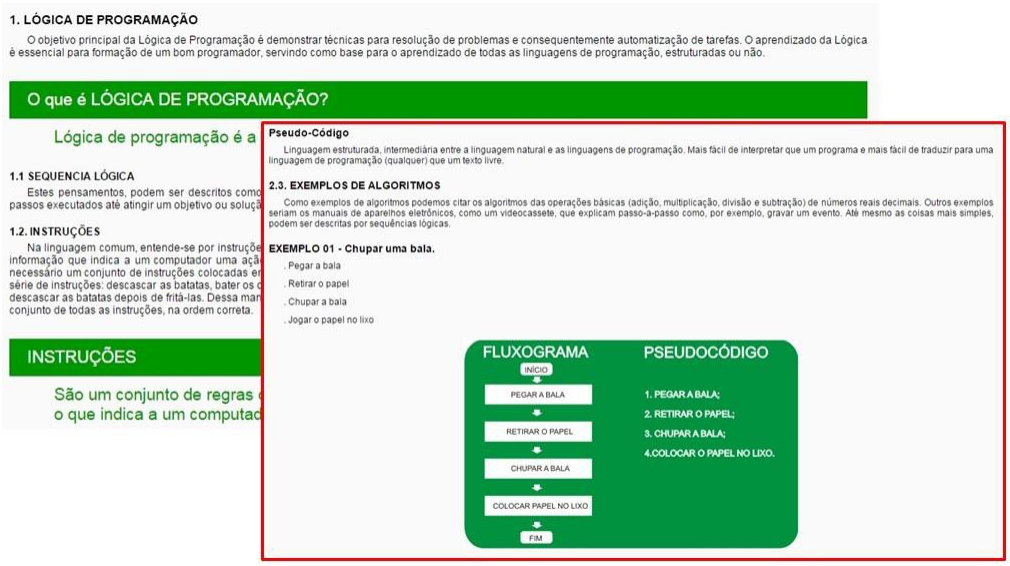
Figura 22-Aprendizagem significativa



Fonte: Próprio autor.

- A estruturação e sequência dos conteúdos foram adequadas, em todas as etapas desta abordagem, respeitando o principio da complexidade de Wiley (2014) e a teoria da carga cognitiva (SWELLER, 1998), com a composição dos materiais de forma simples e atraente, integrando textos, imagens e animações em um agrupamento visual, com o foco na facilidade do entendimento e interpretação das informações transmitidas, de acordo com o proposto por Miller (1956). Um exemplo disto é demonstrado na organização do material sobre introdução à lógica de programação e variáveis, Figura 23.

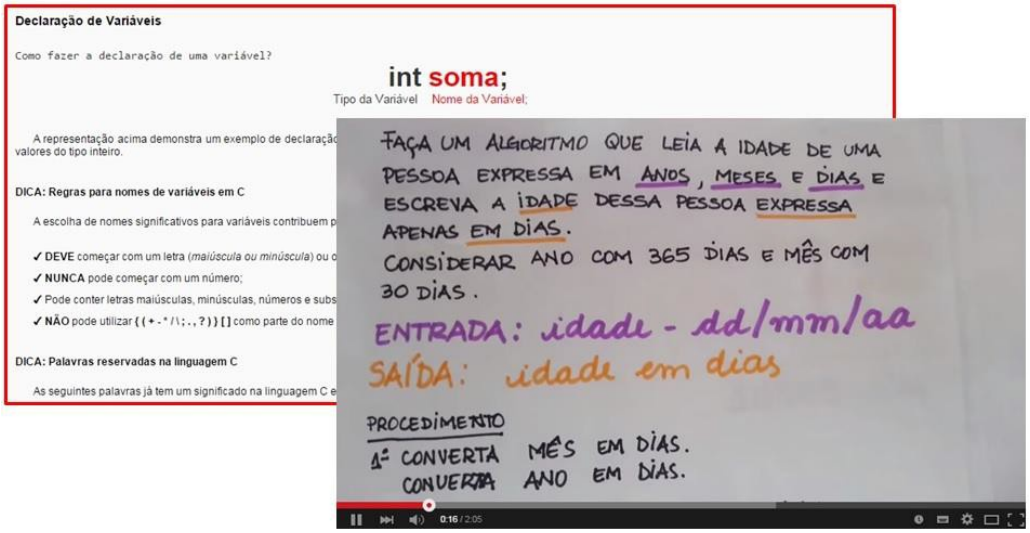
Figura 23–Material de Introdução a Lógica de Programação



Fonte: Próprio autor.

- Com o intuito de evitar a sobrecarga cognitiva (MAYER, 2001; SWELLER, 2003), houve um maior cuidado com a elaboração de materiais nas etapas 2, 3 e 4, em que uma quantidade significativa de dados era disponibilizado aos alunos. Os princípios da representação múltipla e proximidade espacial foram atendidos com a apresentação dos conteúdos através de vídeos e textos ilustrados (Figura 24). A proximidade temporal foi alcançada pela utilização de textos animados, em que ilustrações e redação eram apresentadas simultaneamente. Para atender às diferenças individuais dos alunos, foram elaborados diferentes materiais sobre o mesmo conteúdo, (apostilas, animações, vídeos) permitindo ao estudante escolher o que melhor se adeque ao seu estilo cognitivo. Por fim, o princípio da coerência foi contemplado na composição de todos os conteúdos, através da busca de diferentes recursos multimídia, contudo implementados de forma simples e pontual ao tema abordado. A coerência é alcançada nesta etapa, por meio do descarte de materiais estranhos ao assunto discutido, como palavras, sons ou imagens, não relevantes. A Figura 24 demonstra o conteúdo sobre variáveis em um tutorial e o vídeo sobre o mesmo conteúdo, descrevendo o passo-a-passo para utilização de uma variável.

Figura 24–Estudo sobre variáveis



Fonte: Próprio autor.

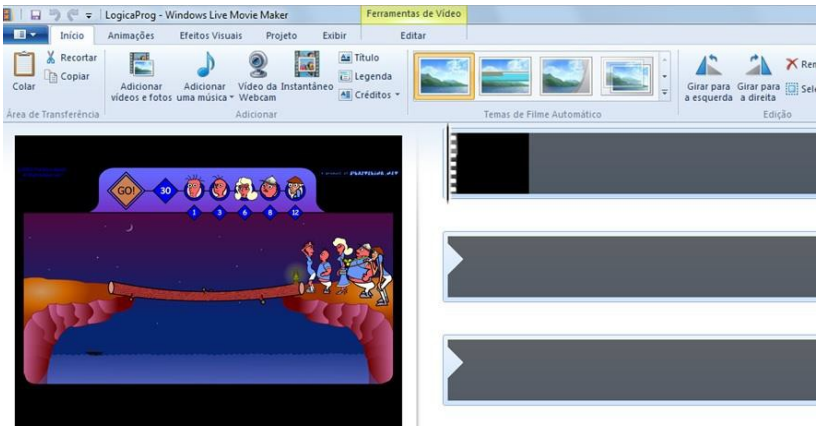
Já a Figura 25 se refere a materiais utilizados em sala de aula e que se apropriam dos princípios de proximidade espacial, temporal e coerência.

Figura 25–Princípios de proximidade espacial, temporal e coerência

Fonte: Próprio autor.

- Aspectos motivacionais também foram o foco desta abordagem, inseridos em todas suas etapas, materiais interativos como vídeos e animações buscaram aproximar os estudantes aos conteúdos apresentados, estimulando o processo de aprendizagem (MAYER & MORENO, 2002; VALENTE, 1997; KAMF & DIAS, 2003). Neste contexto foram utilizadas diferentes aplicações como Flash[11], Camtasia Studio [12] e Windows Live Movie Maker[13] para a composição dos vídeos educacionais (Figura 26).

Figura 26–Edição de um vídeo explicativo sobre Lógica



Fonte: Próprio autor.

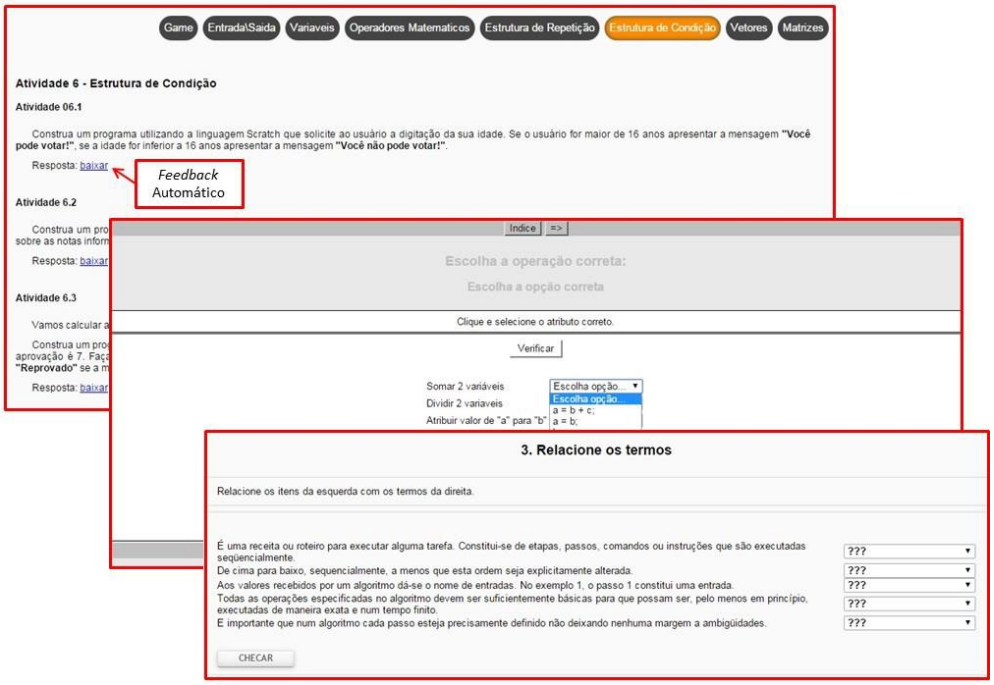
[11] http://www.adobe.com/br/products/flashplayer.html

[12] https://www.techsmith.com/camtasia.html

[13] http://windows.microsoft.com/pt-br/windows-live/movie-maker

Através do HotPotatoes[14] foram implementados exercícios interativos, com *feedback* automático (MOREIRA & FAVERO, 2009; MUSSOI *et al*. 2010), permitindo ao aluno identificar imediatamente seus erros e desta maneira corrigir eventuais falhas na sua aprendizagem (Figura 27).

Figura 27–Exercícios com *feedback* automático



Fonte: Próprio autor.

Este ciclo buscou realizar a formação básica do estudante de forma interativa e usando recursos multimídia para cativar sua atenção e aumentar a motivação para uma área de estudo que é tradicionalmente conhecida como árida e abstrata. Algoritmos tem esta reputação entre os estudantes e isto é responsável pelos problemas de reprovação e alta evasão apontados no início deste estudo. Do sucesso desta capacitação depende o crescimento cognitivo do estudante com vistas a tornar-se apto a resolver problemas mais complexos que são encontrados no extremo superior da aprendizagem de algoritmos e linguagens de programação.

A adoção desta abordagem permitiu também alinhar o processo de ensino à proposta da teoria de aprendizagem experiencial (KOLB, 1984) com o foco na concentração de conhecimentos e respostas aos estímulos gerados pelo ambiente de ensino. Desta maneira, a

[14] https://hotpot.uvic.ca/

organização dos materiais e conteúdos vislumbrou promover uma experiência concreta ao discente, por meio do contato com novos assuntos e práticas, motivando a reflexão sobre as ações e conceitos, levando o estudante a criar suas conclusões (aprender), possibilitando assim o planejamento e experimentação ativa dos novos conhecimentos, como base para resolução de outros problemas.

Ao se observar as etapas propostas foi possível verificar a relação direta com a dinâmica de todas as teorias utilizadas como alicerce da construção desta tese, onde tem-se o conhecimento como um processo contínuo de integração de experiências e conceitos. Todo material didático desenvolvido nesta etapa foi utilizado durante todas as demais fases deste estudo (Projeto Piloto 1,2,3 e no Experimento Final), com pequenas alterações durante a evolução dos experimentos.

Esta fase do projeto piloto teve a duração de um semestre (2013/2), com um grupo de 60 alunos previamente matriculados na disciplina “Algoritmos e programação” da Universidade Federal do Pampa. Deste total o trabalho pode ser efetivamente realizado com uma amostra de 33 discentes, visto que 27 estudantes nunca compareceram às aulas (esta taxa de evasão inicial é frequente em instituições públicas similares à Universidade onde o experimento foi realizado). Estes discentes eram oriundos dos cursos de Engenharia de Computação (13), Engenharia de Energias Renováveis e Ambiente (4), Engenharia Química (5), Engenharia de Produção (5), Engenharia de Alimentos (1), Licenciatura em Matemática (2), Licenciatura em Matemática (2) e Licenciatura em Física (3).

A primeira ação na elaboração deste experimento foi a reorganização do Plano de Aula da disciplina, para que a mesma contemplasse as novas práticas propostas. O plano tradicional possuía uma carga horária de 60 horas, com a Linguagem C como ferramenta padrão de desenvolvimento. A ementa da disciplina contempla os seguintes temas: Conceito de algoritmo, partes do algoritmo, atribuição e operações, entrada e saída, estruturas de condição, estruturas de repetição, vetores, matrizes. Subalgoritmos: procedimentos e funções. É tarefa do professor atender a todos estes assuntos de forma ampla durante o semestre. A reformulação concebida modificou sensivelmente esta organização, com o reagrupamento dos conteúdos e sua abordagem transferida para o período inicial do semestre, sendo denominado nesta pesquisa como Modelo 1, conforme pode ser observado no infográfico da Figura 28.

Figura 28–Infográfico da nova estrutura para o plano de ensino (Modelo 1) – Projeto Piloto Fase 2

Fonte: Próprio autor.

O conjunto de atividades do experimento e seu posicionamento cronológico na estrutura do Plano de Ensino foram definidos com base na concepção de que os estudantes, neste momento, portariam uma bagagem inicial sobre programação, o que possibilitaria um reforço na construção do conhecimento sobre o Pensamento Computacional e a lógica para o desenvolvimento de soluções computacionais com programas. É importante salientar que a alocação dos novos temas do plano de ensino não alterou a carga horária prevista para a disciplina.

Com relação à metodologia de avaliação dos alunos, no plano tradicional estavam previstas quatro avaliações, organizadas como segue no Quadro 16.

Quadro 16–Organização das avaliações no plano de ensino tradicional

Avaliação	Conteúdo
Avaliação 01	Conceito e definição de algoritmos, estudo de problemas Identificação dos valores de entrada e de saída Noções de lógica Conceito de variável, tipos de variáveis Comandos de entrada e saída Operadores (aritméticos e atribuição) e Estruturas Condicionais Estruturas de condição e operadores (relacionais e lógicos)
Avaliação 02	Estruturas de repetição Teste de mesa Vetores Matrizes Funções auxiliares (strings)
Avaliação 03	Funções, variáveis (locais e globais) e passagem de parâmetros Procedimentos e passagem de parâmetros
Avaliação 04 Recuperação	Todo conteúdo da disciplina

Fonte: Próprio autor.

Quadro 17–Organização das avaliações com a nova proposta (Modelo 1 - Experimento)

Avaliação	Conteúdo	
Avaliação 01	Conceito e definição de algoritmos, estudo de problemas Identificação dos valores de entrada e de saída Noções de lógica Conceito de variável, tipos de variáveis Comandos de entrada e saída Operadores (aritméticos e atribuição) e Estruturas Condicionais Estruturas de condição e operadores (relacionais e lógicos)	
Avaliação 02	Estruturas de repetição Teste de mesa Vetores Matrizes Funções auxiliares (strings)	
Avaliação 03 **Conteúdos Propostos**	Atv 01	Conhecimentos básicos sobre Scratch
	Atv 02	Variáveis, comandos de entrada e saída operadores aritméticos, estrutura de condição/repetição
	Atv 03	Vetores e matrizes
	Atv 04	Implementando soluções para o Mundo Virtual com LSL
Avaliação 04	Funções, variáveis (locais e globais) e passagem de parâmetros Procedimentos e passagem de parâmetros	
Avaliação 05 **Recuperação**	Todo conteúdo da disciplina: Utilização de todos paradigmas estudados em aula (C, Scratch, Scratch4OS, LSL)	

Fonte: Próprio autor.

No Quadro 17 é apresentada a organização das avaliações para o semestre. Uma nova atividade é introduzida (Avaliação 03), a qual contempla todos os conteúdos estudados sobre os novos paradigmas e o ambiente imersivo. Esta verificação possui o mesmo peso das demais provas, sendo constituída pelas atividades (01, 02, 03,04), que correspondem a 25%, cada uma, concretizando desta forma o montante da nota referente à Avaliação 03.

A evolução do semestre ocorreu de maneira semelhante aos demais, contudo diferentes teorias foram agregadas na elaboração de cada encontro. No entanto, todas as práticas foram estruturadas com base no ciclo da aprendizagem experiencial (KOLB, 1997), através da qual buscou-se:

- Com a experiência concreta apresentar os novos conceitos de lógica e programação e, a partir disto exercitar a utilização dos mesmos;
- Com a observação reflexiva traçar uma linha de raciocínio, relacionando os conhecimentos prévios aos novos conceitos aprendidos;
- A conceitualização abstrata deu-se pela compreensão/assimilação destes conceitos e a identificação do seu correto emprego na resolução dos problemas propostos;
- A experimentação ativa calcou-se na realização de práticas relacionadas ao desenvolvimento de algoritmos e construção de programas, utilizando os novos conhecimentos adquiridos. Estas ações viabilizaram aos discentes identificar erros

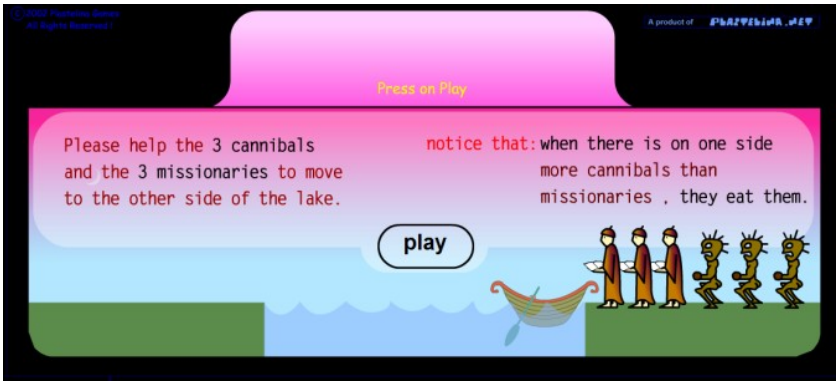
cometidos anteriormente, elevando o nível de precisão das soluções algorítmicas elaboradas.

Estas ações foram incorporadas em todos os momentos didáticos, desde o início da disciplina, contemplando a totalidade dos assuntos abordados, desde a lógica até o uso dos mundos virtuais.

Algumas das ações introduzidas nas atividades de aula seguem:

- Atividade de nivelamento, com o intuito de equiparar a base de conhecimento dos estudantes;
- Utilização de animações e vídeos com a intenção de estimular o raciocínio lógico dos discentes. Exemplo desta prática é apresentado na Figura 29, em um exercício utilizando a tecnologia em flash para resolver o problema de travessia de missionários e canibais, de um curso de água.
- Construção de fluxogramas e pseudocódigos para atividades cotidianas de cada aluno, forçando a concepção de modelos que pudessem ser interpretados por seus colegas;
- Implementação de organogramas que evidenciassem todas as fases necessárias para a compilação de um determinado programa, a fim de concretizar o entendimento da formalidade necessária para a escrita de um programa em qualquer linguagem de programação.

Figura 29–Problema de lógica interativo



Fonte: Próprio autor.

Nos encontros em que foram apresentados o Scratch, Scratch4OS, OpenSim e a linguagem LSL, as práticas seguiram critérios mais dinâmicos, com a apresentação das possibilidades de criação a partir de tais tecnologias. No primeiro contato com o Scratch, os estudantes foram desafiados a construir um jogo. Após o entendimento sobre a concepção da ferramenta, o objetivo das aulas foi direcionado à integração dos conhecimentos de

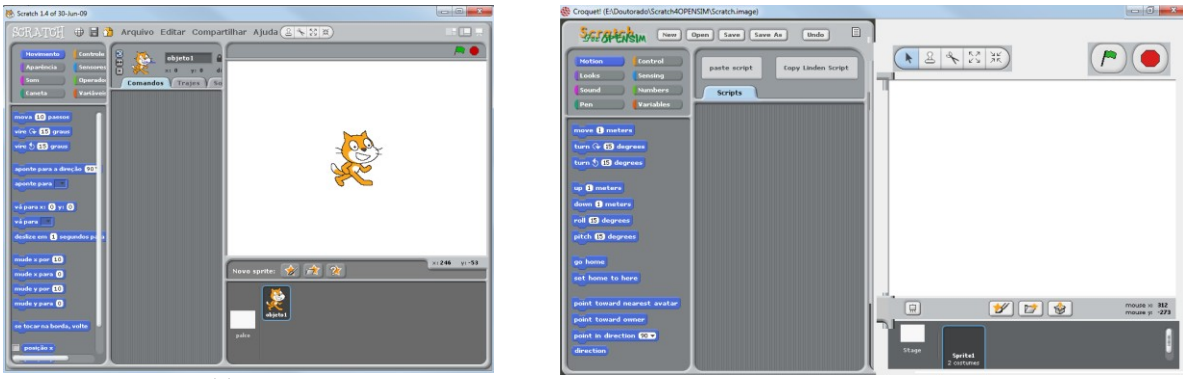
programação já adquiridos, a fim de explorar o Scratch como uma linguagem de programação e desenvolver aplicações para problemas reais. Com este intuito foi concebida uma apostila, Figura 30, alinhando os conteúdos previstos na disciplina de algoritmos (Linguagem C) com a estrutura de comandos do Scratch, possibilitando a percepção de como utilizar variáveis, operadores aritméticos, estruturas de condição/repetição, vetores e matrizes em uma linguagem orientada a blocos de programação visual.

Figura 30–Guia de Referência Scratch

Fonte: Próprio autor.

Após as práticas de programação com Scratch, foi apresentado aos alunos o Scratch4OS, uma ferramenta de desenvolvimento semelhante ao Scratch, porém com a funcionalidade de fornecer os códigos em linguagem LSL, dos programas elaborados na sua interface. Estes códigos fontes podem ser agregados a objetos no Mundo Virtual, os quais passam a ter as funcionalidades definidas pelo programa. A Figura 31 mostra uma comparação entre as interfaces do Scratch e Scratch4OS.

Figura 31–Comparação entre as interfaces do (a) Scratch e (b) Scratch4OS

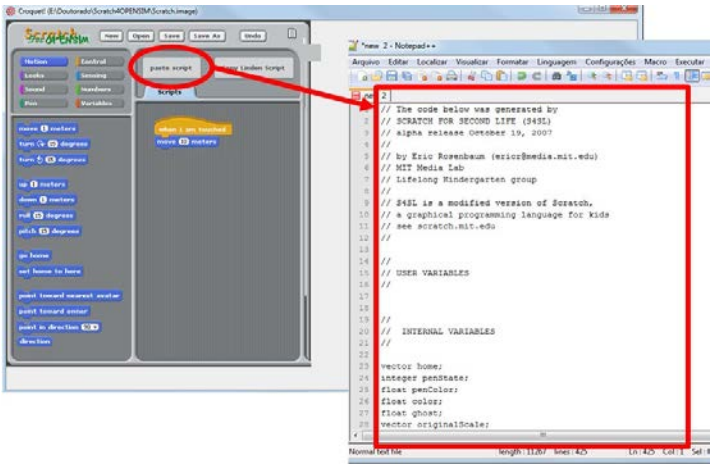


(a) **(b)**

Fonte: Próprio autor.

Na Figura 32 é demonstrado um trecho de código fornecido pelo Scratch4OS, de um programa básico, para a movimentação em dez metros de um determinado objeto.

Figura 32–Obtenção do código fonte em LSL no Scratch4OS



Fonte: Próprio autor.

A introdução ao Mundo Virtual OpenSim, ocorreu da mesma forma que na fase 01 do projeto piloto, iniciando pela configuração dos avatares e entendimentos dos objetos que compõem o mundo. Com entendimento de que os programas desenvolvidos no Scratch4OS poderiam ter seu comportamento repassado para tais objetos do OpenSim, os alunos foram orientados e estimulados a testar todos os códigos programados em aula, nos MV, conforme mostra a Figura 33.

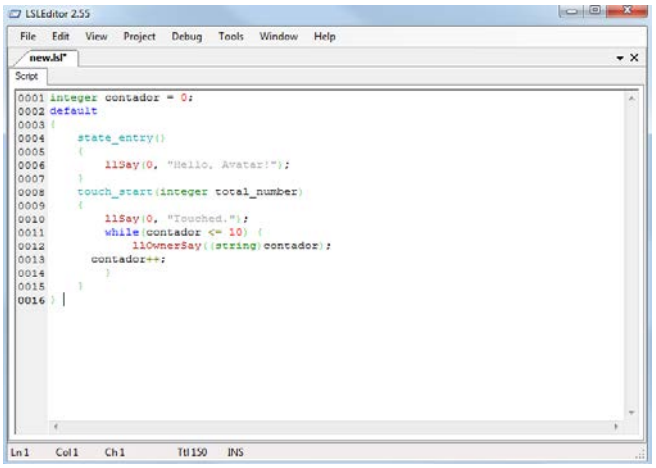
Figura 33–Prática no MV sobre a inserção de códigos do Scratch4OS

Fonte: Próprio autor.

A partir da compreensão da interação no MV, os estudantes passaram a estudar a estrutura de programação com a Linguagem LSL, a fim de programar ações e estados diretamente nos objetos. Para isso foi realizada em sala de aula uma atividade específica, focada no alinhamento dos conhecimentos sobre programação em linguagem C e Scratch com a linguagem LSL, isto posto, os estudantes puderam refletir como utilizar a estrutura/sintaxe

do LSL e vislumbrar possibilidades na criação de soluções para o OpenSim. Na programação foi utilizado um editor LSL (LEC, 2014), uma IDE para o desenvolvimento em linguagem LSL, por meio da qual é possível visualizar o esquema de cores durante a construção de código (Figura 34).

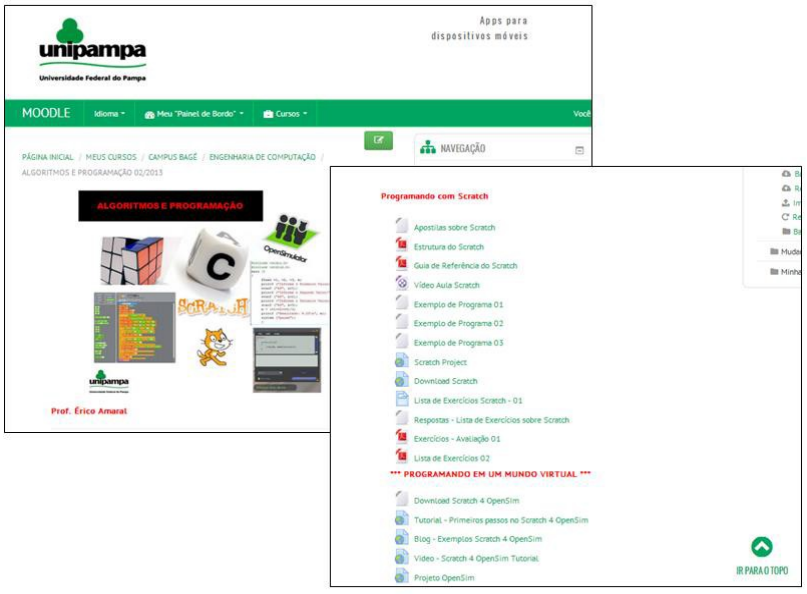
Figura 34–Programando com o LSL Editor



Fonte: Próprio autor.

Para um contato mais direto com os alunos, além dos momentos presenciais, foi instituída como forma de apoio uma sala de aula virtual, com o AVA Moodle (Figura 35), através da qual se disponibilizou todos os materiais referentes às atividades realizadas, além de servir como repositório para o envio das tarefas e acesso aos conteúdos auxiliares.

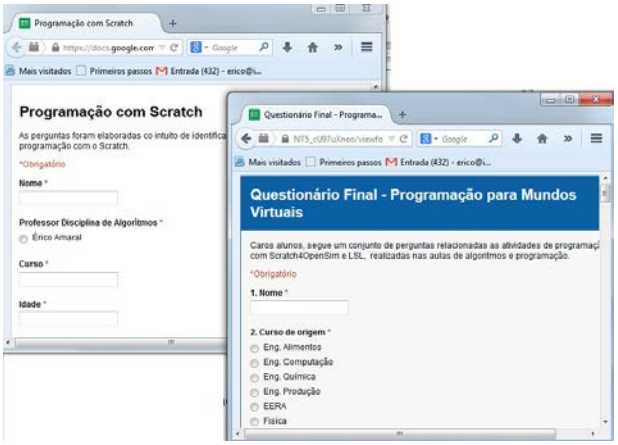
Figura 35–Sala de aula da disciplina de algoritmos (Moodle)



Fonte: Próprio autor.

Ao longo de todo o semestre foram utilizados instrumentos para coleta de dados sobre a percepção dos alunos em relação à praxe, novos conteúdos e reorganização da didática na disciplina de algoritmos. Os dois principais questionários disponibilizados, via Moodle, podem ser vistos na Figura 36. A fim de manter a organização deste estudo, os instrumentos receberam a identificação de Instrumento 1, o qual abrange questões sobre o uso do Scratch e, Instrumento 2, com a abordagem sobre a programação com Scracth4OS, LSL e algumas perguntas pertinentes ao experimento.

Figura 36–Questionários sobre as atividades na disciplina (Instrumento 1 e 2)



Fonte: Próprio autor.

As considerações e reflexões sobre os resultados obtidos neste experimento são apresentadas na próxima seção.

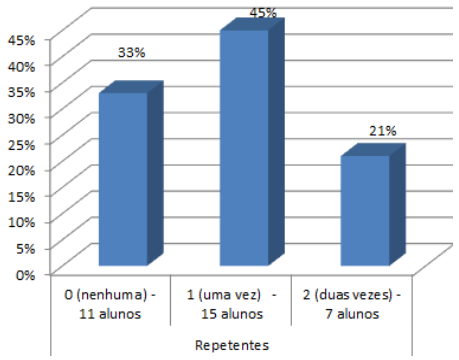
4.2.1 ANÁLISE DE RESULTADOS – PROJETO PILOTO FASE 02

A coleção de dados coletados durante o semestre constitui um repositório rico de dados sobre o experimento, contudo, serão apresentadas aqui apenas as informações de maior relevância para o projeto neste momento. Em relação aos participantes, na primeira atividade do semestre (nivelamento), foram levantadas informações pertinentes para o desenvolvimento das práticas propostas, conforme segue:

Uma questão importante está associada ao conhecimento prévio de programação dos alunos. Sendo assim os mesmos foram questionados se já haviam cursado a disciplina de algoritmos. Quinze alunos (45%) já haviam realizado uma vez a disciplina, (33%), ou seja,

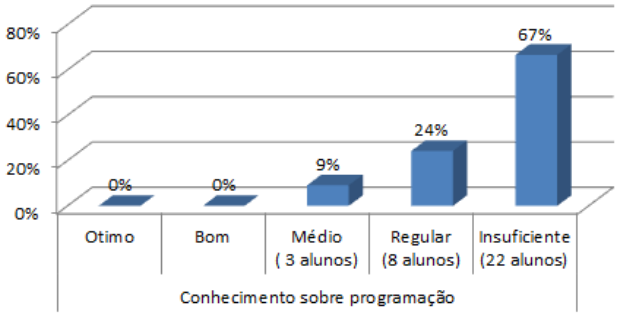
onze estudantes estavam cursando pela primeira vez e a minoria, sete discentes (21%) já haviam se matriculado duas vezes na disciplina (Gráfico 12). Diretamente relacionado a esta questão, solicitou-se uma auto avaliação do aluno, a fim de que eles definissem seus conhecimentos sobre programação, em que vinte e dois estudantes descreveram um entendimento insuficiente, oito disseram possuir um nível básico de conhecimento e apenas três apontaram um entendimento médio (Gráfico 13). Em contrapartida nenhum dos entrevistados apontou um saber bom ou ótimo sobre programação, o que mostra um relativo nível de consciência sobre suas habilidades na construção de soluções com algoritmos.

Gráfico 12–Número de vezes que cursou a disciplina



Fonte: Próprio autor.

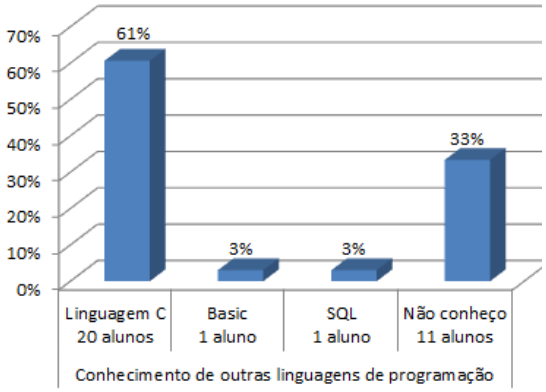
Gráfico 13–Auto avaliação sobre o conhecimento em programação



Fonte: Próprio autor.

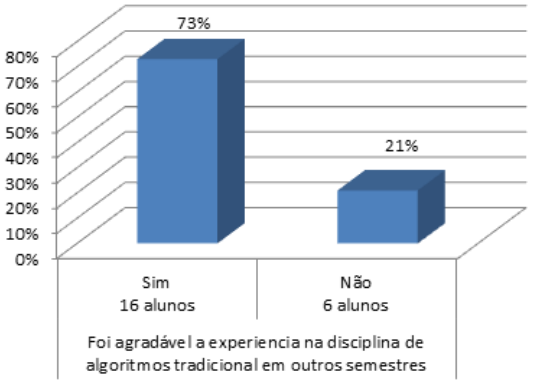
O Gráfico 14 mostra a resposta sobre o conhecimento de outras linguagens de programação. Esta foi uma pergunta aberta, em que o estudante poderia citar se já havia tido contato com outras ferramentas de programação. Como esperado, a grande maioria, vinte alunos apontaram a linguagem C, onze dos estudantes disseram não conhecer nenhuma linguagem.

Gráfico 14–Conhecimento de outras linguagens de programação



Fonte: Próprio autor.

Gráfico 15–Experiência em outras disciplinas de algoritmos



Fonte: Próprio autor.

Uma pergunta direcionada aos alunos repetentes questionou se foi agradável a experiência na disciplina de algoritmos tradicional, nos outros semestres. A resposta gerou certo nível de estranheza, visto que dezesseis alunos afirmaram que sim, contra seis que disseram não ter sido uma boa experiência (Gráfico 15). Ao buscar um esclarecimento sobre o percentual de alunos que aprovaram a metodologia de ensino dos semestres anteriores, os alunos descreveram que o método de ensino era o único que os mesmos já haviam tido contato até aquele momento, sendo que todos descreveram também que não cursaram a disciplina até o final do semestre e, por isso acreditavam que poderiam ter galgado a aprovação na matéria se tivessem uma maior dedicação à mesma.

Como última atividade de nivelamento, foi solicitado aos estudantes, que possuíam conhecimento de programação, desenvolver cinco programas de forma algorítmica ou com uma linguagem de programação. Nesta atividade dez alunos conseguiram resolver apenas um problema, quatro solucionaram dois problemas e apenas um aluno resolveu todos os exercícios, o que denota a baixa assimilação dos conteúdos por aqueles estudantes que já haviam cursado a disciplina.

Os dados de matrículas, evasão, aprovação e reprovação da disciplina de algoritmos ao final do semestre letivo, estão apresentados nos Quadros 18 e 19, respectivamente.

Quadro 18–Dados de matrícula e frequência

Dados de Matrícula e Frequencia	
Total matriculados	60
Evadidos (nunca compareceram)	27
Alunos regulares	33
Desistentes (evadidos após recesso)	12
Regulares (finalizaram a disciplina)	21

Fonte: Próprio autor.

Quadro 19–Dados de aprovação/reprovação

Dados de Aprovação/Reprovação			
	Geral	% 21	% 33
Aprovados	13	61%	40%
Reprovados	8	39%	24%
Evadidos	12	0%	36%

Fonte: Próprio autor.

Conforme o Quadro 18, o total de matriculados na disciplina do experimento foi de sessenta alunos, contudo, vinte e sete destes nunca compareceram as aulas, permanecendo trinta e três alunos regulares. Houve doze desistências registradas após a realização da avaliação 02, o que é explicado em grande parte por um recesso letivo ocorrido logo após esta atividade, que aconteceu devido ao calendário acadêmico estar desorganizado em relação aos outros anos, devido às paralisações (greves) ocorridas no ano de 2012. Ao final do semestre concluíram a disciplina vinte e um estudantes.

Com relação aos dados sobre aprovações/reprovações, o Quadro 19 demonstra que treze alunos que finalizaram a disciplina foram aprovados, oito reprovados e doze evadidos. O número de evadidos é apontado em relação aos trinta e três estudantes que frequentaram a disciplina, não computados os 27 discentes que abandonaram a disciplina sem nunca comparecer as aulas. Desta situação específica, o percentual de aprovações/reprovações pode ser analisado em duas situações:

- Considerando os vinte e um estudantes que realmente participaram de todas as atividades da disciplina, o que resultaria em 61% de aprovações contra 39% de reprovações;
- Considerando os trinta e três alunos que iniciaram a disciplina, mas que não finalizaram, têm-se 40% de aprovações, 24% de reprovações e 36% de evasões.

A partir do levantamento dos dados de aprovações/reprovações no âmbito da Universidade Federal do Pampa de 2010 a 2012 (Quadro 20), é possível afirmar que o experimento alcançou de forma positiva seus objetivos. Contudo é necessário entender os dados para chegar a esta conclusão.

No Quadro 20 são apresentados os totais de alunos Aprovados no período (1.039), o que gera um percentual de 46% sobre o montante de alunos que se matricularam e cursaram toda a disciplina de algoritmos, reprovando por nota (2.239), os quais são denominados aqui como “S/ Evasão”. O percentual de Aprovados em relação ao montante total de alunos matriculados no período (2.806), independentes de terem evadido ou reprovados por nota são aqui denominados “C/ Evasão”, e ficaram com 37%. Visto o objetivo deste trabalho em propor uma metodologia diferenciada para a abordagem e ensino de algoritmos, os dados de aprovação “C/ Evasão” não são considerados, pois não refletem a realidade dos alunos que frequentaram todo o semestre e participaram efetivamente dos experimentos.

A comparação entre a média de aprovações do experimento, nesta fase do projeto piloto, e a média de aprovações geral da Unipampa, com base no sistema tradicional do ensino de algoritmos, é demonstrada no Quadro 21. É possível observar neste Quadro que computando apenas os alunos frequentes em todo semestre “S/ Evasão”, o Experimento obteve um índice de aprovação de 61%, contra 46% da Unipampa, isto significa uma melhora de 15% no número de alunos aprovados na disciplina. Ao se analisar o total de alunos matriculados, independente de serem evadidos ou participarem de todo semestre (C/ Evasão), a eficiência obtida no Experimento é reduzida para 3% em relação ao número de aprovados na Universidade.

Quadro 20–Dados de aprovações/reprovações UNIPAMPA

Dados de Aprovação/Reprovação Algoritmos – UNIPAMPA (2010-2012)			
	Geral	% 2.239	% 2.806
Aprovados	1.039	46%	37%
Reprovados	1.200	54%	43%
Evadidos	567	0%	20%

Fonte: Próprio autor.

Quadro 21–Comparativo dos resultados alcançados

Comparação de resultados				
	EXPERIMENTO		UNIPAMPA	
	S/ Evasão	C/ Evasão	S/Evasão	C/Evasão
Aprovados	61%	40%	46%	37%
Reprovados	39%	24%	54%	43%
Evadidos	0%	36%	0%	20%

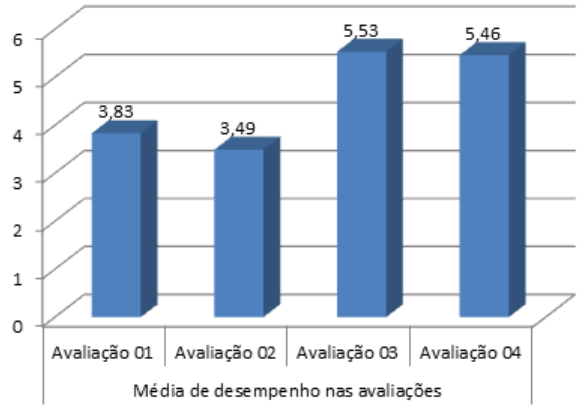
Fonte: Próprio autor.

A comprovação na efetividade desta proposta pode ser demonstrada a partir da mensuração do desempenho dos alunos nas avaliações. O Gráfico 16 ilustra a média de nota dos alunos, considerando as quatro avaliações realizadas durante o semestre. Nota-se que a média dos alunos nas duas primeiras provas ficou 3,83 e 3,49 respectivamente, já na avaliação

03 a média passa para 5,53, lembrando que nesta estão contempladas as atividades referentes ao uso do Scratch, Scratch4OS, LSL e OpenSim e, a média da prova 04 fica em 5,46. Buscando o embasamento teórico para corroborar com o resultado alcançado, é pertinente apontar que o contato com o ambiente de desenvolvimento visual, assim como as interações no mundo virtual permitiram construir habilidades necessárias para o desenvolvimento de soluções algorítmicas, não utilizando apenas as linguagens Scracth e LSL, mas também a linguagem C, sendo estas características propostas pela abordagem construcionista adotada.

Outro aspecto importante a ser frisado está relacionado ao aprimoramento das competências dos estudantes, os quais foram estimulados diretamente pelos aspectos motivacionais inerentes à estrutura de ensino disponibilizada. Ao encerrar as ações diretas do Experimento e retomar o uso da linguagem C, denotou-se claramente o reconhecimento, demonstrado pelo aluno, as estruturas necessárias para a implementação dos programas nesta linguagem. A questão da sintaxe continuou sendo um problema, contudo os estudantes conseguiram propor soluções lógicas para a resolução dos mesmos, o que permite afirmar que os participantes desenvolveram suas capacidades de Pensamento Computacional.

Gráfico 16–Média de desempenho dos alunos.

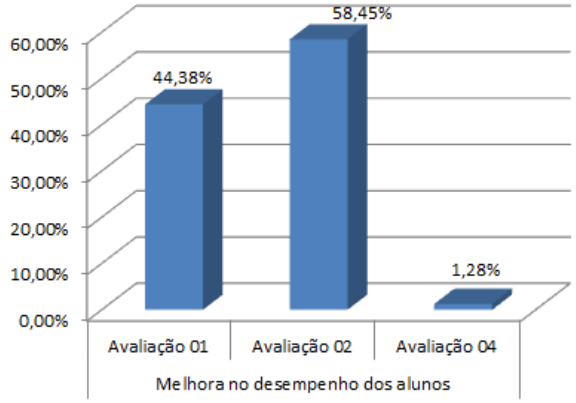


Fonte: Próprio autor.

A melhora no desempenho dos estudantes em relação ao entendimento dos conceitos/conteúdos e resolução de problemas, alcançados durante o Experimento, é comprovada no Gráfico 17, o qual apresenta os índices relativos a este desempenho. Os alunos atingiram uma média 44,38% menor na avaliação 01 em relação à avaliação 03, 58,45% menor na avaliação 02, também em comparação a avaliação 03. Contudo a construção do Pensamento Computacional e a mudança no perfil do entendimento da lógica de programação são percebidas por ocasião da avaliação 04, pois os estudantes apresentaram um índice de desempenho de apenas 1,28% menor em relação a avaliação 03. É importante

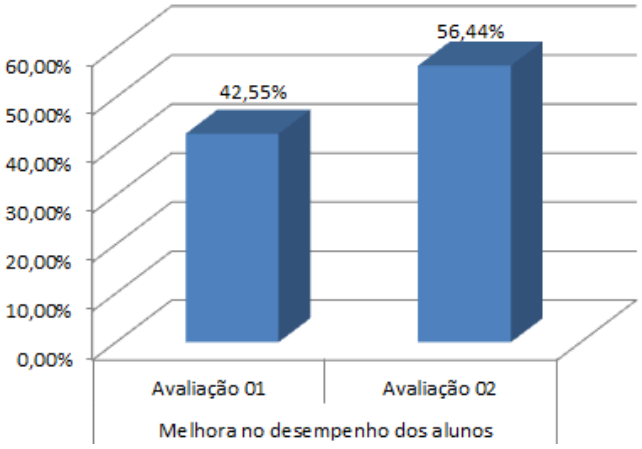
recordar que a primeira, segunda e quarta avaliações foram realizadas com base apenas na linguagem C, com uma proposta pedagógica tradicional.

Gráfico 17- Comparação na melhora de desempenho dos alunos na avaliação 03 em relação às avaliações 01, 02 e 04.



Fonte: Próprio autor.

Gráfico 18–Comparação na melhora de desempenho dos alunos na avaliação 04 em relação às avaliações 01, 02.



Fonte: Próprio autor.

O resultado na apreciação da atuação dos estudantes nas avalições 03 e 04, e sua superioridade em relação às médias alcançadas pelos alunos nas duas primeiras avaliações (01 e 02), conforme o Gráfico 18, onde fica caracterizado que o resultado das avaliações 03 e 04 ficaram com média 42,55% maior que o desempenho na avaliação 01 e, 56,44% maior que a avaliação 02, comprovando que os estudantes dedicaram-se de maneira mais efetiva após a avaliação 02, o que pode ser justificado por uma maior identificação cognitiva e motivacional com as ações propostas no Experimento, se tornando assim, participantes ativos do processo de aprendizagem, o que refletiu diretamente no desempenho na avaliação 04. Esta percepção

evidencia que a utilização do mundo virtual e das ferramentas de programação em blocos visuais fortaleceram a construção do conhecimento, corroborando para um melhor desempenho do estudante.

4.3. PROJETO PILOTO – FASE 03

A terceira fase do Projeto Piloto ocorreu durante o primeiro semestre de 2014 (2014/1) e caracterizou-se pela inversão na ordem dos conteúdos do plano de ensino proposto, conforme mostrado na Figura 37, na qual são apresentados os conteúdos relacionados ao Scratch, Scratch4OS e OpenSim no início do semestre, como conteúdos iniciais da disciplina, diferentemente da Fase dois, em que estes assuntos foram abordados após o estudo da linguagem C. Esta nova organização dos conteúdos foi denominada Modelo 2.

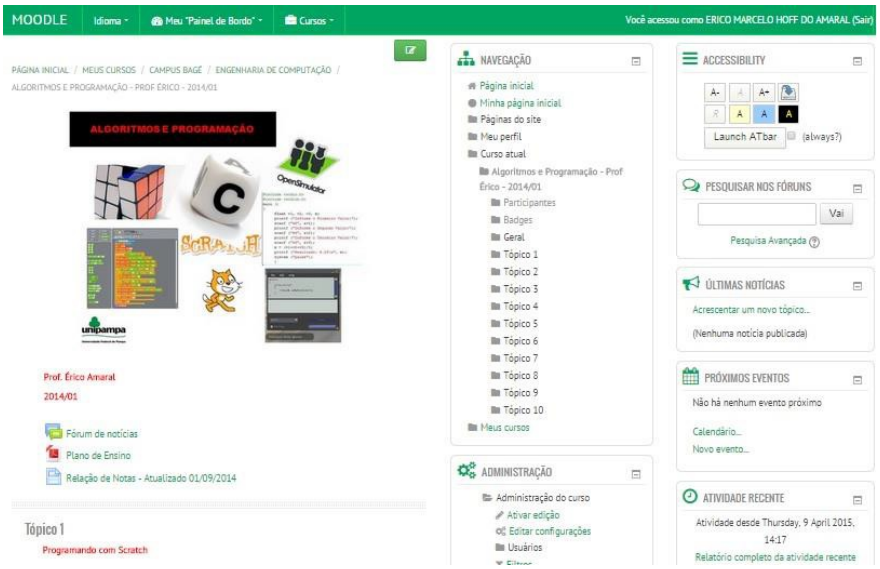
O intuito deste experimento é identificar em qual modelo de ensino o aluno demonstra um melhor desempenho na assimilação dos conteúdos, relacionados à disciplina de algoritmos e programação, se ao interagir com o Modelo 1, em que os conteúdos sobre programação com blocos visuais e mundos virtuais são apresentados ao final do período letivo, ou no Modelo 2, com estes elementos apresentados logo no início do semestre.

Figura 37–Plano de Ensino Adaptado (Modelo 2) – Projeto Piloto Fase 3

Fonte: Próprio autor.

Nesta etapa foram utilizados todos os materiais didáticos já implementados durante o projeto piloto 1 e 2, com pequenas alterações de contexto nas avalições, visto que o intuito era manter o nível de dificuldade das atividades, buscando ao máximo uma equidade nos resultados a serem obtidos. O contato e suporte aos alunos também foi realizado com a utilização de uma sala de aula no Moodle, conforme a Figura 38.

Figura 38–Sala de aula no ambiente Moodle – Projeto Piloto Fase 3 (2014/1)



Fonte: Próprio autor.

Em relação ao universo da pesquisa neste piloto, como em todo semestre letivo, a disciplina de algoritmos obteve 60 alunos matriculados, dos quais 21estudantes nunca compareceram às aulas, o montante de evadidos chegou a 35 discentes e apenas 25 frequentaram todo período letivo, compondo desta forma a amostra do experimento para esta etapa. Os cursos de origem e quantidade de alunos são apresentados no Quadro 22.

Quadro 22–Relação de matriculados e participantes do experimento

Participantes 2014/1				
Curso	**Matriculas**	**Nunca compareceram**	**Evadidos**	**Frequentes**
Engenharia de Energias Renováveis e Ambiente	10	3	3	4
Engenharia de Produção	50	18	11	21
Total	**60**	**21**	**35**	**25**

Fonte: Próprio autor.

4.3.1 ANÁLISE DE RESULTADOS – PROJETO PILOTO FASE 03

Neste ponto do estudo buscou-se subsídios para definir qual modelo de ensino apresentou maior efetividade em relação ao processo de aprendizagem na disciplina de algoritmos, se o Modelo 1 proposto no semestre 2013/2 ou Modelo 2, proposto em 2014/1.

Embora a alteração na disposição dos conteúdos destes Modelos pareça algo sutil, aspectos muito importantes foram considerados, como a diferença na construção da estrutura de pensamento do aluno, pois o estilo de programação orientado a blocos visuais é muito mais intuitivo, visualmente atrativo e promove a aprendizagem ativa (KOORSSE *et al*. 2015), diferente da programação com linguagens imperativa como o C.

Para alcançar um veredito sobre qual o Modelo deveria ser adotado no experimento final, vários elementos foram observados e elencados para apoiar esta decisão, os quais são apresentados a seguir.

A primeira informação considerada se referiu ao nível de evasão durante o semestre, visto que este é um problema já reportado em diferentes pesquisas, justificadas pela complexidade apresentada por programadores iniciantes, na construção do raciocínio lógico necessário para a implementação de programas (VASSILEV, 2015; ROCHA *e. al.* 2010).

Identificar o índice de alunos que abandonaram a disciplina durante o semestre letivo, em ambos os períodos (2013/2 e 2014/1), poderia servir como indicador para a definição dentre os Modelos de ensino propostos, qual o mais efetivo, que estimulou de forma positiva os alunos a concluírem todas as atividades propostas em algoritmos e programação. O Quadro 23 apresenta uma comparação entre o índice de evasão computado durante os dois semestres avaliados. O número de vagas/matriculas é igual em ambas às amostras (60), com 27 alunos que nunca compareceram as aulas em 2013/2 e 30 em 2014/1, um valor próximo e com importância relevante, mas não explorada neste estudo, visto que estes estudantes não tiveram contato algum com a proposta didática elaborada.

Quadro 23–Relação entre matriculados e evadidos (2013/2 e 2014/1)

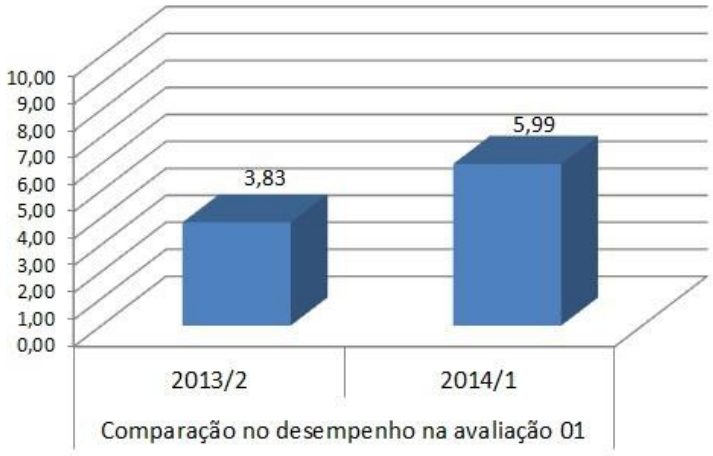
Matriculas X Evasão (2013/2 - 2014/1)									
Semestre	Matriculas	Nunca compareceram		Evasão Após Avaliação 01	Evasão Após Avaliação 02	Evasão Geral		Frequentes	
2013/2 (Modelo1)	60	27	45%	10	02	39	65%	21	35%
2014/1 (Modelo 2)	60	30	50%	02	03	35	58%	25	41%
Total	**120**	**57**	**47%**	**11**	**04**	**72**	**60%**	**48**	**40%**

Fonte: Próprio autor.

Na observação sobre a evasão após a primeira avalição (Quadro 23), tanto no Modelo 1 quanto no Modelo 2, tem-se uma diferença considerável entre os semestres, com 10 estudantes evadidos em 2013/2 e 2 discentes em 2014/1 – esta diferença plotada pode ser atribuída ao fato dos alunos apresentarem maior dificuldade no entendimento inicial da programação, utilizando a linguagem C (2013/2) e pelo fato da programação com a linguagem Scratch (2014/1) ser mais intuitiva, simples e motivadora (NIKOU & ECONOMIDES, 2014), em que o aluno consegue, a partir de uma única aula, desenvolver aplicações de forma autônoma. Esta observação demonstra que o Modelo 2 poderia estimular a redução no abandono, pois Castro *et al.* (2003), descreve, como um dos principais motivos da evasão em cursos de computação, a incapacidade dos estudantes realizarem abstrações lógicas no início das disciplinas de programação.

A diferença de desempenho na avaliação 01 pode também ser relacionada ao índice de abandono, visto que as médias dos estudantes em 2014/1(5,99) foram 56% superiores aos do grupo de 2013/2 (3,83), conforme Gráfico 19, corroborando com a afirmação de Malan & Leitner (2007) que apontaram em sua pesquisa uma queda no número de alunos dispersos em cursos de computação, quando a Linguagem Scratch é utilizada para introduzir conceitos de programação.

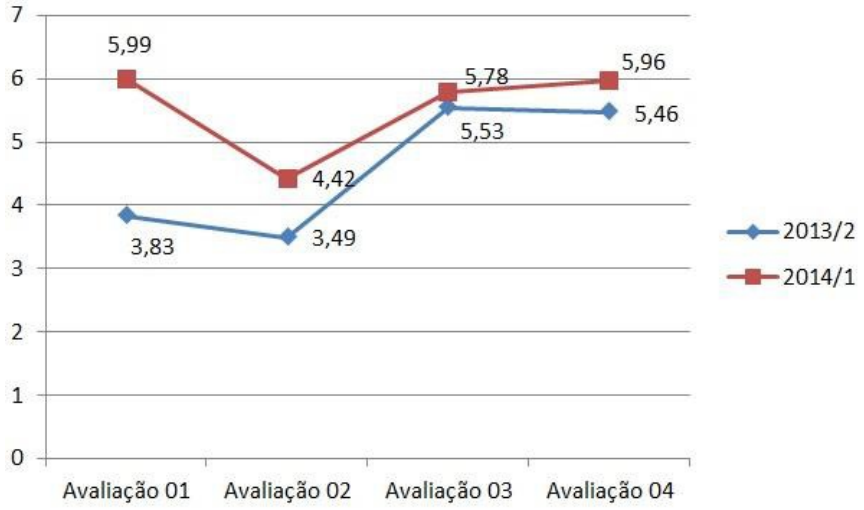
Gráfico 19–Comparação entre notas na primeira avaliação (2013/2 e 2014/1)



Fonte: Próprio autor.

Mantendo o foco sobre o desempenho dos alunos nas avaliações, durante os dois semestres aqui estudados, percebeu-se uma nítida diferença entre as médias alcançadas durante o período 2014/1 (Módulo 2) e 2013/2 (Módulo 1), conforme observado no Gráfico 20. A superioridade observada nas notas do Módulo 2 relaciona-se a inserção das linguagens Scratch, Scratch4OS e LSL no primeiro trimestre da disciplina, o que serviu como um ingrediente motivador e facilitador na compreensão dos conteúdos além de estimular a construção de conhecimento sobre lógica de programação. A condução dos conteúdos na ordem cronológica proposta em 2014/1 permitiu ao estudante construir uma ligação entre os conceitos introduzidos com as linguagens Scratch/LSL e os métodos para aplicar tais conceitos na Linguagem C (RESNICK *et al.* 2009; KOORSSE *et al.* 2015), estimulando a evolução dos discentes em relação à prática da programação. Mesmo ao se considerar a avaliação 04, que contempla os conteúdos de funções e procedimentos, assuntos de maior complexidade para iniciantes em programação (BARANAUSKAS, 1993), o grupo de 2014/1 obteve melhores médias.

Gráfico 20–Comparação entre médias das avaliações (2013/2 e 2014/1)



Fonte: Próprio autor.

A realização da inferência estatística, novamente utilizando o aplicativo SPSS, sobre os dados das médias dos alunos foi o caminho natural para a verificação de proximidade, ou não, destes valores.

Ao avaliar a normalidade das médias, considerando o universo de 21 alunos em 2013/2 e 25 alunos em 2014/1, que efetivamente participaram de todo experimento, percebeu-se que as amostras não caracterizavam-se como dados normais, conforme o resultado obtido através do teste de Shapiro-Wilk (Quadro 24), em que observou-se que os graus de significância permaneceram inferiores a 0,05 nos dois casos (2013/2 com *sig* 0,002 < 0,05 e 2014/1 com *sig* 0,031 < 0,05), indicando a necessidade da realização de testes não paramétricos.

Quadro 24–Teste de Normalidade das Médias (2013/2 e 2014/1)

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Grupo	Estatística	df	Sig.	Estatística	df	Sig.
Notas	2013/2	,316	21	,000	,832	21	,002
	2014/1	,189	25	,022	,911	25	,031

a. Correlação de Significância de Lilliefors

Fonte: Próprio autor.

Contudo a assimetria de ambas as amostras demonstrou graus de -0,394 (2013/2) e -0,760 (2014/1), apresentadas no Quadro 25, respeitando assim o intervalo de normalidade aproximada de Leech *et al.* (2005), a qual define em sua teoria, que dados de assimetria entre -1 e 1 permitem a utilização de testes paramétricos, como o teste t.

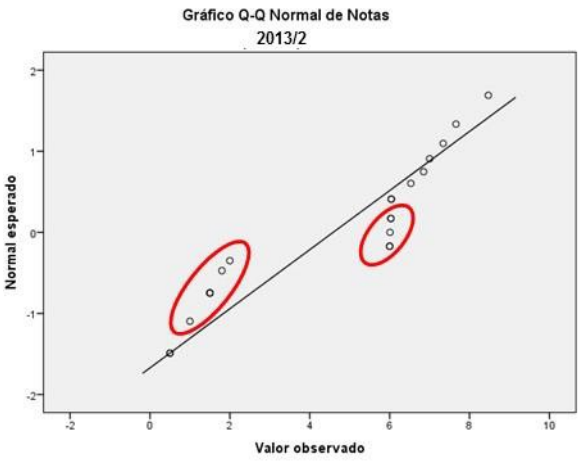
Quadro 25–Análise de assimetria das amostras de médias (2013/2 e 2014/1)

	Grupo			Estatística	Erro Padrão
Notas	2013/2	Média		4,5857	,59870
		95% Intervalo de Confiança para Média	Limite inferior	3,3368	
			Limite superior	5,8346	
		5% da média aparada		4,5990	
		Mediana		6,0100	
		Variância		7,527	
		Desvio Padrão		2,74360	
		Mínimo		,50	
		Máximo		8,47	
		Intervalo		7,97	
		Intervalo interquartil		5,19	
		Assimetria		-,394	,501
		Curtose		-1,627	,972
	2014/1	Média		5,5820	,37351
		95% Intervalo de Confiança para Média	Limite inferior	4,8111	
			Limite superior	6,3529	
		5% da média aparada		5,6727	
		Mediana		6,1300	
		Variância		3,488	
		Desvio Padrão		1,86757	
		Mínimo		1,60	
		Máximo		7,90	
		Intervalo		6,30	
		Intervalo interquartil		2,80	
		Assimetria		-,760	,464
		Curtose		-,481	,902

Fonte: Próprio autor.

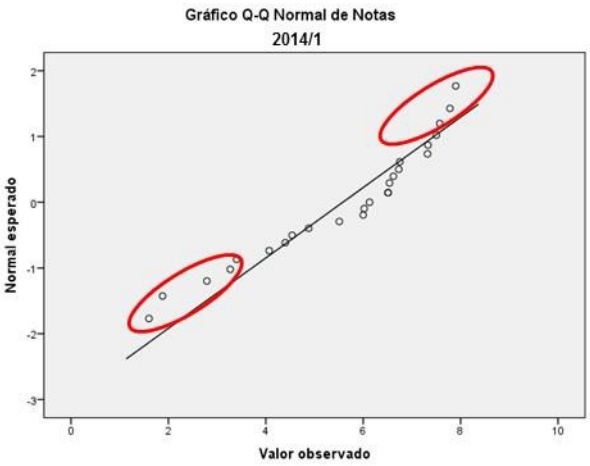
A plotagem dos dados em gráficos QQ (Gráficos 21 e 22) permitiu a visualização da característica de normalidade das médias, em que os pontos encontram-se dispersos da normal esperada, contudo em grau não elevado.

Gráfico 21–Gráfico QQ: Normalidade das Médias de 2013/2



Fonte: Próprio autor.

Gráfico 22– Gráfico QQ: Normalidade das Médias de 2014/2



Fonte: Próprio autor.

A comparação das médias, com o teste de Levene e teste t (Quadro 26), permitiu a verificação da homogeneidade das variâncias, destacando que estas são diferentes em ambos os grupos (2013/2 e 2014/1). Segundo o teste de Levene a significância associada resultou em 0,002, inferior a 0,05, descartando assim a homogeneidade das variâncias. Ao analisar as saídas para o teste t, considerando as variâncias distintas, obteve-se uma significância nas extremidades de 0,167, maior que 0,05. Este resultado aponta para a inexistência de diferenças estatísticas significativas entre as medias para um intervalo de 95% de confiança.

Quadro 26–Comparação da médias (Testes de Levene e Teste t)

Teste de amostras independentes

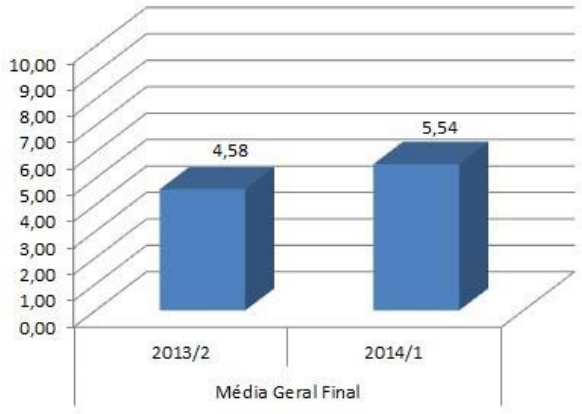
		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Notas	Variâncias iguais assumidas	11,179	,002	-1,459	44	,152	-,99629	,68299	-2,37277	,38020
	Variâncias iguais não assumidas			-1,412	34,272	,167	-,99629	,70566	-2,42994	,43737

Fonte: Próprio autor.

Embora o teste t tenha resultado numa proximidade estatística para as médias dos experimentos, a análise da média geral final dos dois semestres (Gráfico 23), aponta para uma diferença de 0,96, quase um ponto superior, para os alunos que participaram na disciplina organizada sobre a proposta definida como Modelo 2. Ao se tratar de uma disciplina de algoritmos e programação é plausível assumir que este seja um resultado positivo, visto que

uma pequena diferença na média pode significar uma evolução considerável no desempenho do aluno (DE OLIVEIRA *et al.* 2006).

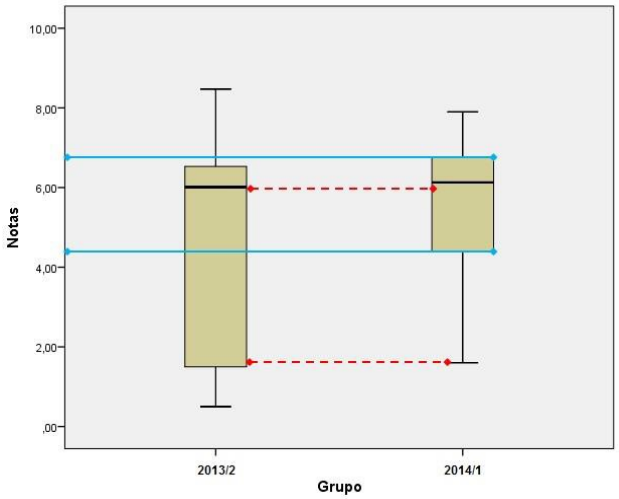
Gráfico 23–Médias finais (2013/2 e 2014/1)



Fonte: Próprio autor.

Outro recurso que destaca a efetividade do Modelo 2 é apresentado no Gráfico 28, uma representação em boxplot dos dados de média final dos alunos, em que é possível observar tendências, descartar os *outliers* (valores discrepantes da amostra) e entender o grupamento das informações (KAMPSTRA, 2008; PFANNKUCH, 2006). Neste caso específico das médias, o limites inferior (percentil 25%) e superior (percentil 75%), da amostra de 2014/1 destacam uma região com notas entre 4,0 e 7,0 (interquartil), evidenciando que o padrão de notas deste grupo apresentou um interquartil de maior desempenho que os discentes do Modelo 1 (2013/2). A mediana de ambas as médias ficaram próximas, porém maior para os dados de 2014/1.

Gráfico 24–Boxplot das médias (2013/2 e 2014/1)



Fonte: Próprio autor.

O estudo sobre o desempenho dos alunos durante o projeto piloto 1 e 2, avaliado por suas médias, mostrou de forma singular que a proposta didática para o ensino de algoritmos, definida como Modelo 2, caracterizou-se mais efetiva que o Modelo 1, além de ser uma prática válida para a melhoria no processo de ensino e de aprendizagem da lógica e da programação de computadores.

Com o intuito de corroborar com as observações realizadas nesta etapa do projeto em relação ao melhor Modelo didático para a disciplina de algoritmos, foram elencadas algumas questões do instrumento de pesquisa respondido pelos alunos (Instrumento 1 e 2). A amostra dos dados correspondente a estes instrumentos é composta por um universo de 51 respostas. O Quadro 27 apresenta as indagações analisadas.

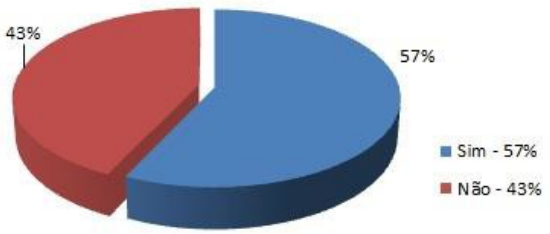
Quadro 27–Questões avaliadas do Instrumento 1 e Instrumento 2

1. A partir das atividades realizadas com as Linguagens Scratch, LSL e C, você considera-se apto a, de forma autônoma, construir novos projetos de programação.
2. Sobre o desenvolvimento da disciplina de algoritmos, qual a linguagem você considera que deveria ser utilizada no inicio do semestre.
3. Avaliando as linguagens de programação estudadas durante o semestre, ordene-as de acordo com o nível de dificuldade, segundo a sua percepção (da mais difícil para a mais fácil).
4. Analisando a organização da disciplina de algoritmos, qual a ordem de estudo você consideraria a mais adequada para a aprendizagem de programação.

Fonte: Próprio autor.

A questão 1, uma pergunta direta com resposta objetiva (Sim/Não), apontou que 57% dos estudantes após o contato com a linguagem Scratch, LSL e C declararam-se aptos a implementar soluções de programação (Gráfico 25). Esta afirmação indica um elevado nível de motivação e confiança do estudante na capacidade de resolver problemas por meio de recursos computacionais, corroborando com Tanrikulu & Schaefer (2011), os quais mostram em seu estudo, que programadores iniciantes sentem-se mais satisfeitos ao iniciarem seus estudos em ambientes de desenvolvimento visuais como o Scratch.

Gráfico 25–Construção de projetos de programação de forma autônoma



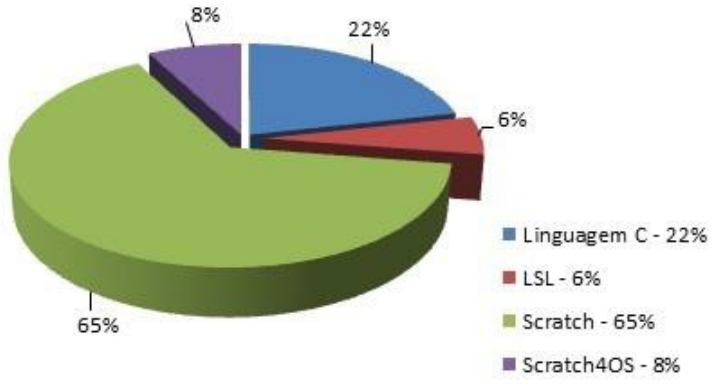
Fonte: Próprio autor.

Quando questionados sobre qual a linguagem deveria ser utilizada no início da disciplina de algoritmos (questão 2), 65% sugeriu a utilização do Scratch, 22% a Linguagem C, 8% a adoção do Scratch4OS e apenas 6% propôs o uso da Linguagem LSL (Gráfico 26).

A compreensão destes resultados é facilitada com o entendimento que os ambientes de programação tradicionais estão atrelados a regras rígidas de sintaxe e pontuação, impondo uma barreira aos estudantes e levando-os a uma sobrecarga (MEERBAUM-SALANT *et al.* 2010; WOLZ *et al.* 2008), diferente da interação com uma Linguagem baseada em um paradigma de blocos de programação visual, como o Scratch, que é caracterizado por disponibilizar uma abordagem de desenvolvimento simples, sem a necessidade do aluno se preocupar com detalhes de sintaxe e permitindo que o mesmo concentre esforços na lógica para resolução dos problemas (RESNICK *et al.* 2009).

O alto percentual relacionado a Linguagem C é justificado pelo fato dos estudantes já possuírem uma bagagem prévia de conhecimento sobre este ambiente de desenvolvimento, conforme descrito por alguns alunos. O LSL e Scratch4OS são abordados de forma igualitária nas atividades da disciplina, contudo ao questionar os estudantes sobre este resultado (6% para o uso do Scratch4OS), os mesmo alegaram que estas linguagens deveriam sempre complementar os estudos sobre Scratch.

Gráfico 26–Linguagem a ser utilizada no início do semestre

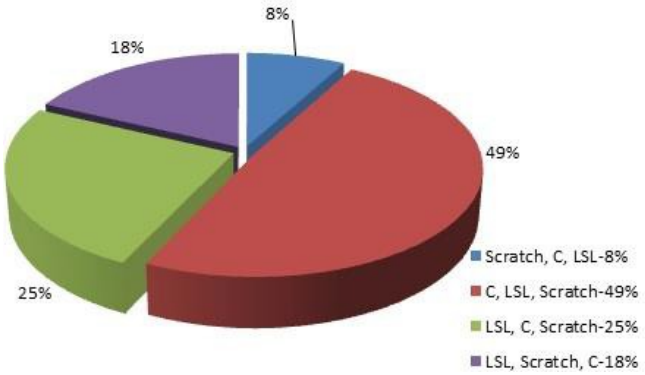


Fonte: Próprio autor.

A relação entre o nível de complexidade das linguagens de programação foi avaliada na questão 3, através da qual os estudantes deveriam ordenar os ambientes de programação de

acordo com o nível de dificuldade (Gráfico 27). Um percentual significante (49%) citou a linguagem C como a mais complexa, seguida da linguagem LSL e por último a linguagem Scratch. Em segundo, com 25%, a ordem de complexidade ficou LSL, C e Scratch. Em terceiro (18%) LSL, Scratch e C, e por último (8%) Scratch, C e LSL. Visto os resultados anteriores, as informações coletadas a partir desta questão foram pertinentes ao panorama identificado até o momento. As linguagens apontadas como mais complexas (C e LSL) são baseadas em uma sintaxe muito semelhante, o que reporta ao estudante a necessidade de aprender corretamente a sintaxe e especificações do ambiente, para apenas depois construir o raciocínio lógico necessário à resolução dos problemas.

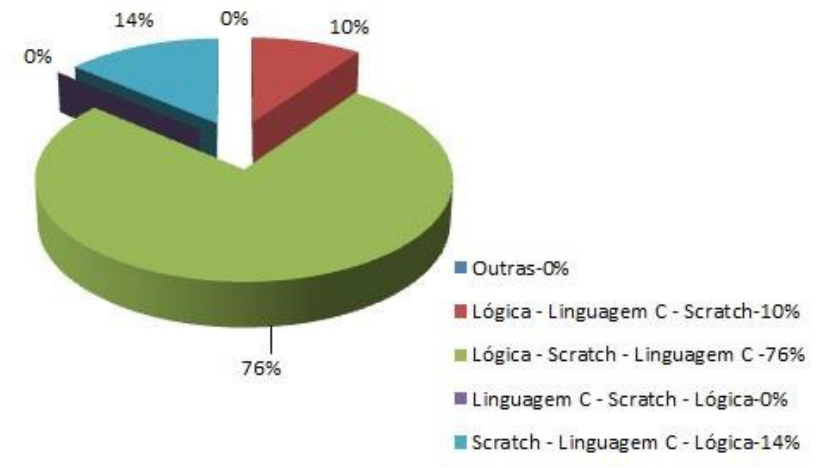
Gráfico 27–Complexidade dos ambientes de programação



Fonte: Próprio autor.

A última pergunta avaliada (questão 4 – Gráfico 28) está diretamente relacionada à discussão sobre qual Modelo (Modelo 1 ou 2) deveria ser adotado na disciplina de algoritmos, pois solicita ao estudante opinar em relação a este tema, sugerindo a ordem dos conteúdos a ser abordado durante o semestre. A principal resposta apresentada pelos estudantes apontou para uma abordagem baseada na sequência de conteúdos Lógica, Scratch e C (75%), a qual está precisamente alinhada à proposta do Modelo 2. Um percentual de 10% dos estudantes indicou a adoção da Lógica, linguagem C e Scratch e, 14% sugeriram iniciar com o ambiente Scratch, seguido do C e por último contemplar o tema Lógica de programação, resposta que deve ser desconsiderada, visto que não é viável deixar os conceitos de lógica para o final do semestre.

Gráfico 28–Ordem dos conteúdos a serem ministrados na disciplina de algoritmos



Fonte: Próprio autor.

Como conclusão final do Projeto Piloto 03, pode-se afirmar que o Modelo 02, proposto nesta etapa do estudo, atende à demanda de um modelo didático mais efetivo para o ensino de algoritmos e programação. As diferentes implicações entre a adoção de uma linguagem de programação tradicional ou ambientes de programação baseados em blocos visuais e programação para mundos virtuais foram amplamente consideradas e analisadas, mostrando que a ordem dos conteúdos inicialmente abordados, neste tipo de disciplina, influencia diretamente o desempenho dos estudantes.

A inferência estatística sobre os dados de desempenho dos alunos (notas e médias), considerando os semestres do Projeto Piloto 2 (2013/2) e Projeto Piloto 3 (2014/1), mostraram de forma clara que o Modelo 2, adotado em 2014/1, estimulou o estudante a obter melhores índices nas avaliações realizadas no período, alcançando uma melhora de quase 1,0 ponto na média geral da turma. Além deste resultado, a análise de 4 questões pontuais dos Instrumentos 1 e 2, tratando sobre a complexidade e ordem dos conteúdos a serem trabalhados em sala de aula, resultou em uma afirmação pontual partindo dos alunos, de que o Modelo adotado no semestre letivo de 2014/1 permitiu um maior entendimento sobre os temas relacionados à lógica, programação e construção do Pensamento Computacional.

Pode-se afirmar, neste ponto da pesquisa, que o Modelo 2 caracterizou-se como um Sistema Personalizado de Ensino (SPE), pois o professor passou a acompanhar, treinar e gerenciar os processos de ensino, diferente do modelo tradicional, em que o docente simplesmente transmite a informação (TODOROV & TRISTÃO, 1975). Além disso, a caracterização da metodologia SPE é sustentada pelos procedimentos adotados durante o

Modelo 2, tais como: a disponibilização através do Moodle de diversos materiais de apoio, coerentes e com diferentes níveis de detalhamento; organização dos materiais com base no nível de complexidade e, de forma sequencial; mobilização do professor e monitor para atendimento aos alunos, em momentos distintos, o que permitiu um acompanhamento na evolução do desempenho de cada estudante; pelas avaliações realizadas durante a disciplina, a fim de monitorar e organizar a trajetória dos alunos em relação aos assuntos abordados e; pela característica de *feedback* das linguagens adotadas nas etapas iniciais do Modelo 2 (ROCHA *et al.* 2010; CHASE & HOUMANFAR, 2009).

O aluno, por sua vez, foi um participante ativo do processo de aprendizagem de algoritmos e programação, o que permitiu o aprimoramento dos métodos de transmissão dos conteúdos e o estímulo, das diferentes capacidades dos estudantes, em assimilar os assuntos discutidos em sala de aula.

A participação ativa foi estimulada pela facilidade na construção de resposta aos problemas iniciais devido à simplicidade e facilidade do uso do ambiente de programação Scratch. Os blocos de comando visuais, característicos deste ambiente de programação foram pensados pelos criadores da linguagem Scratch (MIT, 2014) como passíveis de serem usados mesmo por crianças, daí sua simplicidade. Na medida em que os alunos venceram as primeiras etapas com facilidade foram adquirindo autoconfiança que se manteve quando passaram para o ambiente menos amigável típico das linguagens de programação mais tradicionais.

5. EXPERIMENTO FINAL

Os resultados obtidos com o Experimento Piloto demonstraram que a adoção de uma abordagem didática, calcada sobre novos paradigmas, para o processo de ensino e de aprendizagem de algoritmos é uma necessidade e, pode ter efeitos positivos, corroborando com Vosinakis *et al.* (2014), os quais defendem em sua pesquisa que o ensino de programação por meio de um paradigma tradicional, baseado em ferramentas não visuais, pode confundir e desmotivar alunos inexperientes. Nesta linha de raciocínio a utilização, neste estudo, de recursos visuais como mundos virtuais e programação baseada em blocos visuais permitiram ao estudante a construção de um processo de dedução, através de relações lógicas, o que também foi comprovado por Mondshein *et al.* (2010) e Senay & Lazzeri (1991).

A proposta de uma metodologia diferenciada, para uma disciplina de algoritmos e programação, foi explorada e devidamente analisada nos experimentos iniciais deste estudo de tese, dentre as quais, destacou-se o que ficou definido como Modelo 2, uma estrutura didática de ensino organizada em uma concepção inicial sobre lógica e raciocínio, construção de fluxogramas, programação com a linguagem Scratch, desenvolvimento com o Scratch4OS, construção de *scripts* para mundos virtuais com o LSL e, por fim, utilização da linguagem C, com uma programação estruturada e imperativa, conforme apresentado no infográfico da figura 39.

Figura 39-Infográfico sobre conteúdos do Modelo 2

Fonte: Próprio autor.

A partir da identificação da viabilidade de utilização do Modelo 2, o Experimento Final aqui descrito, replicou as ações do Experimento Piloto 03, durante o segundo semestre de 2014, empregando a mesma metodologia e recursos, contudo, visto que a melhora no ganho de desempenho dos alunos ao ter contato com este esquema didático já ficou comprovada, nesta etapa do estudo buscou-se respostas para 2 questões importantes: Como se deu o desenvolvimento e construção do Pensamento Computacional dos estudantes durante as atividades propostas e; qual a melhora (quantitativa) na média destes alunos em relação aos demais estudantes que cursaram a disciplina de algoritmos e programação no período desta pesquisa.

O plano de ensino e atividades avaliativas desenvolvidas durante o semestre estão apresentados na Figura 40.

Figura 40-Plano de Ensino e Atividades Avaliativas do Experimento Final

PLANO DE ENSINO – Experimento Final

Conteúdo Programático de Algoritmos
Conceito e definição de algoritmos, estudo de problemas
Identificação dos valores de entrada e de saída
Noções de lógica e Conceitos de variável
Introdução a Programação com Scratch
Exercícios práticos com Scratch
Operadores (aritméticos e atribuição), condições de operadores (relacionais e lógicos) e Estruturas Condicionais com Scratch
Estruturas de Repetição, Vetores e Matrizes com Scratch;
Reconhecendo o Scratch4OS
Introdução do Mundo Virtual OpenSim
Programando objetos no OpenSim com o Scratch4OS
Introdução a Linguagem de Programação LSL
Programando objetos no OpenSim com LSL
Conceito de variável, tipos de variáveis na linguagem C
Comandos de entrada e saída
Operadores (aritméticos e atribuição) e Estruturas Condicionais
Estruturas de condição e operadores (relacionais e lógicos)
Estruturas de repetição
Teste de mesa
Vetores
Matrizes
Funções auxiliares (strings)
Funções, variáveis (locais e globais) e passagem de parâmetros
Procedimentos e passagem de parâmetros

Avaliações		Conteúdo
Avaliação 01 Scratch Scratch4OS LSL	Avaliação 1.1	Identificação dos valores de entrada e de saída; Noções de lógica; Construção de aplicações simples com Scratch;
	Avaliação 1.2	Conceitos de variáveis; Operadores (aritméticos e atribuição), condições de operadores (relacionais e lógicos) e Estruturas Condicionais com Scratch;
	Avaliação 1.3	Estruturas de Repetição, Vetores e Matrizes com Scratch; Scratch4OS;
	Avaliação 1.4	Programando com LSL;
Avaliação 02 Linguagem C		Conceito de variável, tipos de variáveis; Comandos de entrada e saída; Operadores (aritméticos e atribuição); Estruturas Condicionais; Estruturas de condição e operadores (relacionais e lógicos);
Avaliação 03 Linguagem C		Estruturas de repetição; Teste de mesa; Vetores; Matrizes; Funções auxiliares (strings);
Avaliação 04 Linguagem C		Funções, variáveis (locais e globais); Passagem de parâmetros; Procedimentos;
Avaliação 05 - Recuperação Scratch, Scratch4OS, LSL, Linguagem C		Todo conteúdo da disciplina

Fonte: Próprio autor.

Além das atividades avaliativas, um conjunto de práticas formativas, denominadas de Casos, foram propostas em sala de aula, a fim de coletar subsídios e indícios da construção do Pensamento Computacional e raciocínio lógico dos estudantes. Algumas destas práticas estão apresentadas no Quadro 28.

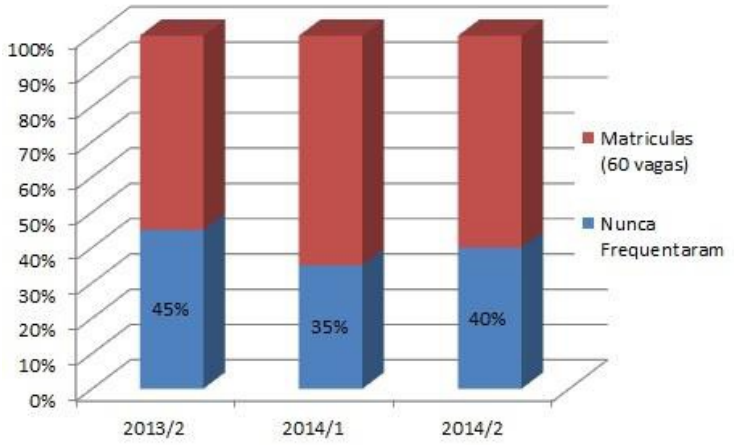
Quadro 28- Práticas Formativas realizadas (Casos)

Casos	Titulo	Descrição
Caso01	Nivelamento	Atividade baseada em um instrumento de pesquisa, composto por questões abertas e objetivas abordando o conhecimento dos alunos sobre programação. Os alunos deveriam implementar códigos em linguagem C.
Caso02	Lógica	Atividade realizada com a finalidade de estimular o raciocínio lógico dos alunos, por meio da interação com a aplicações desenvolvidas em flash.
Caso03	Fluxograma	Os alunos deveriam construir fluxogramas para resolução de problemas com demandas diferenciadas.
Caso04	Scratch	Prática desenvolvida para explorar a capacidade dos alunos em implementar um programa simples, utilizando os recursos da linguagem Scratch.
Caso05	Interpretação	Exercício voltado para a análise e interpretação de dos requisitos em aplicações desenvolvidas com Scratch.
Caso06	Identificar Erros	Tarefa com propósito de estimular o senso critico do aluno, sua aptidão em analisar programas prontos e de apontar a existência ou não de erros.
Caso07	Programação LSL	Uma prática voltada a interpretação de códigos desenvolvidos com a linguagem LSL.
Caso08	OpenSim	Implementação de aplicações para o Mundo Virtual.
Caso09	Programação com Linguagem C	Este exercício consistiu no desenvolvimento de aplicações com a Linguagem C, a partir de requisitos descritos em fluxogramas.
Caso10	Interpretação de Códigos em C	Caso composto por duas atividades de interpretação de códigos desenvolvidos em linguagem C, com a finalidade de identificar erros de sintaxe, lógica e classificação de níveis de acerto.

Fonte: Próprio autor.

No semestre da realização deste experimento, como nos demais, houveram 60 alunos matriculados na disciplina de algoritmos e programação, dos quais, 24 (40%) nunca compareceram as aulas. O número de evadidos, que em nenhum momento estiveram presentes nas aulas, foram equivalentes nos 3 semestres observados, conforme pode ser percebido no Gráfico 29. Esta característica deve ser avaliada em outros trabalhos, relacionados especificamente a este tipo de evasão, contudo uma das causas potenciais para este tipo de situação é a matrícula não consciente dos alunos, devido à disponibilidade na oferta de vagas todo semestre para esta disciplina. Também é passível de consideração, neste momento, o estudo sobre índices de desistência e reprovação em disciplinas de algoritmos apresentado Deters *et al.* (2008), o qual descreve percentuais aproximados de 60% para evasão ou reprovação de estudantes nestes cursos, relacionando diretamente estas taxas com o processo de ensino e de aprendizagem.

Gráfico 29–Quantitativo de alunos que nunca compareceram na disciplina de algoritmos



Fonte: Próprio autor.

A distribuição dos alunos frequentes, nos primeiros encontros da disciplina, é demonstrado no Quadro 29, com um número de 36 alunos (60%) presentes.

Quadro 29- Relação de matriculados e participantes do experimento Final

Participantes 2014/2	
Curso	**Alunos**
Engenharia de Computação	13
Engenharia de Energias Renováveis e Ambiente	7
Engenharia Química	3
Engenharia de Produção	10
Engenharia de Alimentos	1
Licenciatura em Matemática	2
Total	36

Fonte: Próprio autor.

Com o intuito de acompanhar o desenvolvimento das habilidades em programação dos alunos, durante o semestre, foi proposto um conjunto de atividades formativas e avaliativas, as quais forneceram subsídios para o estudo e definição de um parecer sobre as características do estudante, durante o período de formação, com base no modelo de ensino recomendado desta pesquisa. A fim de abordar tais aspectos este capítulo está dividido em três subcapítulos: O primeiro com uma análise de atividades formativas, visando apontar particularidades na construção do raciocínio lógico do aluno; o segundo com uma avaliação estatística do desempenho dos estudantes, a partir da observação de suas notas em atividades avaliativas, e;

Por último, o fechamento do Capítulo, com a descrição do desenvolvimento e aplicação de uma taxonomia específica para a identificação na evolução do Pensamento Computacional dos discentes participantes deste estudo.

5.1. ANÁLISE DA CONSTRUÇÃO DO RACIOCÍNIO LÓGICO

Entendendo que uma das principais causas relacionadas aos problemas de programação, percebidas por programadores novatos, segundo Proulx (2000) e Faria & Adán Coello (2004), baseiam-se no desconhecimento de padrões de raciocínio lógico na solução de problemas, o objetivo nesta etapa do estudo foi perceber a evolução dos alunos em relação as suas capacidades de resolver problemas por meio da lógica de programação, com diferentes paradigmas, para isso elaborou-se um método de identificação do progresso destes estudantes, por meio da análise de um conjunto de ações de programação, aplicadas durante todo o semestre letivo, que permitiram acompanhar a transformação da consciência destes indivíduos sobre a forma possível de pensamento, para diferentes propostas, em resposta as questões disponibilizadas.

Para alcançar a finalidade desta etapa da pesquisa propôs-se uma estratégia para a análise das atividades dos alunos, embasada na técnica de Raciocínio Baseado em Casos (RBC), que é uma abordagem para o aprendizado calcada em experiências passadas (AAMODT & PLAZA, 1994; VON WANGENHEIM, 2003; AYELDEEN *et al.* 2015). A justificativa na adoção desta técnica é corroborada por Leake (1996), o qual aponta que o RBC apresenta como um de seus focos a modelagem do comportamento humano, tendo como princípio a descrição de que a solução proposta para problemas passados estão diretamente relacionadas com respostas aos problemas atuais.

Neste contexto, o reconhecimento de casos e teorias anteriormente adotadas servem de base para novos raciocínios, os quais permitem a construção de uma solução, em parte ou completa, para novos problemas (HEINZEN, 2002). A concepção da RBC nesta pesquisa é percebida como um modelo para entender os aspectos relacionados ao pensamento dos indivíduos, a fim de perceber se o estudante, na medida em que avança nos conteúdos apresentados, acumula conhecimento necessário para formalizar o raciocínio lógico e Pensamento Computacional, indispensáveis para a resolução algorítmica de questões reais.

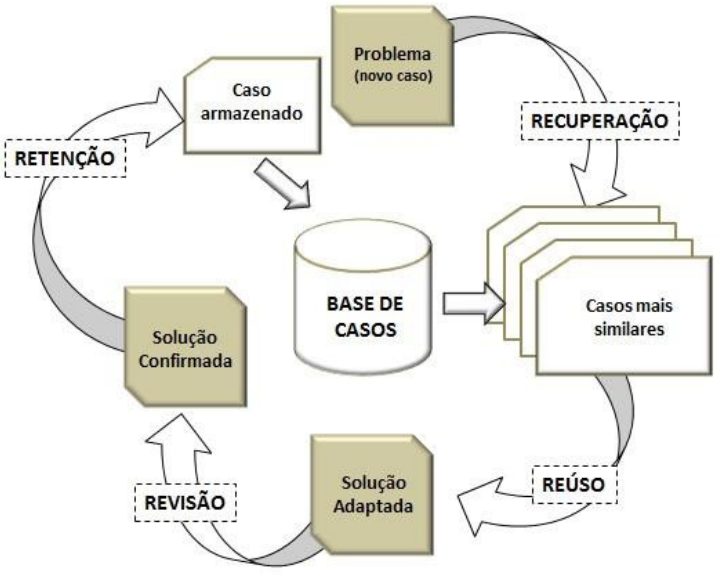
Também, ao entender a concepção de Piaget (PIAGET, 1964) que o conhecimento é gerado a partir da relação e da contínua interação do sujeito-organismo e objeto-meio, calcado nas experiências do indivíduo com o objeto, em um processo comumente provocado em relação a um tópico específico é prudente assumir a viabilidade de aplicação da técnica de

RBC neste estudo, visto que a estrutura do raciocínio baseado em casos prevê uma sequência de conteúdos previamente organizados, com o intuito de estimular de forma apropriada às competências dos alunos em disciplinas de algoritmos e programação.

A variante na adoção da RBC neste trabalho se refere à observação da evolução do aluno em termos do seu raciocínio lógico, por meio da disponibilização de atividades relacionadas como casos passados e correntes (AAMODT & PLAZA, 1994; MELCHIORS & TAROUCO, 1999), com uma associação explorada de forma intuitiva, através de conteúdos correlatos e complementares. O intuito desta prática é descobrir a validade do Modelo 2, como forma de estímulo ao aluno na busca de experiências semelhantes em memória, afim de auxiliá-lo para a tomada de decisões atuais. As atividades elencadas no Quadro 28 são tidas como casos, representados em diferentes formatos, contudo calcados sobre conhecimentos específicos de lógica e programação. Busca-se que, as experiências promovidas por estas tarefas sejam armazenadas pelo estudante em sua estrutura de memória, se tornando úteis para alcançar objetivos, atuais ou futuros, conforme defendido por Kolodner (1993).

A Figura 41 apresenta um modelo de RBC, de aceitação geral, sugerido por Aamodt & Plaza (1994) o qual demonstra um ciclo de raciocínio contínuo composto por um conjunto de casos. As etapas deste ciclo são descritas como (VON WANGENHEIM, 2003): recuperar, casos armazenados na base; reutilizar, utilizar os casos passados para encontrar soluções para casos atuais; revisar, avaliar a resposta implementada e; reter, armazenar a experiência atual para utilização futura.

Figura 41-Ciclo de Raciocínio Baseado em Casos



Fonte: Aamodt & Plaza (1994).

Por sua vez, a Figura 42 descreve a utilização da RBC, como uma estrutura organizacional, para o apoio na função de analisar a construção do conhecimento sobre raciocínio lógico dos alunos na disciplina de algoritmos, com base na prática pedagógica denominada Modelo 2. A adaptação da RBC é percebida pela restrição dos casos, que neste modelo são oriundos do conjunto de atividades realizadas em sala de aula. Espera-se que naturalmente as ações dos alunos contemplem as etapas de recuperação, reuso, revisão e retenção, populando a base de casos com o conhecimento sobre as tarefas já resolvidas. A observação deverá ser realizada na etapa de retenção, a partir da verificação das respostas entregues pelos discentes durante o semestre do experimento.

Figura 42-Proposta de utilização do RBC

Fonte: Adaptado de Aamodt & Plaza (1994).

A seguir será apresentado o conjunto de atividades avaliadas durante o semestre, cada uma interpretada e avaliada de forma independente, de acordo com a sua finalidade e conteúdo abordado.

5.1.1. ANÁLISE DE RESULTADOS – CASO01

A primeira ação para esta análise baseou-se na observação de um instrumento de pesquisa respondido pelos alunos no primeiro dia de aula (Caso01 – Quadro 28), proporcionando um *feedback* dos 36 estudantes participantes da atividade, os quais apresentaram uma faixa etária de 26 anos. Alguns pontos relevantes a serem descritos:

1. 44% dos estudantes estavam cursando pela primeira vez algoritmos, 28% já haviam cursado uma vez e, 28% cursaram duas ou mais vezes esta disciplina.
2. Do total de alunos que já haviam frequentado as aulas de programação em semestres anteriores, a grande maioria (68%), afirmou reconhecer que seu desempenho foi insuficiente, justificado pelo fato da sua reprovação e da consciência sobre este fato.
3. Quando questionados sobre a utilização de exemplos básicos para a resolução de problemas, técnica conhecida como epitomização e apresentada por Wiley (2000), 85% dos estudantes descreveram utilizar códigos prontos como suporte para resolução de problemas e, apenas 16% não usou deste artifício. Estas respostas apontam para uma característica desejada no grupo, a capacidade de interpretar códigos e a partir de elementos básicos, como ponto de partida, resolver problemas diferentes, ficando enquadrado na teoria de sequenciação (WILEY, 2000; TAROUCO *et al.* 2006).
4. 94% dos estudantes nunca realizou qualquer atividade ou curso relacionado a programação fora da academia, justificando o fato de 41% afirmar conhecer apenas a linguagem C, por ser o paradigma padrão adotado na universidade, 47% descreveu não ter tido contato com nenhuma linguagem de programação. Este contexto mostra que uma grande parte dos discentes (88%) chega ao curso superior sem conhecer a prática de programação, fato que torna a disciplina de algoritmos um desafio (DOS SANTOS & COSTA, 2006).
5. Durante a resposta a este instrumento foram apresentados dois códigos do mesmo programa, em Scratch e com a linguagem C, a partir disto foi questionado qual modelo de programação seria mais interessante a ser adotado na disciplina de algoritmos. 57% dos alunos indicaram que a programação com blocos visuais seria mais interessante, 33% não se posicionaram em relação ao tema, por não estarem seguros e, 10% acharam a programação com C mais agradável. Este panorama corrobora com Abelson (2010), apontando que o contato com linguagens visuais

pode estimular o aluno em atividades de programação, visto que em uma ação simples de exposição do tema, já é percebida a influência gerada sobre os discentes para a adoção deste paradigma.

6. Por último, a fim de identificar o nível de conhecimento dos alunos que já haviam cursado a disciplina de algoritmos, foi disponibilizado um desafio com 5 problemas, solicitando a implementação de programas utilizando qualquer linguagem de programação. Responderam a esta tarefa 11 alunos, todos adotaram a linguagem C para resolução dos problemas. A correção dos programas propostos demonstrou que 3 estudantes não conseguiram desenvolver nenhum dos algoritmos, 5 implementaram apenas um código sem erro e, apenas 1 aluno conseguiu construir respostas válidas para 2, 3 e 4 questões. Vale destacar que os problemas foram implementados com níveis de complexidade distintos, sendo que os 5 alunos que responderam uma única questão, o fizeram construindo programas para o problema mais simples, em que a aplicação deveria apresentar apenas uma mensagem na tela. Por meio da análise destes resultados foi possível identificar que, embora 56% dos estudantes já haviam cursado a disciplina de algoritmos ao menos uma vez, apenas 1 aluno conseguiu construir mais de 3 algoritmos válidos, demonstrando que o grupo realmente não possuía a capacidade de compor ações programáveis sequenciais, as quais são necessárias para construção de programas segundo Soloway & Ehrlich (1984). Esta afirmação permitiu enquadrar todos os 36 alunos em um mesmo nível, básico, de compreensão sobre tema algoritmos e programação.

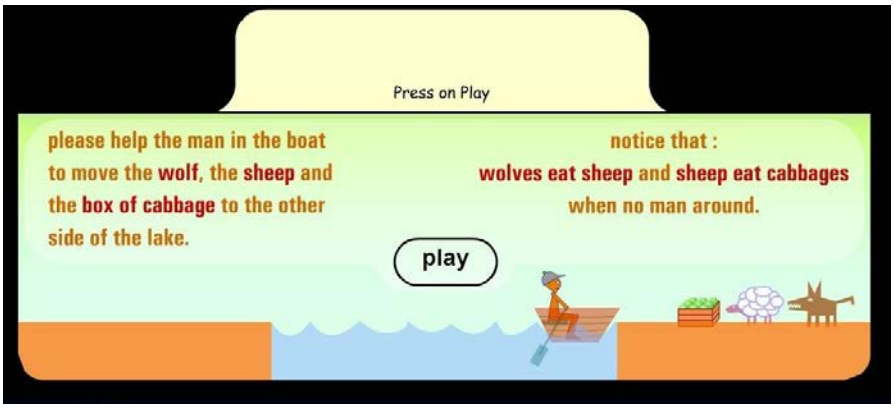
5.1.2. ANÁLISE DE RESULTADOS – CASO02

A segunda atividade no semestre (Caso02 – Quadro 28) foi baseada em um conjunto de 3 problemas de raciocínio lógico, implementados em flash, em que através da interação com o *software* os alunos deveriam, em um tempo pré-estabelecido (20 minutos para cada exercício), propor e descrever uma solução para cada uma das questões. O importante nesta tarefa, era que a composição disponibilizada pelo estudante, seguisse uma sequência lógica que permitisse qualquer um de seus colegas ler a resposta e solucionar o desafio. Esta prática vislumbrou estimular os estudantes a organizarem seu raciocínio, de forma a alcançar uma solução para o problema e organizá-la de forma inteligível. Os níveis de complexidade para

solução dos problemas aumentaram gradativamente a cada questão disponibilizada, sendo elas:

1. Problema 01 (ovelha, frutos e lobo): deve-se atravessar estes 3 elementos de uma margem a outra do rio, um de cada vez, contudo é necessário respeitar a regra de que a ovelha não pode permanecer sozinha na mesma margem que os frutos e, nem permanecer com o lobo (Figura 43).

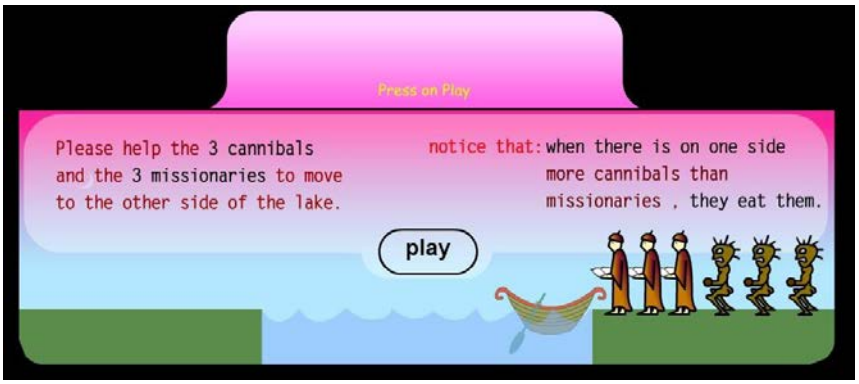
Figura 43-Prática de raciocínio lógico (ovelha, frutos e lobo)



Fonte: http://www.plastelina.net/

2. Problema 02 (monges, canibais): em uma margem de um rio encontram-se 3 monges e 3 canibais e, os mesmos devem ser transportados de uma margem para outra de um rio. A regra neste problema é que não se pode deixar um número menor de monges que canibais na mesma margem (Figura 44).

Figura 44-Prática de raciocínio lógico (monges e canibais)

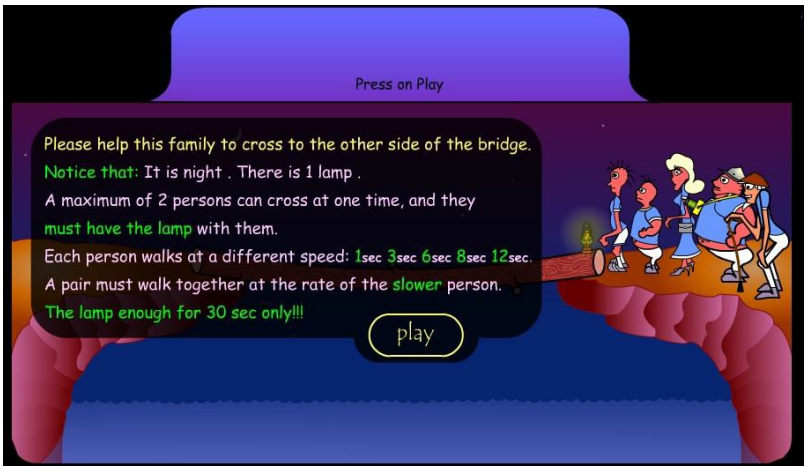


Fonte: http://www.plastelina.net/

3. Problema 03 (transposição da ponte): tem-se 5 indivíduos em um lado da ponte, cada pessoa consegue atravessar a ponte em um tempo específico, mas para ter sucesso é

necessário organizar a travessia de forma que o tempo total de transposição não ultrapasse 30 segundos (Figura 45).

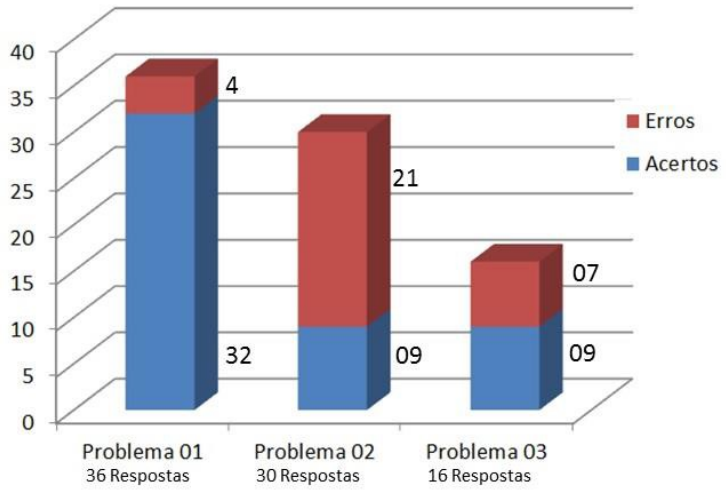
Figura 45-Prática de raciocínio lógico (transposição da ponte)



Fonte: http://www.plastelina.net/

Os resultados observados nesta atividade estão demonstrados no Gráfico 30, apontando as respostas nos 3 problemas. O primeiro exercício (problema 01) foi considerado o mais simples, visto que a lógica envolvida estava baseada no fato de transportar os elementos (ovelha, lobo e frutas) em uma sequência correta. A simplicidade ficou comprovada a partir do número de respostas (36), em que 32 alunos responderam de forma correta e apenas 4 não conseguiram construir uma solução válida. A segunda tarefa (problema 02) obteve 30 repostas, sendo 09 válidas e 21 incorretas, esta diferença de desempenho está vinculada diretamente ao aumento da complexidade do problema, visto que neste caso o aluno deveria trabalhar em uma solução para 6 elementos diferentes. Seguindo esta linha de interpretação, o problema 03 obteve o menor índice de repostas, apenas 16, com 09 acertos e 07 erros. Esta última tarefa exigiu do estudante uma capacidade de abstração, para desconsiderar a ordem óbvia de transposição da ponte e, adotar um raciocínio matemático para cumprir a tarefa. O número de acertos nas duas últimas questões foi o mesmo, contudo o total de respostas do segunda foi quase o dobro da terceira, apontando uma maior dificuldade na compreensão e elaboração de uma solução para o problema envolvendo lógica e matemática.

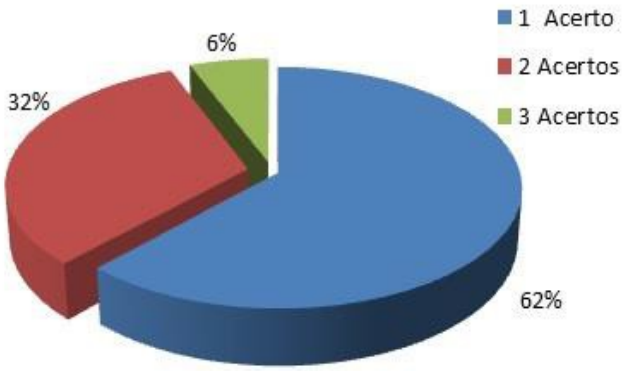
Gráfico 30-Respostas da atividade - Caso02



Fonte: Próprio autor.

Uma avaliação geral da atividade Caso02 apontou que a maioria dos participantes (62%) conseguiram acertar ao menos 1 dos 3 problemas (na grande maioria soluções para o problema 01). 32% obtiveram êxito na resolução de 2 questões e apenas 6% resolveram corretamente todas as tarefas, conforme o Gráfico 31.

Gráfico 31-Quantitativo de acertos da atividade - Caso02



Fonte: Próprio autor.

Alguns dos pontos observados nas respostas propostas pelos estudantes nesta tarefa:

1. Muitos alunos não conseguiram construir uma explicação clara sobre o como resolver os problemas, embora a solução descrita estivesse correta.

2. Na maioria dos casos propuseram soluções sucintas, válidas, porém que não permitiam de forma simples a sua interpretação.
3. Alguns estudantes utilizaram representações gráficas para demonstrar os passos a serem executados para completar os exercícios.
4. Em vários casos, o estudante iniciou uma resposta, de forma correta, contudo durante o desenvolvimento da mesma perdeu o foco, entregando um resultado inválido.
5. Inicialmente muitos dos alunos não tentaram responder a terceira questão, mas ao perceberem que se tratava de lógica matemática, se propuseram a tentar encontrar uma solução viável para o problema.

Nesta tarefa, todos os estudantes conseguiram acertar, ao menos uma resposta, contudo, constatou-se uma distância entre estes alunos e os discentes que resolveram todos os problemas. A observação sobre aspectos de motivação, na resolução das atividades e, a dificuldade dos alunos em formalizar uma resposta correta e entendível apontam para a necessidade da adoção, em sala de aula, de práticas que estimulem as capacidades nestes indivíduos de pensar e raciocinar logicamente.

5.1.3. ANÁLISE DE RESULTADOS – CASO03

Na sequência da disciplina foram apresentadas aos alunos, diferentes técnicas para auxiliar o estudante a construir soluções computacionais, focando no aprendizado da lógica de programação, quando foi realizado o terceiro caso (Caso03 – Quadro 28). Entendendo que os indivíduos apresentam uma maior capacidade de reconhecer algoritmos por meio de recursos visuais, adotou-se a representação gráfica por fluxogramas (BRUSILOVSKY, 1994; BENTLY & KERNINGHAN 1991; DAZZI *et al.* 2004; HOSTINS & RAABE, 2007), para que o aluno pudesse implementar respostas estruturadas e testar conceitos de maneira fácil e intuitiva, a partir da organização de símbolos com um fluxo e sequência lógica formalizados.

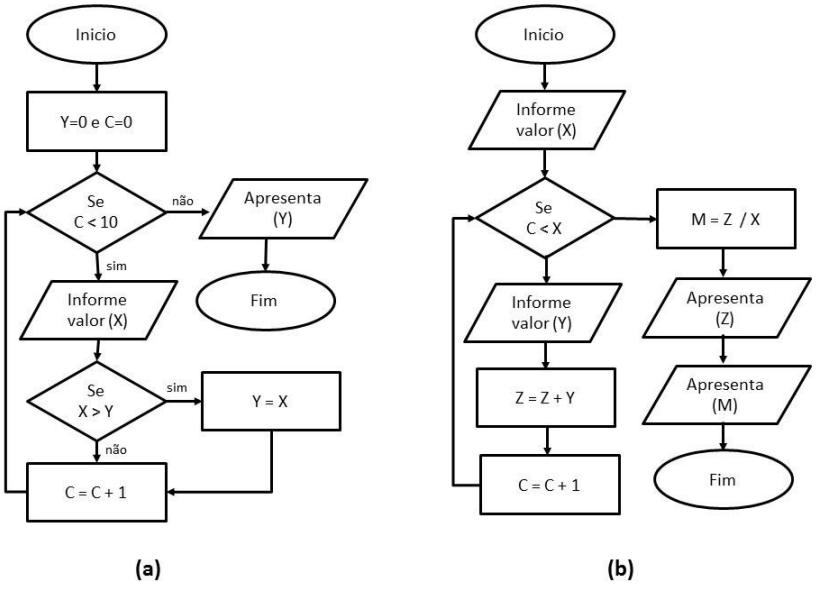
Scalan (1989) e Crews & Ziegler (1998) já defendiam em suas pesquisas que a utilização de fluxogramas estimula o discente na melhora do entendimento sobre a construção de algoritmos, reduzindo a taxa de erros na solução de problemas complexos e incentivando a confiança destes estudantes, o que leva a resolução de problemas simples de forma rápida e correta. A melhora no desempenho dos iniciantes em programação ao adotar este tipo de representação é atribuída por Gondim & Ambrósio (2008) a três fatores: sintaxe mínima dos

fluxogramas; sua representação universal, por meio de diagramas simples e; a facilidade de compreensão das estruturas desenvolvidas com esta ferramenta.

Considerando que a estrutura do fluxograma permite descrever a lógica de solução para um problema, sem considerar a complexidade de detalhes da linguagem de programação, é possível avaliar a capacidade do aluno em acompanhar o raciocínio de uma estrutura pronta, focando exclusivamente na lógica necessária para a composição de uma resposta válida. Este entendimento permitiu a composição da atividade Caso03, realizada após a apresentação dos conceitos, diagramas e diferentes exercícios durante as aulas de algoritmos. Nesta prática os estudantes deveriam interpretar 2 fluxogramas, com diferentes níveis de complexidade, conforme apresentado abaixo:

1. Questão 01: O sistema solicita 10 valores, armazenados em X, e apresenta como saída o maior valor informado (Figura 46a);
2. Questão 02: O sistema solicita que o usuário informe um valor (X), que indica o número de vezes que será solicitada a digitação de um número qualquer (Y). Um contador (C) valida a quantidade de elementos informados (C < X). Ao final a soma dos valores informados (Z=Z+Y) e a média (M=Z/X) são apresentados (Figura 46b).

Figura 46-Atividade sobre interpretação de fluxogramas – Caso03

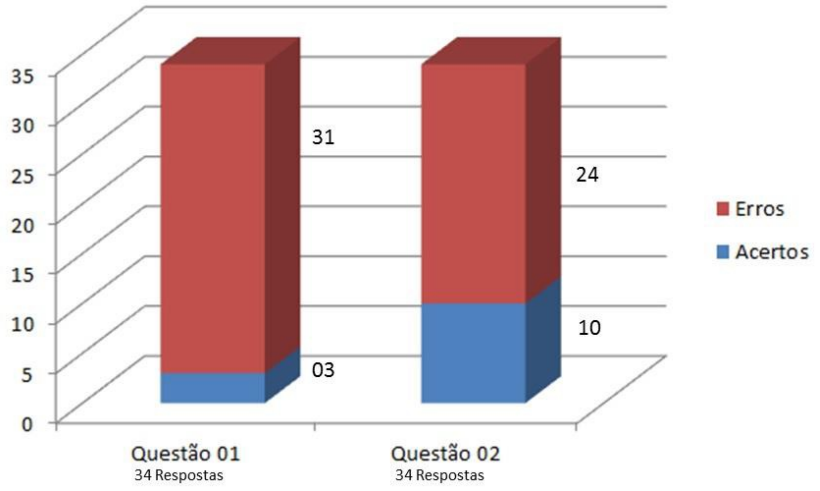


Fonte: Próprio autor.

Participaram da prática 34 alunos e, a análise da atividade foi realizada com base em dois critérios: acerto e erro. Sendo os erros computados em classes distintas: erros de interpretação (quando o aluno não compreendeu a finalidade do fluxograma) e erros de lógica (quando o estudante não conseguiu descrever o raciocínio lógico do fluxograma).

A análise simples de acertos e erros (Gráfico 32) demonstrou que apenas 9% dos alunos acertaram a primeira questão, enquanto 29% conseguiram responder corretamente a finalidade do segundo fluxograma. O paradoxo de que o maior número de alunos acertou a questão mais complexa se explica pela ordem das questões, ou seja, durante a primeira tarefa houveram vários momentos de diálogo entre o professor e os alunos, com explicações sobre a atividade. Neste ponto percebeu-se que os alunos apresentaram uma dificuldade em iniciar a resolução do exercício, após a saída do estado de inércia a prática fluiu normalmente.

Gráfico 32-Respostas da atividade - Caso03



Fonte: Próprio autor.

Para os resultados corretos assumiu-se que o discente demonstrou ter habilidade necessária para interpretar e propor soluções para os problemas disponibilizados, por meio das construções visuais apresentadas, atestando sua capacidade de abstração inerente neste tipo de técnica, que segundo Medina & Fertig (2005), são práticas úteis para o entendimento sobre algoritmos.

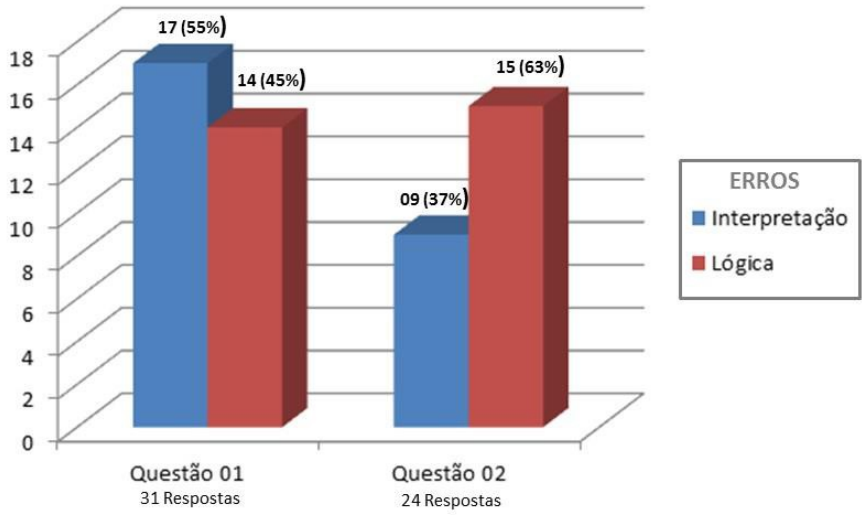
Quanto à avaliação dos erros, analisou-se qual a deficiência apresentada pelo aluno, se a mesma se deu em relação à compreensão da finalidade do fluxograma ou, se a deficiência baseou-se na incapacidade de abstração para compreender a base lógica da representação.

Sabendo que um algoritmo é um conjunto ordenado de passos executáveis, não ambíguos, apresentando um processo com início, meio e fim, esperou-se nesta etapa constatar como os novatos em programação refletem e interpretam as soluções demonstradas na atividade.

Interpretação, segundo os dicionários Michaelis e Aurélio, se refere ao ato de aclarar, explicar o sentido de, explicar a si próprio ou a outrem (MICHAELIS, 2015; AURÉLIO, 2015). A partir desta concepção, um dos enquadramentos de erros nesta atividade, foi imputado à falha na interpretação, ou seja, a resposta apresentada não explicou de forma correta a finalidade do fluxograma. É importante esta percepção, pois como afirmam Kelleher & Pausch (2005) e Tanrikulu & Schaefer (2011), alunos iniciantes em programação necessitam aprender a identificar e interpretar a estrutura de um problema para propor soluções válidas para o mesmo.

Desta forma, ao analisar os erros de interpretação, a fim de identificar se o estudante demonstrou a capacidade de organizar os fragmentos que representam ações sequenciais em ambos os fluxogramas, percebeu-se que 86% dos alunos erraram, ao menos, uma das questões. Os erros de interpretação compuseram um total de 47%, sendo que 17 discentes (55%) não responderam corretamente à questão 01, enquanto 09 estudantes (38%) não acertaram a segunda (Gráfico 33). A redução nos erros de interpretação da primeira questão para a segunda demonstrou que, embora a tarefa 2 apresentasse um maior grau de dificuldade, a prática inicial estimulou os alunos a examinar cuidadosamente o problema, conseguindo assim descrever a função do segundo fluxograma de maneira clara, o que ficou explícito na forma como as respostas foram apresentadas na segunda questão.

Gráfico 33-Análise de Erros - Caso03



Fonte: Próprio autor.

Compreendendo que o termo lógica se refere à ciência do raciocinar, ou seja, o modo de pensar, coerentemente, tal como de fato se exerce (AURÉLIO, 2015), podendo ser esta, uma lógica simbólica a qual entende-se como a ciência do desenvolvimento calcada sobre a representação de princípios lógicos, mediante símbolos, postulados e regras que permitem deduções exatas (MICHAELIS, 2015), buscou-se a partir deste conhecimento um segundo padrão para a análise de erros dos alunos na atividade 03. Sobre este enfoque observou-se (Gráfico 37) que 45% (14 estudantes) apresentaram dificuldades de compreensão sobre a lógica envolvida na primeira questão e 63% em relação à tarefa 02. A majoração em 18% nos erros, relacionados à segunda questão, foram justificadas pela dificuldade dos estudantes em entender dois pontos específicos no fluxograma:

- O fato do sistema utilizar como base, para estrutura de repetição, um valor informado pelo próprio usuário;
- O cálculo da média dos valores informados;

O baixo nível de abstração, demonstrado pelos alunos ao identificar as funcionalidades da estrutura de repetição, é descrito por Jenkins (2002) como fator de insucesso em disciplinas de programação. A prática sobre estes temas, assim como a compreensão da matemática, adjacente ao processo de resolução de problemas são características que devem ser aprimoradas por iniciantes em programação, a fim de construir as capacidades necessárias para se tornarem programadores (SLOANE & LINN, 1988).

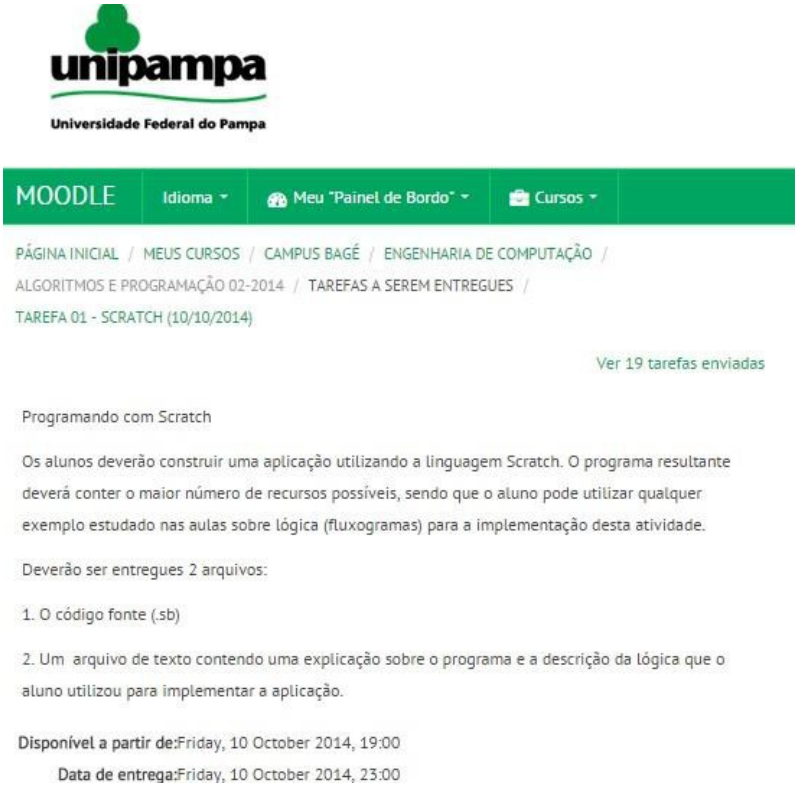
5.1.4. ANÁLISE DE RESULTADOS – CASO04

Após os estudos sobre lógica de programação os alunos passaram a ter contato com a linguagem Scratch. Visto a facilidade de utilização deste ambiente (RESNICK, 2009), em apenas 4 períodos letivos de 55 minutos, os estudantes já estavam construindo programas simples. Contudo o conteúdo explorado nestas primeiras aulas, segundo o plano de ensino (Figura 46), contemplou apenas o reconhecimento básico da estrutura do ambiente e seus comandos.

O Caso04 (Caso04 – Quadro 28) teve como objetivo explorar a capacidade dos alunos em implementar um programa simples, utilizando os recursos da linguagem Scratch. O tema para elaboração da aplicação era livre, contudo o sistema deveria apresentar uma estrutura lógica clara, sendo funcional e com o maior número de comandos possíveis. A descrição da

atividade é apresentada na Figura 40. Além da aplicação os alunos deveriam entregar um arquivo contendo uma explicação do programa e a descrição da lógica utilizada para a sua construção. Esta foi uma prática realizada em sala de aula e os alunos deveriam enviar as respostas via Moodle.

Figura 47-Programando com Scratch – Caso04



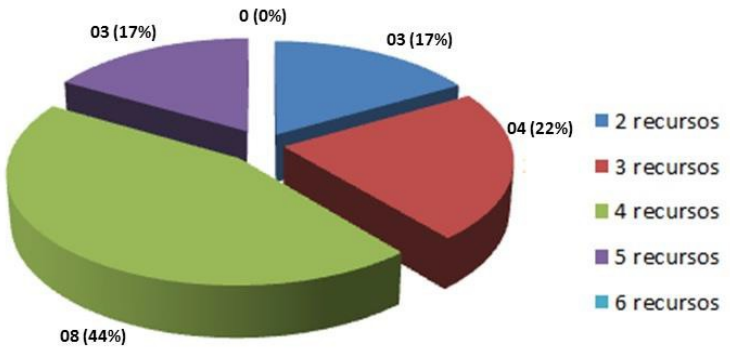
Fonte: Próprio autor.

Participaram do Caso04 19 alunos, que propuseram diferentes soluções para a tarefa, sendo que neste grupo apenas 2 entregaram programas incorretos (não rodavam). A avaliação das respostas deste exercício baseou-se na observação do quantitativo de recursos utilizados e descrição do raciocínio lógico.

Quanto ao conjunto de recursos foram analisadas a presença das seguintes estruturas nas aplicações: Comandos de Entrada e Saída, Variáveis, Estruturas de Condição, Estruturas de Repetição, Equações Matemáticas e Recursos Multimídia. A investigação dos resultados apontou que 17% dos programas continham 2 recursos distintos, 22% utilizaram 3 recursos, 44% empregaram 4 recursos e 17% adotaram 5 recursos na aplicação, no entanto nenhuma das respostas entregues continham os 6 recursos analisados (Gráfico 34).

Compreendendo que no momento da atividade os alunos conheciam apenas o funcionamento da linguagem e não seu conjunto de recursos, é possível afirmar que o percentual de adoção de 4 recursos demonstrou que os alunos realmente se interessaram pela programação com a linguagem e, desta forma exploraram as funcionalidades do Scratch com êxito, corroborando com o fato desta ser uma linguagem acessível e que possibilita um enriquecimento no processo de aprendizagem de algoritmos e programação (FAL & CAGILTAYC, 2013).

Gráfico 34-Quantitativo de recursos utilizados nas aplicações – Caso04



Fonte: Próprio autor.

A relação dos recursos e o percentual de emprego dos mesmos, nos programas entregues nesta atividade, estão expostos no Quadro 26, através do qual é possível identificar quais estruturas foram utilizadas com maior frequência nas aplicações:

- O uso de comandos de entrada e saída despontam como instruções mais adotadas (79% - 15 aplicações), o que é justificado pelo caráter de interação com o usuário, proporcionado por este tipo de comando, além da sua facilidade de utilização no Scratch, o que estimula o estudante a utilizar tais códigos, conforme descrito por eles durante a realização da atividade.
- Estruturas de Condição e Recursos Multimídia apareceram empatadas, sendo utilizadas em 13 aplicações (68%). Este quantitativo, para ambas as estruturas, está diretamente relacionado ao fato de que os alunos foram incitados a construir soluções interativas para casos reais, o que direciona ao uso de sons e imagens, assim como impele a necessidade de validação dos dados passados pelos usuários.

- As Estruturas de repetição foram encontradas em 42% dos programas (08 respostas), principalmente nas soluções que tinham como objetivo a movimentação dos objetos no palco do Scratch. Contudo dois programas utilizaram este tipo de comando para manter um ciclo de interação com o usuário.
- Equações matemáticas foram observadas em 04 das aplicações entregues (21%), nos casos em que o aluno propôs soluções com Scratch para cálculo de área de um quadrado, cálculo do IMC (Índice de Massa Corporal) e fórmula de Bhaskara.
- Por último, 05 programas (26%) foram implementados com base nas atividades realizadas em aulas anteriores, ou seja, os alunos demonstraram a capacidade de traduzir soluções na forma de fluxograma para aplicações com o Scratch.

Quadro 30-Recursos utilizados em programas – Caso04

Recursos Observados	Programas	
Comandos de E/S	15	79%
Variáveis	12	63%
Estrutura de Condição	13	68%
Estrutura de Repetição	08	42%
Equações	04	21%
Recursos Multimídia	13	68%
Baseado Atividades Anteriores	05	26%

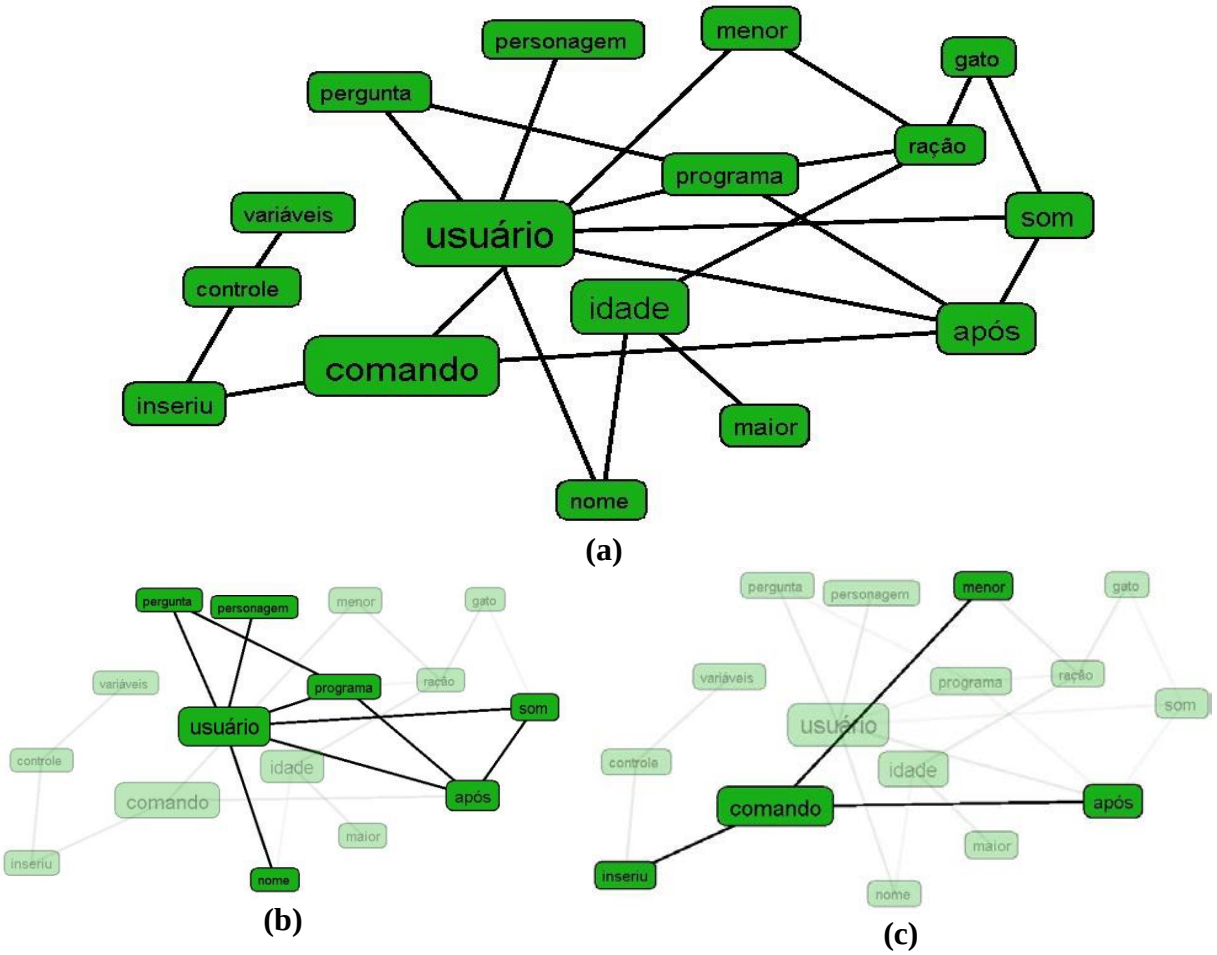
Fonte: Próprio autor.

A segunda etapa de investigação das respostas dos alunos, para o Caso04, contemplou uma análise qualitativa das narrativas dos estudantes sobre a lógica utilizada na construção dos programas. Embora esta tenha sido uma das primeiras atividades de programação, os alunos apresentaram programas logicamente corretos e com um razoável grau de maturidade.

Para a observação das explanações adotou-se a técnica de mineração de dados, que segundo Hearst (1999), Tan (1999) e Azevedo *et al.* (2009) permitem a identificação e extração das características representativas das informações em textos, a partir de métodos oriundos de áreas específicas como recuperação de informações, linguística computacional e estatística, ou seja, permite o reconhecimento de informações válidas armazenadas como dados não estruturados (FELDMAN & SANGER, 2006). Com base no exposto, a fim de

verificar a capacidade dos alunos em descrever as aplicações e expor o raciocínio na construção dos programas, adotou-se a ferramenta de mineração textual Sobek (REATEGUI *et al.* 2011), a qual gera grafos de relação entre conceitos extraídos de textos, permitindo desta forma indicar a associação entre os termos descritos pelos discentes. A Figura 48 apresenta o processo de mineração textual realizado sobre as respostas descritivas do Caso04, em que são expostas as recorrências de termos relevantes as observações nesta etapa do estudo.

Figura 48-Mineração de texto referente à resposta descritiva – Caso04



Fonte: Próprio autor.

A interpretação do grafo gerado com o Sobek (Figura 41a) permite discernir que as palavras “usuário” e “comando” diferem das demais, pelo destaque no seu tamanho, ocasionado por se tratarem de termos com maior recorrência nas descrições dos alunos e, também pelo fato de possuírem mais relações com outras sentenças.

O termo “usuário” é encontrado no texto 28 vezes (Figura 41b), enquanto a palavra “comando” possui uma incidência em 25 momentos distintos nas respostas (Figura 41c).

Estas observações apontam para o fato de que os estudantes, neste momento, já iniciaram a se apoderar dos conceitos e vocábulos comumente utilizados por desenvolvedores, fato positivo, pois esta percepção denota uma convergência com a teoria de Berlinsk (2002), que descreve a necessidade do programador entender um vocabulário simbólico e instruções precisas para a proposta de soluções algorítmicas. Para Raabe (2005) a prática demonstrada pelos estudantes está relacionada diretamente à construção do raciocínio lógico e alinhada ao processo de ensino e de aprendizagem de programação.

Visto o grau de liberdade desta tarefa, as argumentações dos estudantes apresentaram diferentes contextualizações, desde explicações detalhadas sobre a finalidade e desenvolvimento do programa, até frases sucintas, sem nenhum nível de detalhamento, conforme é mostrado no Quadro 31.

Quadro 31-Recursos utilizados em programas – Caso04

Aluno	Resposta
A01	*Foi realizado para a Tarefa 1 em programação por Scratch a Fórmula de Baskhara. O programa inicialmente pergunta para o operador os valores das variáveis "A,B e C", observando o detalhe de que o coeficiente da variável "A" tem de ser maior que 0 para que possa ser calculada a fórmula. Ao serem dados valores para "A, B e C" são armazenados em suas respectivas variáveis. Após foi criado uma variável delta para realizar a equação das vaiáveis. OBS: DELTA tem de ser maior que 0 se não as raízes são imaginarias. Utiliza-se -1*B para representar o –B, somando a raiz quadrada do delta, para duas variáveis X' e X''. Após é apresentada a resposta.*
A02	*Para a construção de um programa simples que interagisse diretamente com o usuário, usei diversos recursos disponibilizados pelas ferramentas do programa Scratch. Iniciamos com a construção lógica perguntando ao usuário seu nome, através do operador "aparência". Usei, após isso, a ferramenta "sensores", para obter do usuário um lado de um quadrado qualquer, afim do personagem equacionar os valores de ÁREA e VOLUME, separando este dado em uma variável denominada "VLADO", usando o parâmetro "variáveis". O cálculo se deu através da ferramenta "operadores", multiplicando o valor armazenado na variável. Após mais uma interação com o usuário, usou-se a ferramenta "som" e "aparência", afim de fazer o personagem executar um som e trocar o sprint para o "traje2". Após esta primeira construção com o personagem 1, induzimos o usuário a clicar no segundo personagem a fim de iniciar outra interação. Nesta etapa utilizei as ferramentas para que o usuário fosse submetido a um sorteio, introduzindo um número entre 0 e 10, onde através das ferramentas "variáveis", utilizada para armazenar o numero introduzido pelo usuário, e "operadores", afim de comparar o número sorteado pela máquina com o dado emitido. Após isso, a personagem induz o usuário a clicar no primeiro personagem e some, através da "aparência". Neste terceiro ato, o personagem usa um loop de ações de "movimento" e encerra o programa.*
A03	*Objeto (Gato) interagindo com usuário. Usuário interagindo com objeto. Gato faz 2 perguntas ao usuário: Qual seu nome? Qual sua idade? Ele usa a idade que o usuário informa para exibir a própria idade.*
A04	*O usuário pergunta o nome e a idade, depois da boa noite*

Fonte: Próprio autor.

Os exemplos apresentados são uma amostra de 4 descrições distintas, entregues nesta tarefa, que se percebe de maneira clara o nível de detalhamento apresentado pelos alunos identificados como A01 e A02. Estes buscaram explicitar o funcionamento das suas

aplicações, com isso delinearam quase todas as etapas da execução do *software*, demonstrando assim uma capacidade de organizar seus pensamentos e expressá-los na forma computacional, concebendo os passos a serem seguidos para alcançar o retorno esperado. Este nível de abstração, alcançado na resolução de um problema, é definido por Dijkstra (1982) como a habilidade de raciocinar logicamente.

A descrição do aluno A03 demonstra-se sucinta e pontual, embora siga uma sequência natural e válida. O intuito da aplicação proposta foi solicitar dados de nome/idade do usuário e apresentar na tela apenas a idade digitada. É plausível entender que, neste momento na disciplina, os discentes estejam descobrindo as diferentes maneiras de se construir soluções computacionais, através de uma linguagem de programação. O estímulo de explorar as suas capacidades de propor algoritmos, validá-los, entender seu funcionamento e observar os resultados pode ser considerado um elemento motivador para o processo de aprendizagem e formação de novos programadores (SANTIAGO & DAZZI, 2003).

Como último exemplo, tem-se a resposta do aluno A04, o qual entregou um programa correto, porém com um número mínimo de recursos, alinhado à descrição apresentada para tal aplicação. Várias teorias podem ser elencadas para justificar o baixo rendimento deste aluno, contudo o contato com o discente em diferentes atividades em sala de aula permitiu apurar que o mesmo estava matriculado na disciplina algoritmos apenas para cumprir créditos, a fim de solicitar remoção para outro curso, que não possui programação no currículo. Este fato se concretizou antes do final do semestre, culminando com o abandono da disciplina.

A instigação realizada sobre esta atividade, considerando a relação dos recursos utilizados nos programas entregues, assim como a análise da descrição do raciocínio para o desenvolvimento destes, revelou que a organização dos conteúdos propostos nesta pesquisa favoreceu os indivíduos na construção da abstração necessária à concepção do raciocínio lógico. O RBC foi validado e identificado ao observar o resultado das tarefas de diferentes alunos, que implementaram suas soluções com base em casos realizados anteriormente na disciplina e, pelo conjunto de termos comumente utilizados em aula, os quais foram incorporados às explicações fornecidas sobre os programas.

5.1.5. ANÁLISE DE RESULTADOS – CASO05

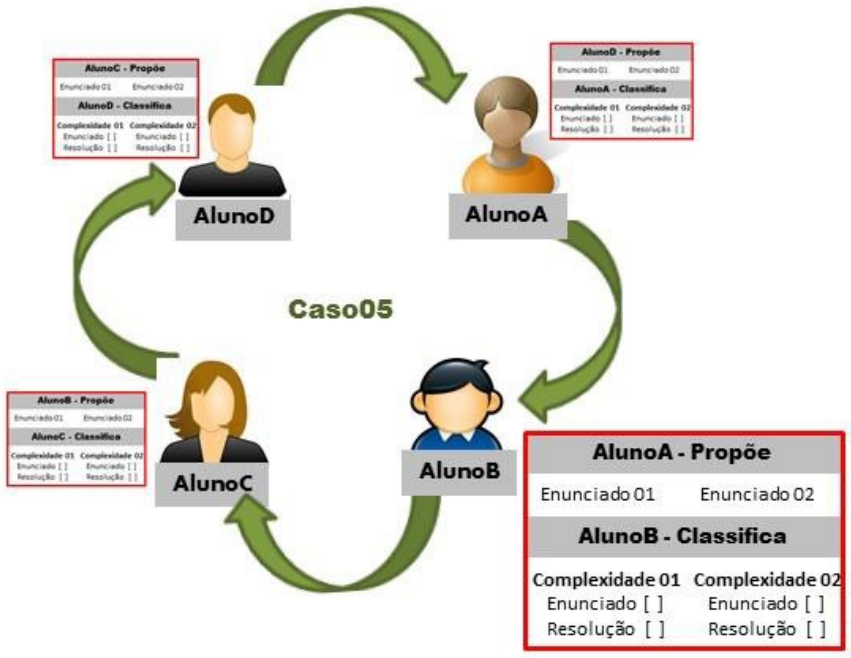
O Caso05 (Caso05 – Quadro 28) contemplou um exercício voltado para um tema de extrema importância no processo de aprendizagem de algoritmos, a interpretação de

enunciados. Segundo Souza (2011) a dificuldade encontrada pelo estudante em abstrair e descrever soluções para problemas inicia pela tarefa de entender a sua descrição, definir seus requisitos, traçar conjecturas, identificar variáveis, recursos lógicos e propor estruturas de programação factíveis, sendo todo este processo baseado na compreensão dos enunciados disponibilizados (IEPSEN, 2013a).

Assumindo a existência desta dificuldade a tarefa 05 contemplou uma atividade invertida (HERREID & SCHILLER, 2013), em que os alunos propuseram dois enunciados diferentes, com tema livre. A descrição dos enunciados deveria permitir a construção de sistemas, para um problema específico, com a utilização da linguagem Scratch, tendo como base os recursos já estudados em sala de aula.

Após a problematização das questões, os enunciados eram entregues a outro aluno, em um ciclo, a fim de que todos os estudantes permutassem suas questões (Figura 49).

Figura 49-Infográfico da atividade – Caso05



Fonte: Próprio autor.

Após receber a descrição do problema, o aluno realizou 3 ações distintas:

1. Definir o nível de complexidade percebido no enunciado. A mensuração deste deveria seguir uma escala likert de 5 pontos, tendo como referência o Quadro 32.
2. Implementar um programa com Scratch, atendendo as demandas e requisitos descritos no enunciado.

3. Apresentar o nível de complexidade encontrado no desenvolvimento do sistema, seguindo também as recomendações do Quadro 32.

Quadro 32-Escala para definição de complexidade – Caso05

Escala	Descrição
1	Muito Simples (MS)
2	Simples (S)
3	Razoável (R)
4	Complexo (C)
5	Muito Complexo (MC)

Fonte: Próprio autor.

Para garantir a equidade das questões que foram propostas, antes do aluno entregar a tarefa ao seu colega, o professor realizou uma revisão nos exercícios, com o intuito de identificar a viabilidade no desenvolvimento da aplicação. Detalhes gramaticais ou relacionados ao nível de dificuldade do problema não foram considerados neste momento.

Participaram da atividade 28 estudantes, dos quais apenas 3 (11%) não conseguiram criar uma aplicação para atender à primeira questão e, 4 (14%) não responderam a segunda. Cabe salientar que os mesmos discentes que não conseguiram responder a questão 01 também não implementaram a questão 02.

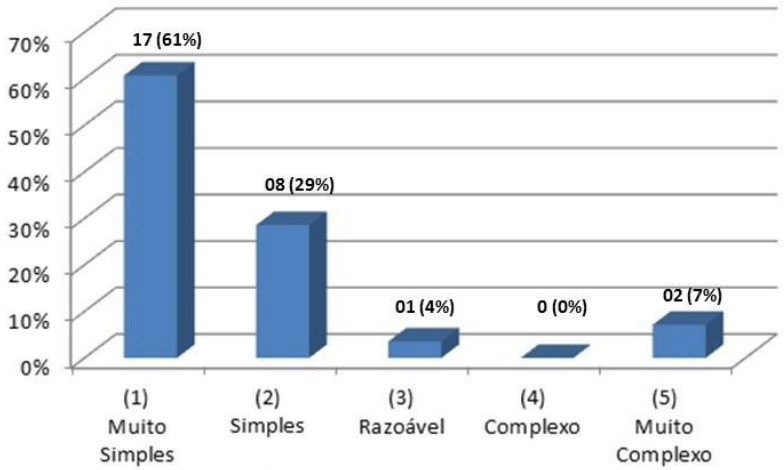
A apreciação das respostas dos alunos demonstrou que 61% dos enunciados referentes à primeira questão foram considerados muito simples, 29% simples, 4% razoáveis e 7% muito complexos (Gráfico 35). Todos os estudantes do grupo que definiram como alto (muito complexo) o grau de dificuldade no entendimento da questão 01, conseguiram implementar o *software* solicitado, o que denota 2 situações convergentes:

1. Os alunos que propuseram as questões não conseguiram expor as demandas do sistema.
2. Os estudantes que receberam a atividade não conseguiram interpretar facilmente os requisitos.

Ao comparar o apontamento sobre a complexidade da resolução (construção do programa) destes mesmos discentes, percebeu-se que em todos os casos a classificação para a

resolução permaneceu com índice abaixo em relação ao enunciado, ou seja, programar foi mais simples do que entender o que estava sendo solicitado.

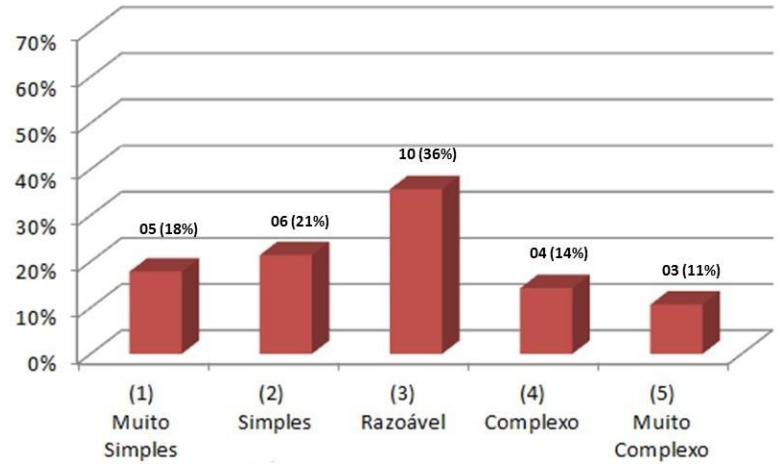
Gráfico 35-Classificação dos enunciados da questão 01 – Caso05



Fonte: Próprio autor.

Em relação à classificação do nível de dificuldade de implementação das aplicações, na primeira questão (Gráfico 36), 18% dos discentes definiram como muito simples o processo de programação, 21% simples, 36% razoável, 14% complexo e 11% muito complexo. A maioria dos estudantes, 10 indivíduos, apontou um nível de dificuldade razoável, isto significa que embora 17 estudantes tenham definido o enunciado como simples, o desenvolvimento das aplicações não foi tão fácil.

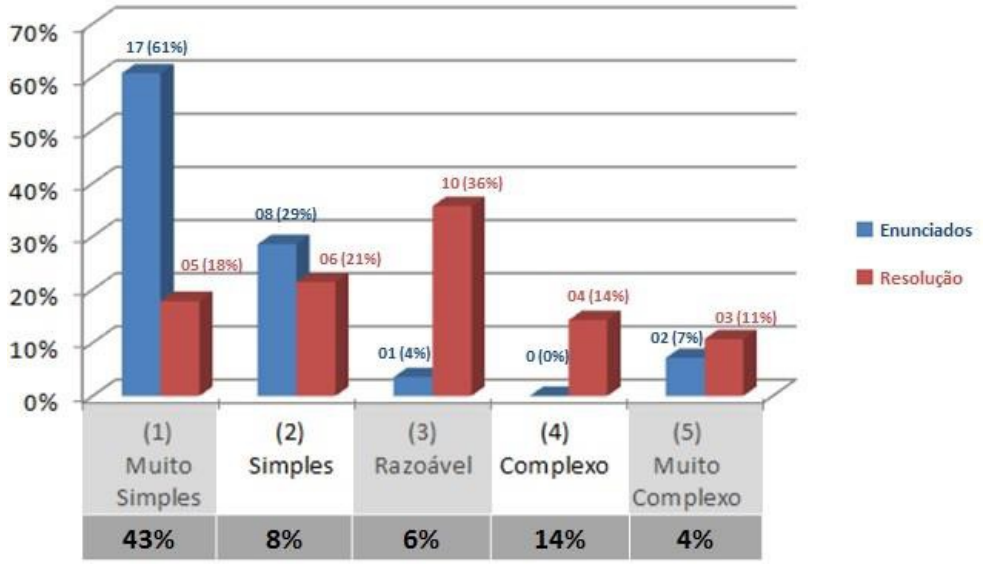
Gráfico 36-Classificação da resolução da questão 01 – Caso05



Fonte: Próprio autor.

O Gráfico 37 apresenta uma comparação entre as classificações dadas aos enunciados e a complexidade na implementação dos programas. A diferença média ficou em 15 pontos percentuais, com destaque à distância observada na definição de dificuldade Simples, em que os alunos especificaram 61% dos enunciados como de fácil entendimento, no entanto, a classificação de implementação simples chegou apenas a 18%, uma diferença de 43%. Este fato demonstra que embora o estudante tenha capacidade de compreender os requisitos, a construção da aplicação ainda não pode ser considerada uma atividade dominada por eles. A discrepância entre os demais itens ficou em 8% para a classificação muito simples, 6% para razoável, 14% para complexo e 4% para muito complexo. Este último, com a menor distância entre as pontuações, representa os alunos com maior dificuldade na disciplina, em ambos os aspectos analisados.

Gráfico 37-Comparação entre as classificações de enunciados e resoluções da questão 01 – Caso05

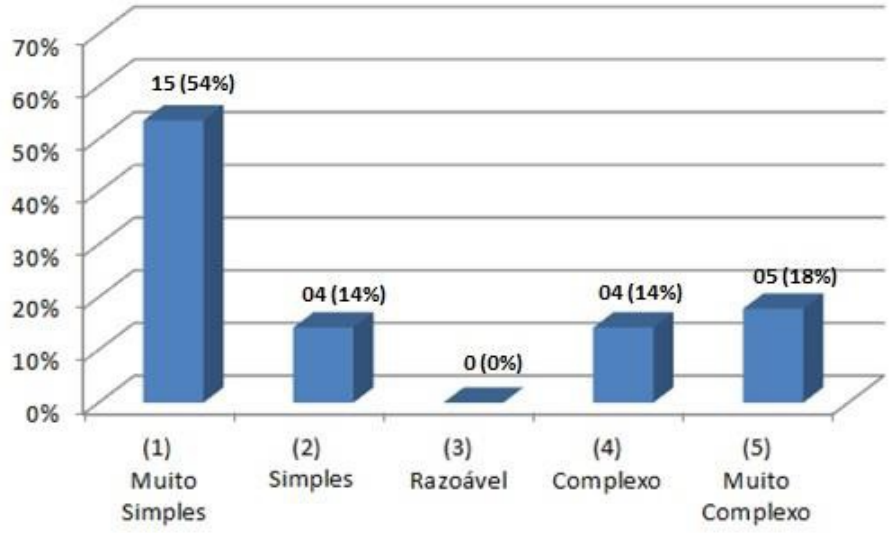


Fonte: Próprio autor.

Por fim, uma particularidade pertinente é percebida ao contabilizar as classificações, pois no caso dos enunciados, 90% do grupo definiu uma complexidade muito simples e simples, enquanto o montante de percentual relacionado às resoluções ficou em 75% para os mesmos itens, com inserção da especificação razoável, que se destacou das demais. Esse fato mostra que a grande maioria dos alunos reconheceu os enunciados e implementou o *software* nesta questão.

A segunda questão seguiu a mesma formatação da primeira, porém os alunos foram incitados a elevar o nível de complexidade na definição dos problemas, ou seja, que os requisitos sugeridos exigissem um maior número de comandos em uma lógica aprimorada, a fim de dinamizar a prática. É percebido que os índices definidos nesta questão ficaram próximos, em ambos enunciados (questão 01 e 02) para a classificação muito simples, com 54% (Gráfico 38). A definição de simples ficou em 14%, complexo também 14% e muito complexo 18%. A ampliação no nível de dificuldade dos enunciados propostos ficou nítida ao observar os valores dos itens 4 e 5, com um aumento considerável em relação a questão 01.

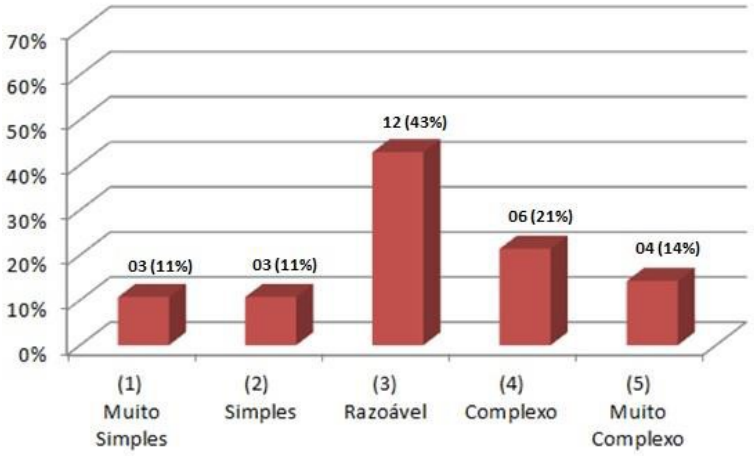
Gráfico 38-Classificação dos enunciados da questão 02 – Caso05



Fonte: Próprio autor.

A impressão dos alunos em relação à resolução da questão 02, também se assemelhou a questão 01, contudo, visto que a complexidade do enunciado foi incrementada, esperava-se um resultado com valores de classificação tendendo aos itens complexo e muito complexo. Ao contrário disto, a complexidade definida para a implementação das aplicações teve um maior percentual para o item razoável (43%). Os itens 4 e 5 ficaram em segunda e terceira posições com 21% e 14% respectivamente. Simples e muito simples apresentaram níveis de 11%, conforme o Gráfico 39. Considerando que 4 alunos não conseguiram entregar esta tarefa, é viável supor que a adversidade dos enunciados/resoluções não foram problemas de vulto, que atrapalhassem a conclusão das tarefas, isto é, esta tarefa demonstrou que os discentes apresentaram um avanço na capacidade de resolver problemas.

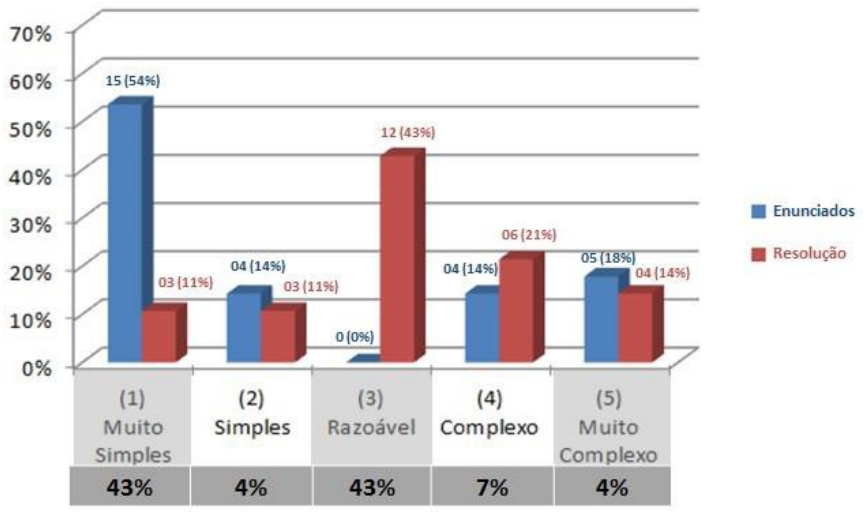
Gráfico 39-Classificação da resolução da questão 02 – Caso05



Fonte: Próprio autor.

A distância na interpretação da descrição dos requisitos e construção do *software*, na questão 02, é apresentada no Gráfico 40, em que se identifica um grande intervalo (43%) entre a classificação do enunciado e a resolução no item 01 (muito simples), o que evidencia, como na questão 01, que os alunos conseguem propor e entender requisitos para o desenvolvimento de *software*, mas apresentam dificuldade na tarefa de programar. A discrepância na definição de razoável, para a implementação, também é observada nesta segunda atividade, com um percentual de 43%, mostrando que no primeiro contato com o problema os discentes tem dificuldade em apontar um enunciado como razoável, contudo é o maior índice para a resolução.

Gráfico 40-Comparação entre as classificações de enunciados e resoluções da questão 02 – Caso05



Fonte: Próprio autor.

Para encerrar a análise deste caso, uma demanda pertinente à formação do raciocínio do aluno deve ser esclarecida, a qual está calcada na identificação da capacidade dos discentes em correlacionar os requisitos apresentados sobre um problema, com o nível de dificuldade para a sua resolução de forma computacional. Com o objetivo de obter esta informação propôs-se um método para reconhecer as competências do aluno, a partir das classificações sugeridas para o enunciado e resolução nas questões 01 e 02. Cabe ressaltar que competência, neste estudo, se refere às habilidades e conhecimentos que interferem diretamente nas ações do indivíduo (FLEURY & FLEURY, 2001).

A equação 04 serviu de base para o modelo, a qual fornece um índice gerado pela subtração do nível de complexidade, definido para a implementação do *software*, pela complexidade estipulada ao enunciado.

(4)

$$\underset{\substack{\text{Diferença}\\ \text{Complexidade}}}{DC} = \text{Complexidade Enunciado} - \text{Complexidade Resolução}$$

Os resultados obtidos com a equação apresentam valores entre -5 e 5, em que o ideal é percebido nas mediações do valor zero, ou seja, o indivíduo com índice nesta região demonstrou aptidão em classificar enunciado e resolução em um mesmo nível de dificuldade, conforme é demonstrado na Figura 50.

Figura 50-Modelo para avaliação de competência do aluno para compreensão de enunciados – Caso05

Fonte: Próprio autor.

Ao considerar que um programador experiente é capaz de observar a demanda de um problema e sugerir sua solução, considerando as particularidades deste (JOHNSON & SLOWAY, 1984), espera-se da mesma forma que o estudante de algoritmos desenvolva

competências semelhantes. A partir deste entendimento o método sugere o enquadramento dos resultados da equação em 3 níveis: Ideal, quando a diferença nas complexidades não ultrapassa 1 ponto, o que significa que o parecer do estudante ao interpretar um enunciado está alinhado com a complexidade na programação da solução; Aceitável, a diferença entre percepção de enunciado e desenvolvimento da aplicação possui no mínimo 2 pontos de distância, o que significa uma deficiência no entendimento dos requisitos ou falta da prática necessária para a implementação do *software*, e; Insuficiente, estando enquadrados nesta classe os estudantes que apontaram índices de enunciado e resolução totalmente desconexos, possivelmente causados por erro de apreciação na descrição do problema, capacidade de raciocínio lógico diminuto ou desinteresse pela atividade.

Ao observar o desempenho individual dos alunos, com a aplicação do método, em ambas as questões (Figura 51), verifica-se um grupamento dos índices gerais entre as classes -4 (insuficiente) com 10%, 9% e 17% nos itens -3 e -2 (aceitável esquerda) respectivamente, 26% nos itens -1 e 0 (ideal) e 12% no item 2 (aceitável direita). A verificação dos quantitativos, nas duas questões, apontou que em 30 respostas os estudantes evidenciaram competências necessárias para compreender e construir programas com índices entre -1 e 1, 22 respostas mostraram que os alunos obtiveram um desempenho aceitável (índices -3, -2 e 2) e, apenas 5 respostas permaneceram na classe insuficiente (índice -4).

Figura 51-Modelo para avaliação de competência do aluno para compreensão de enunciados – Caso05

	INSUFICIENTE		ACEITÁVEL		IDEAL			ACEITÁVEL		INSUFICIENTE	
	-5	-4	-3	-2	-1	0	1	2	3	4	5
Questão 01		7%	10%	21%	24%	31%		7%			
Questão 02		14%	7%	14%	28%	21%		17%			
Geral		10%	9%	17%	26%	26%		12%			

Fonte: Próprio autor.

Os resultados auferidos no estudo das respostas se complementaram, possibilitando constatar que a maioria dos alunos apresentou um nível de habilidade satisfatório para a resolução da atividade invertida. A evolução no raciocínio lógico e Pensamento Computacional tornou-se explícita, comprovado pelo fato de 90% dos discentes se enquadrarem nas classes ideal e aceitável, mostrando que este universo é suficientemente capacitado a compreender enunciados e implementar sistemas nesta fase da disciplina.

5.1.6. ANÁLISE DE RESULTADOS – CASO06

O sexto caso avaliado (Caso06 – Quadro 28), uma atividade formativa, com propósito de estimular o senso crítico do aluno, sua aptidão em analisar programas prontos e de apontar a existência ou não de erros, que são consideradas habilidades necessárias para o processo de construção do raciocínio lógico, fundamentais para a abstração e resolução de problemas (BARCELOS *et al.* 2009).

Compreendendo que o estudante é o único responsável pela sua aprendizagem e, que este processo está calcado sobre elementos como motivação, maturidade, capacidade de perceber a realidade e o funcionamento de suas estruturas cognitivas, conforme apontado por Bord (1996), esta tarefa incitou os alunos a ultrapassar o obstáculo gerado pela característica básica do ser humano, de pensar de forma lógica e individual, ou seja, entender que a resolução apresentada por um programador para um determinado problema, irá seguir seu estilo de pensamento, o qual pode ser diferente para cada caso ou situação. Isto significa que podem existir diferentes soluções para um mesmo problema, sendo esta uma atividade subjetiva, visto que o raciocínio lógico é particular (DE JESUS, & BRITO, 2009), contudo diferentes respostas podem ser consideradas válidas.

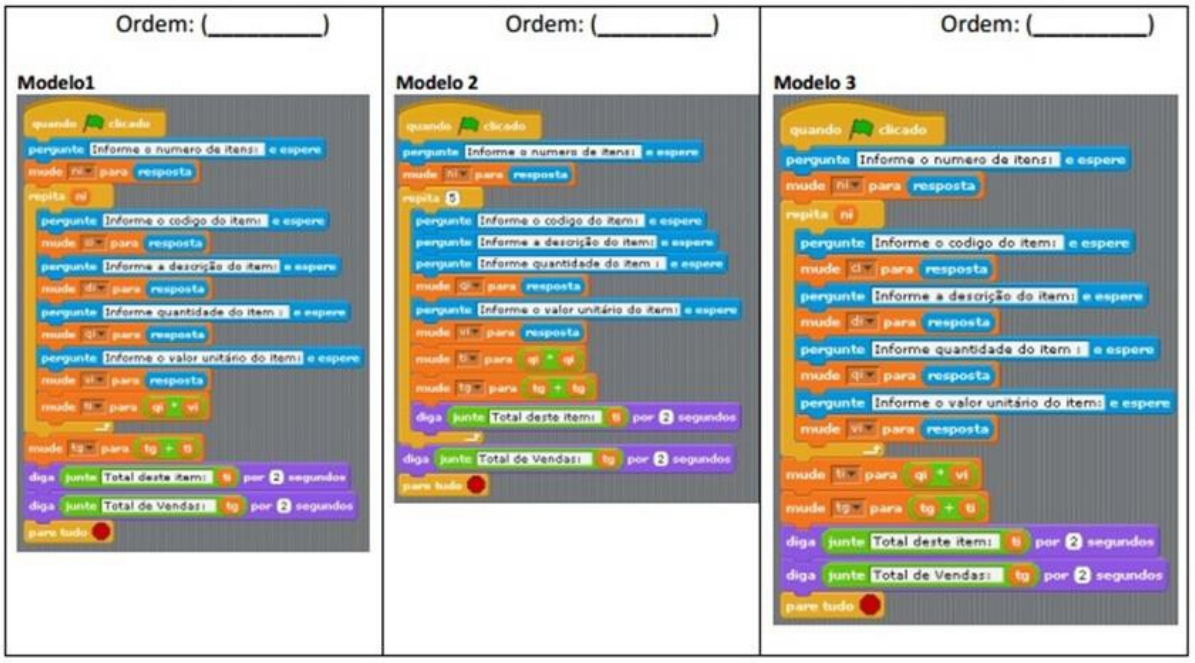
Para obter êxito nesta tarefa, o discente deveria considerar as conjecturas de Duval (2003) e Delgado *et al.* (2005) sobre a compreensão da existência de múltiplas representações para a solução de um problema e, que o entendimento de uma construção passa pela capacidade de construir analogias que apoiem na interpretação do resultado proposto.

Desta forma o exercício apresentado é composto por um conjunto de 04 questões distintas, com as seguintes características:

Questão 01 – É apresentado um enunciado, descrevendo um problema específico, para o qual foram disponibilizados 3 programas completos desenvolvidos com Scratch, como resposta ao enunciado. Contudo todas as sugestões possuíam erros distintos, em menor ou maior quantidade. A partir disto, os alunos deveriam interpretar os diferentes códigos e escaloná-los nos seguintes níveis: 3-totalmente incorreto, quando o programa apresentava diferentes erros de lógica, sintaxe e estrutura; 2-incorreto, para programas com erros em menor quantidade, e; 1-impreciso, quando o programa não apresentou erros ou foram denotados em quantidades mínimas. Visto que as 3 aplicações apresentavam erros, o aluno após a classificação, deveria implementar um programa correto para atender a demanda da questão. A Figura 52 demonstra o enunciado da questão e, os códigos propostos.

Figura 52-Enunciado e programas propostos para questão 01 – Caso06

1. **Problema 01**: Uma empresa precisa de um sistema para o controle de vendas de produtos. O operador deverá informar o números de itens vendidos. Para cada item deverá informar o código, descrição, quantidade e valor unitário. O sistema deverá calcular e informar o total de venda por produto e ao final informar o total de vendas geral.



Fonte: Próprio autor.

Questão 02 – O estudante deveria, com base em um novo enunciado, implementar uma aplicação também utilizando a linguagem Scratch. A característica importante desta atividade é o fato de que a demanda apresentada é muito próxima à solução da questão 01, com um tema diferente. Esperava-se que o aluno identificasse esta peculiaridade e pela observação do exercício anterior construísse uma solução válida, utilizando a técnica de epitomização (WILEY, 2000). O enunciado é demonstrado na Quadro 33.

Quadro 33-Enunciado da questão 02 – Caso06

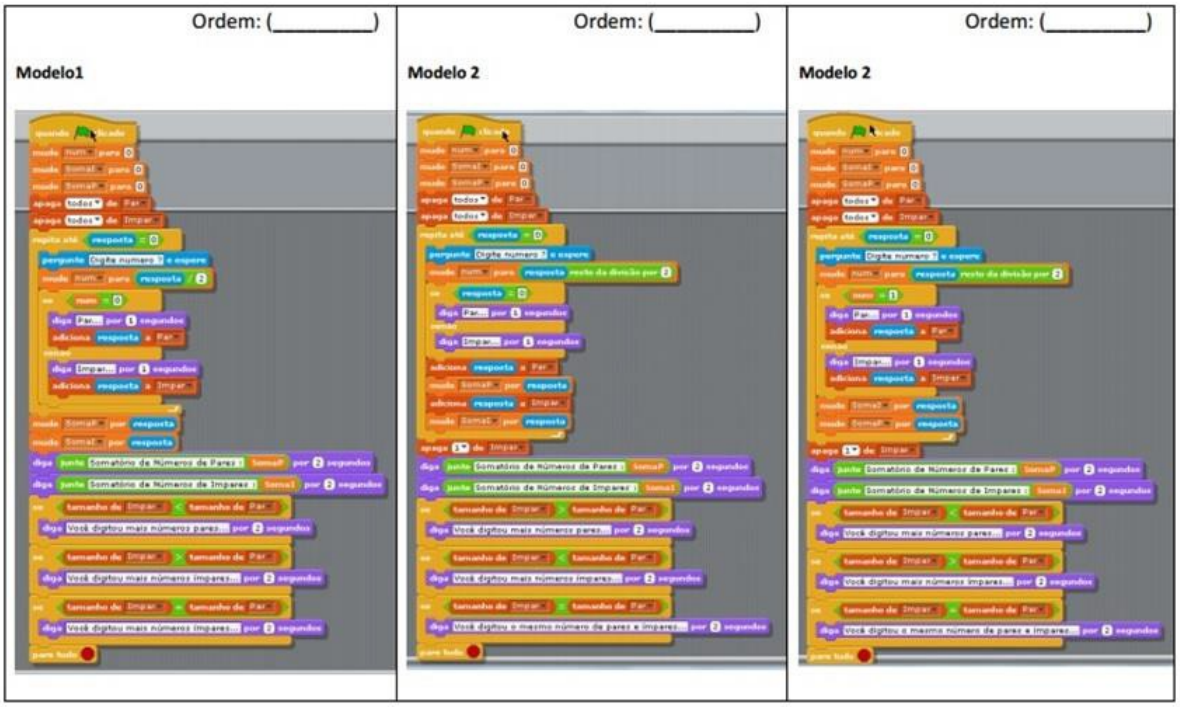
2. Utilizando como base os códigos do Problema 01 ou a resolução que você desenvolveu, implemente uma solução que atenda a seguinte demanda: Um sistema de controle de notas de alunos, onde deve ser informado o número de alunos. A aplicação deve solicitar o nome e 3 notas de cada alunos e informar a média deste. Ao final apresentar a média de todos os alunos.

Fonte: Próprio autor.

Questão 03 – Atividade semelhante à questão 01, porém com uma descrição de problema diferente, conforme Figura 53.

Figura 53-Enunciado e programas propostos para questão 03 – Caso06

3. **Problema 02:** Preciso implementar um sistema que armazene todos os números que o usuário digitar em listas, contudo os números pares devem ser colocados em uma lista chama Par e os números ímpares em uma lista chamada Impar. O sistema deverá apresentar o somatório dos valores armazenados em cada lista, distinguindo a soma dos pares e ímpares e ao final dizer se o usuário digitou mais números pares ou impares. O Sistema deverá ser finalizado quando for digitado 0 (zero).



Fonte: Próprio autor.

Questão 04 – Esta tarefa seguiu o mesmo modelo da questão 02, propondo um problema a ser implementado com Scratch, tendo como base as respostas da questão 03. O enunciado desta atividade é descrito no Quadro 34.

Quadro 34-Enunciado da questão 04 – Caso06

3. **Problema 02:** Preciso implementar um sistema que armazene todos os números que o usuário digitar em listas, contudo os números pares devem ser colocados em uma lista chama Par e os números ímpares em uma lista chamada Impar. O sistema deverá apresentar o somatório dos valores armazenados em cada lista, distinguindo a soma dos pares e ímpares e ao final dizer se o usuário digitou mais números pares ou impares. O Sistema deverá ser finalizado quando for digitado 0 (zero).

Fonte: Próprio autor.

Para a avaliação dos resultados foi importante considerar que a aprendizagem da linguagem de programação, assim como o entendimento de aplicações são os principais desafios para o estudante, segundo Winslow (1996).

A fim de correlacionar as observações sobre os exercícios desta atividade, a correção das questões ocorreram por afinidade, sendo avaliadas em conjunto os casos 01/03 e 02/04. Esta organização fundamentou-se pela proximidade dos atributos necessários para conceber as respostas para os 2 grupos de problemas.

Na correção das questões 01 e 03 avaliou-se a capacidade do aluno em interpretar o código, entender a lógica proposta e com este conhecimento encontrar os erros nos programas disponibilizados, tendo o discernimento de classificá-los de acordo com a severidade das anomalias identificadas. No Quadro 35 é apresentado o gabarito destas questões, demonstrando a ordem das aplicações segundo os erros.

Quadro 35-Gabarito para questões 01 e 03 – Caso06

Questão	Programa	Nível
Níveis possíveis: 1-Impreciso 2-Incorreto 3-Totalmente Incorreto		
Questão 01	Modelo 1	2
	Modelo 2	1
	Modelo 3	3
Questão 02	Modelo 1	2
	Modelo 2	3
	Modelo 3	1

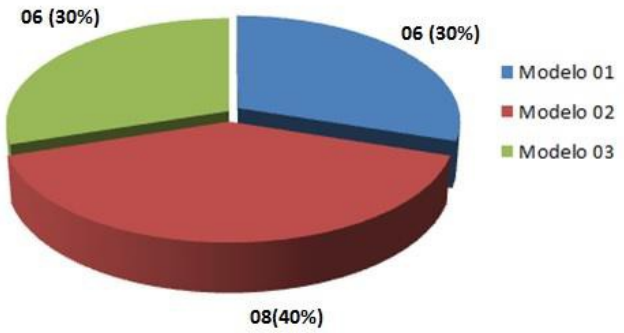
Fonte: Próprio autor.

O entendimento da avaliação da questão 01 passa pela identificação dos modelos disponibilizados aos alunos, os quais possuíam as seguintes características:

- O Modelo 01 possuía a classificação de incorreto (2), pois apresentava o somatório geral dos itens (tg) e a mensagem referente ao total do item fora do laço principal do programa, o que inviabilizava a entrega do montante e informação correta ao usuário. A solução para o código consistia em trocar a posição dos comandos para dentro do laço.
- Modelo 02, definido como impreciso (1), em que a aplicação atendia todas as demandas do enunciado, com um erro simples nas equações de cálculo do valor total do item (ti) e total geral (tg). Para corrigir os erros, o aluno deveria apenas trocar as variáveis repetidas nos cálculos.
- O Modelo 03 estava totalmente incorreto (3), visto que as equações para cálculo do total do item (ti) e total geral (tg), assim como a mensagem referente ao total do item estavam posicionadas fora do laço principal. A correção estava baseada apenas na troca de posição destes comandos. O maior grau de erro imputado a este modelo, em referência ao modelo 01, é justificado pela presença de um número maior de instruções posicionadas incorretamente.

O Gráfico 41 apresenta uma análise geral das respostas, relacionadas à questão 01, onde é percebido que 30%, 06 alunos, acertaram a classificação para o modelo 1 e também para o 3, enquanto 08 estudantes (40%) rotularam corretamente o segundo modelo. Tem-se a maioria dos acertos computados ao segundo programa, o que é plausível, uma vez que o programa apresenta uma solução muito simples, contudo não se descarta os demais índices, muito próximos a este.

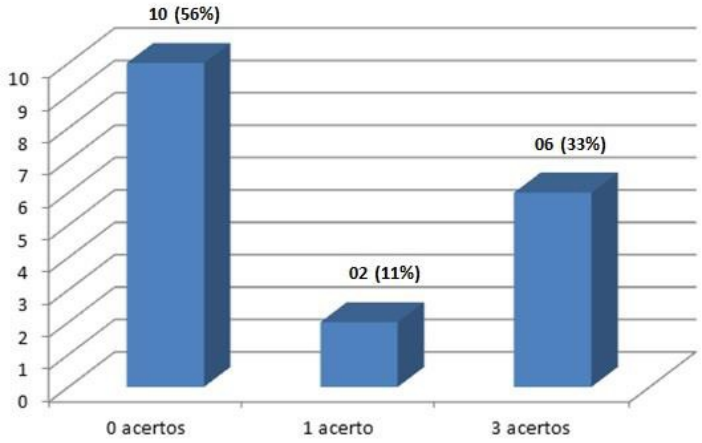
Gráfico 41-Acertos da questão 01 – Caso06



Fonte: Próprio autor.

Uma informação relevante, neste momento, se refere à quantidade de acertos por aluno, no qual 6 alunos (33%) obtiveram êxito na identificação dos 3 programas, 2 estudantes (11%) classificaram corretamente 1 aplicação e, a grande maioria, 56% (10 discentes) erraram todas as especificações, ou seja, não conseguiram interpretar e nem relacionar as anomalias nos códigos (Gráfico 42). Esta observação corrobora com a proposição de Winslow (1996), em que um grande grupo de indivíduos demonstrou dificuldade na compreensão das regras de sintaxe e lógica, o que tornou a identificação das anomalias uma tarefa complexa, mesmo considerando a utilização de uma linguagem baseada em blocos visuais como o Scratch.

Gráfico 42-Acertos por alunos na questão 01 – Caso06



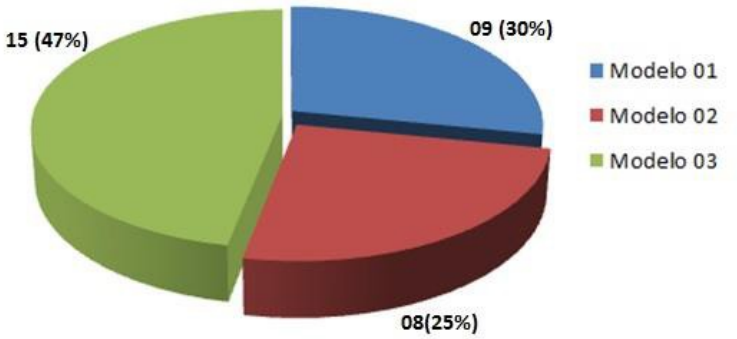
Fonte: Próprio autor.

A questão 03 estava baseada na avaliação de uma aplicação utilizando o recurso de listas (vetores), em que o sistema deveria armazenar todos os valores digitados pelo usuário em 2 listas diferentes, uma com números pares e outra com valores ímpares, informando ao final qual o vetor com maior tamanho. Os programas disponibilizados para esta tarefa possuíam as seguintes características:

- Modelo 01, com classificação 2 (Incorreto), construído com 2 erros: o primeiro na equação da estrutura de condição para identificação se o valor digitado era par ou impar, ao invés de analisar o resto da divisão do valor por 2 a estrutura validava a divisão do números, erro no somatório de valores da lista e, falha na lógica para comparação do tamanho da lista de par e ímpar.
- O Modelo 02, com o maior número de erros, classificado como totalmente incorreto. As seguintes incorreções foram adicionadas ao código: todos os valores digitados em ambas as listas, independentes de pares e ímpares, eram somados além dos erros de lógica em ambas estruturas de condição, utilizadas para definir os tamanhos dos vetores.
- Com apenas um erro (Impreciso), o Modelo 3 possuía uma estrutura de condição que validava erroneamente o resto da divisão por 2, não identificando os valores pares.

Na questão 03 os alunos demonstraram um maior número de acertos em relação ao Modelo com menos erros (47%), a classificação dos Modelos 01 e 02 ficou muito próxima, com 25% e 30% de acertos, respectivamente (Gráfico 43).

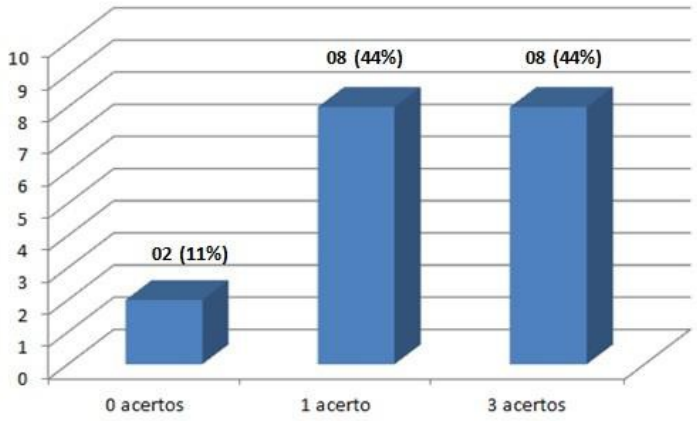
Gráfico 43-Acertos por alunos na questão 03 – Caso06



Fonte: Próprio autor.

Embora a atividade 03 apresentasse um grau maior de dificuldade, inserida pela utilização de vetores, os alunos obtiveram uma evolução no desempenho em relação à questão 01, com 2 estudantes (11%) não identificando corretamente nenhum dos programas, 44% (8 alunos) apontando os erros em ao menos 1 dos programas e, o mesmo quantitativo, relacionado aos 3 programas, conforme Gráfico 44. Este avanço, na concepção dos discentes, pode ser atribuído ao fato desta ser a terceira tarefa realizada durante o período de aula, em que os alunos já haviam executado outros exercícios, ou seja, demonstraram um maior raciocínio crítico e habilidade cognitiva para a resolução do problema, aptidão necessária para iniciantes em programação segundo Bem-Ari (2001).

Gráfico 44-Acertos por alunos na questão 03 – Caso06



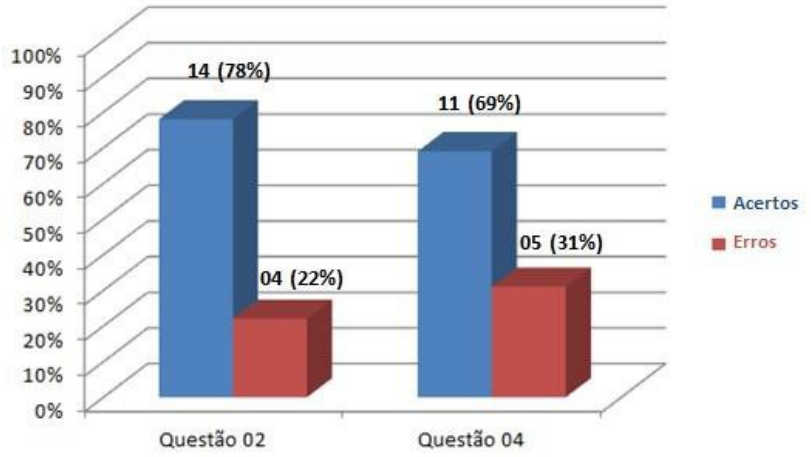
Fonte: Próprio autor.

A análise dos exercícios 02 e 03 tiveram o intuito de considerar a capacidade dos alunos em desenvolver novas aplicações, utilizando como recursos códigos prontos, apenas

com objetivos diferentes, princípio do conceito de epitomização, o qual pode ser descrito também como raciocínio por analogia (DE BARROS *et al*. 2005). Esta é uma técnica comum, recorrida por programadores que, constantemente, adaptam soluções programadas anteriormente para resolver problemas atuais.

Percebeu-se na questão 02 que 78% dos alunos conseguiram implementar o programa de forma correta, sem erros e, 69% acertaram a questão 04. As causas na diferença de percentual entre as atividades, também é denotada nos estudos de Pereira & Rapkiewicz (2004), visto que a segunda atividade exigiu a adoção de um padrão de programação mais rebuscado, com a necessidade da aplicação de estruturas de condição, vetores e equações, o que aumentou a exigência de abstração por parte dos iniciantes. O Gráfico 45 demonstra os índices de acertos e erros destas questões.

Gráfico 45-Acertos por alunos na questão 03 – Caso06



Fonte: Próprio autor.

Contudo, o intuito de ambos os testes foi avaliar a habilidade dos alunos em entender a lógica e sintaxe utilizadas na resolução das atividades 01 e 03 para compor as aplicações referentes aos enunciados 02 e 04. Esta motivação levou a análise minuciosa das respostas fornecidas. Os códigos corretos foram descartados, por se entender que nestes casos a capacidade de epitomização dos estudantes foi válida e efetiva. Nos casos em que as aplicações não funcionaram avaliou-se 2 aspectos relevantes: inconsistências causadas por erros de lógica e erros de sintaxe.

Na segunda questão, 04 programas foram implementados de forma incorreta (Gráfico 49), dos quais 2 com imprecisões de sintaxe (equações erradas) e, 2 de lógica (comandos em posições não válidas). Já na questão 04 foram 2 aplicações com erros de sintaxe (comandos

organizados de forma incorreta) e, 3 com a lógica errada (comandos e variáveis fora de posição e equação de cálculo da média com valores inadequados). Em ambos exercícios os acadêmicos não expressaram as habilidades desejáveis para entender os fragmentos das ações programadas, tampouco as convenções lógicas e composição de planos dos programas prontos, com a meta de proporem soluções equivalentes. As deficiências percebidas neste contexto são também justificadas pela pesquisa de Soloway & Ehrlich (1984), apontadas como exigências básicas para a formação de programadores.

Considerou-se esta uma ação válida, pois os 4 exercícios seguiram as recomendações de Setubal (2000) disponibilizando uma base coerente, abordando objetivos fundamentais de programação, com conceitos claramente definidos, ênfase no pensamento crítico e abordando temas reais, passíveis de implementação. Considerando estas características e pelo desempenho positivo da maioria do grupo na resolução dos problemas, sugere-se que a estrutura de ensino vem alcançando o propósito de estimular o raciocínio lógico dos estudantes.

5.1.7. ANÁLISE DE RESULTADOS – CASO07

Seguindo o plano de atividades da disciplina de algoritmos, proposto nesta tese, os alunos passaram a ter contato com a linguagem Scratch4OS, um *software* baseado na linguagem Scratch, que também trabalha com o conceito de programação pela movimentação de blocos visuais (AVILA *et al.* 2013). O diferencial do Scratch4OS está no fato de permitir a tradução das construções visuais em *scripts* para mundos virtuais, funcionalidade que possibilita a programação de diferentes ações para objetos em um metaverso.

Com a compreensão das funções desta nova linguagem, foi apresentado aos alunos o mundo virtual OpenSim (OS), através de atividades simples de reconhecimento do ambiente, configuração de avatares e interação com outros indivíduos no próprio OS. A partir do momento que os acadêmicos se sentiram confortáveis em utilizar o metaverso, foi demonstrado o roteiro básico para a construção e inserção de ações em objetos neste ambiente.

A introdução à sintaxe da linguagem LSL ocorreu com o transporte de códigos desenvolvidos com Scratch4OS para os objetos (também denominados de *prisms*) do OpenSim e, posterior comparação dos resultados destes com as estruturas visuais do Scratch. O intuito destas intervenções foi promover uma estrutura de ligação entre imagem e texto, a

fim de que o aluno compreendesse a sintaxe dos *scripts* em LSL, relacionando o seu funcionamento com a base de conhecimento construída com os estudos sobre programação no Scratch. A predisposição do aluno em assimilar a nova linguagem, seus conteúdos e conceitos apontou para uma aprendizagem significativa (Ausubel, 2003), em que um novo conjunto de informações se relaciona com as estruturas de conhecimento já construídas anteriormente.

Neste momento os discentes passaram pelo processo de mudança de paradigma, migrando da programação baseada em blocos visuais para a programação imperativa, focada no paradigma estruturado, que segundo Delgado *et al.* (2004) é considerado como um padrão legítimo de desenvolvimento, por tratar vários problemas distintos e desfrutar da vantagem de abordar conceitos importantes de maneira simples.

Por meio de diferentes exercícios relacionados à construção de *scripts*, atribuindo ações a objetos no metaverso, os estudantes reconheceram a sintaxe dos comandos básicos da linguagem LSL, seguindo os preceitos contidos na afirmação de Dijkstra (1989) o qual descreveu que o processo de aprendizagem de algoritmos deve ser lento e gradual, sustentado por diferentes práticas, de maneira que o indivíduo se apodere de conceitos que tragam pequenos retornos. Desta forma foram reconhecidos os conceitos fundamentais da linguagem LSL, tais como: declaração de variáveis, operações básicas, instruções primitivas, estruturas condicionais e de repetição, além da concepção de estados e eventos.

Embora a linguagem LSL possua um conjunto de características próprias (estados, eventos e comandos) sua construção é baseada na linguagem C, o que é considerado um ponto positivo na estrutura desta pesquisa, visto que o acadêmico deverá chegar ao final da disciplina de algoritmos programando em C.

O caso07 (Quadro 28) contemplou uma atividade de interpretação de códigos. Cada aluno recebeu um conjunto de 10 *scripts* diferentes e, deveria analisá-lo, executar os códigos no OpenSim e descrever a funcionalidade de cada aplicação. O Quadro 36 apresenta o conteúdo principal de cada *script*.

Quadro 36-Relação de conteúdos dos *scripts* LSL – Caso07

Questão	Conteúdo	Questão	Conteúdo
01	Evento Entry Comandos de Entrada/Saída	06	Comandos de Entrada/Saída Evento Touch
02	Evento Entry Estrutura de Repetição (while)	07	Envio de Mensagem (llOwnerSay) Evento Touch
03	Evento Entry Comandos de Entrada/Saída Evento Touch	08	Evento Touch Estrutura de Repetição (do...while)
04	Evento Entry Estrutura de Repetição (for) Estrutura de Condição (if)	09	Evento Entry Envio de Mensagem (llOwnerSay) Comandos do OS (Timer)
05	Evento Entry Envio de Mensagem (llOwnerSay) Estrutura de repetição (for)	10	Evento Entry Comandos de Entrada/Saída Estruturas de menu Estrutura de Condição (if)

Fonte: Próprio autor.

Já no Quadro 37 são demonstrados os 3 primeiros *scripts* da atividade. O intuito desta tarefa foi estimular os estudantes a entenderem o significado de cada comando a partir da percepção dos resultados gerados com a execução de cada *script*.

Quadro 37-*Scripts* disponibilizados aos alunos – Caso07

```
Algoritmo 01:

string teste = "Iniciando a programar com LSL";
default
{
    state_entry()
    {
        llSay(0, "Código ativado!!!");
    }
    touch_start(integer total_number)
    {
        llSay(0, teste);
    }
```

```
Algoritmo 02:

integer contador = 0;
integer montante = 0;
default
{
    state_entry()
    {
        llSay(0, " Código ativado!!!");
    }
    touch_start(integer total_number)
    {
        while(contador <= 10) {
            montante = montante + 10;
            llSay(0, "Soma é: "+ montante);
            contador++;
        }
    }
}
```

```
Algoritmo 03:

integer contador = 6;

default
{
    state_entry()
    {
        llSay(0, "Código Ativo!!!");
    }
    touch_start(integer total_number)
    {
        llSay(0, "Tocado.");
        do {
            contador++;
            llSay(0, "Contagem 1");
        } while(contador < 5);
    }
}
```

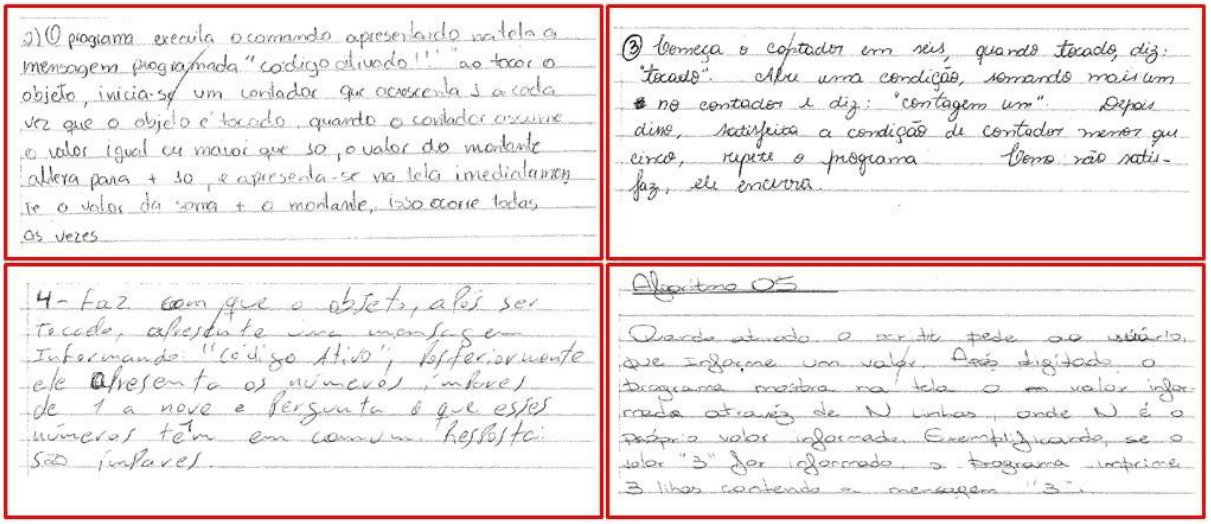
Fonte: Próprio autor.

Nesta atividade participaram 22 alunos, os quais estavam conectados ao mundo virtual disponibilizado em um laboratório. A descrição da finalidade de cada código poderia ser realizada diretamente, sem utilizar o MV ou através da transcrição dos *scripts* para algum objeto do metaverso e observação dos resultados. Todos discentes optaram por esta última linha de ação, fato justificado, visto que nenhum deles demonstrava, neste instante, habilidade suficiente para reconhecer o funcionamento da aplicação sem uma breve execução das mesmas.

Um total de 20 acadêmicos concluiu com 100% de êxito a atividade, ou seja, apenas 2 discentes não explicaram corretamente os 10 *scripts* disponibilizados. Ao analisar este percentual, percebe-se que 90% da turma se demonstraram aptos a reconhecer a finalidade de cada código, independente da sua função. Este foi considerado um excelente índice, pois ao considerar a mudança de paradigma e as dificuldades inerentes à construção de soluções em linha de código, a grande maioria obteve sucesso em descrever a finalidade dos programas, se destacando em relação aos outros Casos observados.

No Quadro 38 tem-se algumas das respostas elaboradas para os *scripts* 2, 3, 4 e 5. A leitura das descrições permite perceber um aumento no grau de maturidade dos alunos ao examinar e explicar os programas. A utilização de termos como contador, condição, estrutura, linhas de código denotam uma relação coerente com o que está sendo proposto pelo algoritmo analisado. Aspectos relacionados ao raciocínio lógico são observados a partir da associação dos comandos em sua ordem de execução e a dependência das ações entre estes.

Quadro 38-Amostra de respostas dos alunos – Caso07



Fonte: Próprio autor.

Cabe ressaltar que esta tarefa (Caso07) foi implementada seguindo os estudos de Lister *et al.* (2004) e Whalley *et al.* (2006), os quais buscaram em suas pesquisas avaliar a capacidade de acadêmicos iniciantes em programação a ler e entender códigos programados. Contudo, nesta atividade não se buscou a comparação entre turmas, mas sim constatar o progresso das habilidades do indivíduo em reconhecer a sintaxe dos comandos e sistematizar o raciocínio lógico, a fim de se tornarem capazes de compor uma dedução válida para problemas de programação.

5.1.8. ANÁLISE DE RESULTADOS – CASO08

À medida que o contato dos alunos com o metaverso OpenSim era aprimorado, a complexidade das atividades de programação neste ambiente também foram ampliadas, com o intuito de estimular os estudantes a praticarem a escrita de código utilizando a linguagem LSL. Vários aspectos positivos foram identificados durante este período, visto as características próprias do ambiente imersivo como a capacidade de permitir aos estudantes resolver situações problema a partir do contato e observação do mesmo (DEDE, 2013), além de ser uma solução para construção de espaços educacionais colaborativos (ÁVILA, 2013).

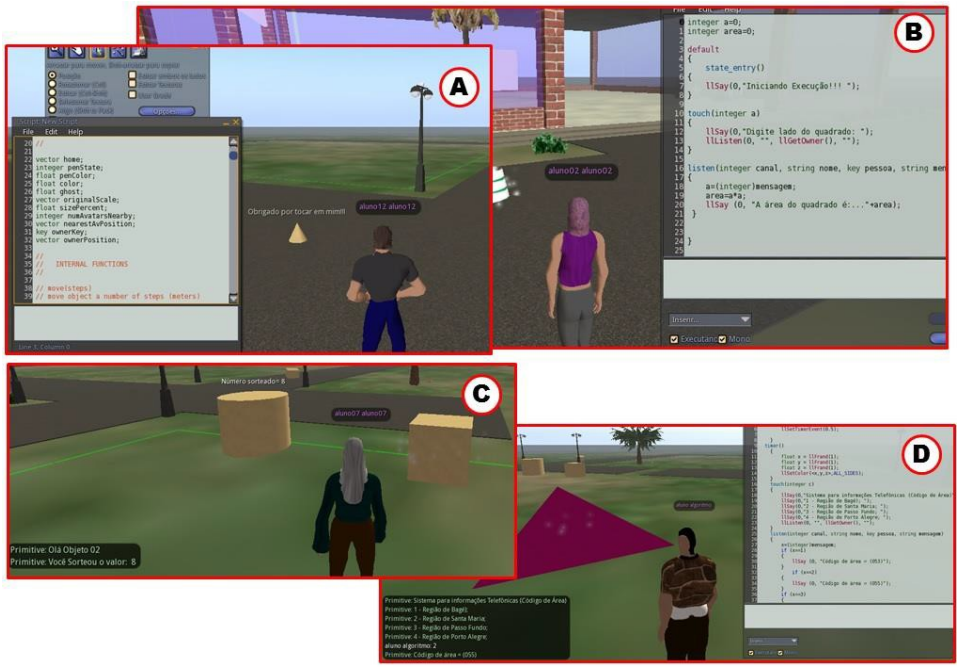
A implementação de *scripts* para os objetos no mundo virtual possibilitam ao aluno observar em tempo de execução e com *feedback* automático (MOREIRA & FAVERO, 2009), os resultados decorrentes de suas aplicações, incitando aspectos de imersão, o que segundo Frozza *et al.* (2009) aproximam o usuário e o sistema. Neste contexto destaca-se também a importância da integração dos elementos que compõem o metaverso como gráficos 3D, inteligência artificial e orientação a objetos, os quais motivam a construção de novos artefatos conforme descrito por Gouveia (2000). Para Pereira (2010) o compartilhamento do ambiente permite a interação colaborativa, criatividade e o processo de aprendizagem, pois o contato com o MV permite explorá-lo e atualizá-lo simultaneamente (LÉVY, 1999), o que tem um forte caráter motivacional, alinhado à proposta do Caso08.

Nesta atividade os alunos foram induzidos a trabalhar em duplas para a implementação de soluções no OpenSim. O tema de desenvolvimento era livre e os alunos poderiam construir códigos com a utilização do próprio LSL ou Scratch4OS, contudo as duplas deveriam descrever os *scripts* concebidos e sua finalidade. Explorou-se neste Caso, além das características de programação para o metaverso, a aprendizagem colaborativa em que os integrantes realizaram uma ação conjunta, porém cada indivíduo sendo responsável pela qualidade na produção, de acordo com seus interesses e possibilidades, conforme descrito por Arnáiz *et al.* (1999) e Damiani (2008), ou seja, se beneficiando do fato de que o ambiente de sala de aula, naturalmente se apresenta como um local de colaboratividade (JERMANN *et al.* 2001).

Estavam presentes em aula 18 alunos que compuseram 09 duplas, as quais programaram diferentes aplicações. A Figura 54 apresenta uma amostra das respostas apresentadas:

- Resposta A – um script que apresentava uma mensagem cada vez que o objeto era tocado (“Obrigado por tocar em mim!!!”), contabilizando o número de toques e, fazendo o objeto se afastar do avatar a cada aproximação do mesmo.
- Resposta B – neste caso os alunos propuseram um algoritmo, com LSL, para o cálculo da área de um quadrado. A solicitação de digitação do lado do quadrado era feita no evento (*state-entry*). O usuário fornece o valor, por meio do campo de comunicação (*chat*), a partir disto o cálculo é executado, sendo o resultado apresentado na tela. Cabe ressaltar que a temática desta resposta já havia sido explorada nos estudos iniciais de lógica, construção de fluxogramas e também implementado com o Scratch.
- Resposta C – esta foi a melhor resposta apresentada para o Caso08, pois os estudantes exploraram conceitos mais avançados para a sua concepção. O script ao ser disparado, por meio do toque ao objeto, gerava um valor aleatório que era transmitido, por meio de um canal de comunicação, para outro objeto que constantemente apresentava na tela o valor sorteado. Percebeu-se nesta situação que os alunos saíram da região de conforto, em que contavam com os materiais didáticos das aulas e buscaram novos conceitos e comandos para o desenvolvimento dos programas. É possível constatar o alto nível de abstração demonstrado por estes estudantes, visto que a organização dos códigos para o funcionamento correto dos *scripts* foi uma tarefa complexa.
- Resposta D – mais um caso em que os alunos buscaram algoritmos já trabalhados em sala de aula e transportaram a lógica para a construção do *script* com LSL. Esta foi uma aplicação que disponibilizava um menu de opções e o avatar deveria escolher uma das opções e receber um *feedback* instantâneo.

Figura 54-*Scripts* desenvolvidos no OpenSim – Caso08



Fonte: Próprio autor.

O Quadro 39 apresenta a relação dos demais *scripts* construídos pelos estudantes neste exercício. Ressalta-se que todos os participantes demonstraram algum nível de dificuldade ao construir os programas. Problemas comuns foram constatados na construção dos *scripts*, relacionados diretamente à sintaxe incorreta na concepção dos comandos, como erros pela não inserção de ponto e virgula (;) ao final da instruções, a falta e parênteses ou chaves para encerrar os blocos de comandos, imprecisão na declaração de variáveis e algumas inconsistências na digitação dos comandos (erros ortográficos). Este tipo de situação não havia sido notada em outros exercícios na disciplina, apontando a dificuldade geral retratada por Proulx (2000) ao descrever a dificuldade e frustração de alunos iniciantes ao programar com linguagens formais, em que é necessário uma alta demanda de preocupação com detalhes de sintaxe.

Apesar de todas as adversidades demonstradas pelas duplas, com relação à sintaxe da linguagem LSL, apenas o item I não foi entregue funcionando corretamente. A dificuldade exibida pelos discentes na construção do programa não estava relacionada pontualmente a questões de lógica, mas sim pela inabilidade de programação com o LSL. De qualquer forma a organização do *script* apresentado seguiu um padrão inteligível, em que o erro estava relacionado a uma falha no evento llListen, uma função que recebe informações repassadas pelo avatar. Identificado o defeito no código, os próprios estudantes solucionaram e finalizaram a aplicação.

Quadro 39-Listagem de *scripts* programados pelos alunos – Caso08

Item	Descrição Script
E	Aplicação com objetivo de fazer um determinado objeto mudar de cores aleatoriamente, a parti de um toque. Um novo toque cessava a mudança de cores.
F	O Script pergunta ao avatar sua idade e retorna uma mensagem dizendo se ele pode solicitar sua habilitação para dirigir.
G	A ser tocado, um objeto muda de tamanho e cores e faz uma contagem baseado em uma estrutura de repetição.
H	A aplicação interage com o avatar, perguntando o seu nome, após apresenta na tela o ID do avatar.
I	O avatar ao interagir com um *prism* é questionado sobre peso e altura. Como saída a aplicação apresenta o índice de massa corporal do avatar.

Fonte: Próprio autor.

Os resultados corroboram para a afirmação que os alunos, com a evolução da disciplina baseada no Modelo 02, foram demonstrando uma maior capacidade de entendimento na construção de soluções, a partir de um raciocínio lógico apurado. As propriedades positivas do mundo virtual (comunicação, interação, imersão e *design*) serviram nesta etapa da pesquisa como um degrau, um incentivo para que os estudantes pudessem migrar suas concepções de programação com blocos visuais (Scratch e Scratch4OS) para a programação imperativa com o LSL e, prepararem-se para a linguagem C.

Outro fator pertinente destaca-se pela exposição dos alunos, ao final do Caso08, os quais afirmaram que 100% das soluções apresentadas estavam baseadas em recursos, programas e atividades anteriormente realizadas na disciplina. Esta postura deixa claro uma maior habilidade dos acadêmicos em abstrair conceitos e propor soluções já conhecidas, sobre novos paradigmas, superando as perspectivas apresentadas por Mendes (2002), o qual apontou a prática na resolução de problemas e reconhecimento das dificuldades como empecilho para a aprendizagem da lógica e programação.

5.1.9. ANÁLISE DE RESULTADOS – CASO09

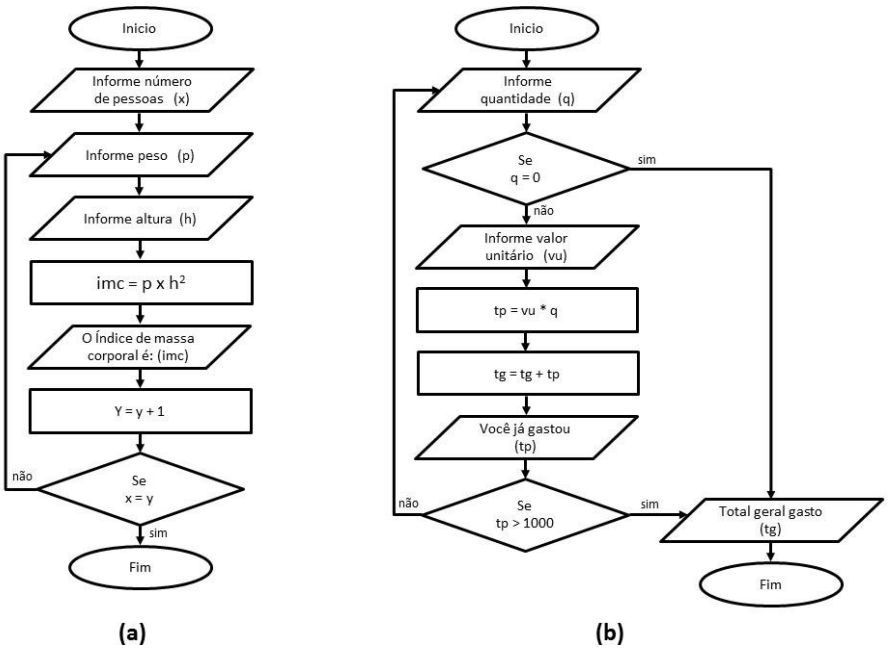
Neste momento da disciplina todo conteúdo previsto, com a utilização dos paradigmas de programação com blocos visuais e no mundo virtual já foram vencidos. Os estudantes passaram então a reconhecer todos os conceitos de desenvolvimento com a linguagem C,

desde a declaração de variáveis até a construção de estruturas de condição, estruturas de repetição, manipulação de *strings*, vetores, matrizes, funções e procedimentos.

Um fato interessante de ser descrito está relacionado a uma particularidade observada, que em nenhum momento os estudantes demonstraram qualquer tipo de sentimento de repúdio ou traumático ao utilizarem a linguagem C, situação distante da afirmação de Pereira e Rapkiewicz (2004), os quais descreveram as dificuldades apresentada pela maioria dos alunos ao assimilarem abstrações em linguagens imperativas. Esta peculiaridade pode ser atribuída à estratégia adotada neste estudo, em que foi possibilitado ao estudante, a partir do contato com outros paradigmas (Scratch e LSL), construir memórias e padrões para a utilização do novo paradigma, situação descrita por Menezes *et al.* (2008).

O Caso09 consistiu em disponibilizar aos alunos 02 fluxogramas, a partir dos quais eles deveriam construir uma solução computacional, desenvolvida em linguagem C, que atendesse a demanda de ambas as representações (Figura 55). Esta tarefa se assemelhou ao Caso03, em que os alunos tiveram que interpretar e descrever a funcionalidade de fluxogramas, contudo neste momento o discente deveria demonstrar além da capacidade de reconhecer a problemática, propor um programa funcional, respeitando a formalidade da linguagem de programação.

Figura 55-Construindo aplicações com a linguagem C a partir de fluxogramas – Caso09



Fonte: Próprio autor.

Os algoritmos a serem desenvolvidos seguiam a seguinte estrutura: na questão 01 (Figura 50a) o sistema solicita a informação da quantidade de indivíduos (x) para cálculo e apresentação do índice de massa corporal ($imc=p/h^2$) de cada um, para isso coletando os dados de peso (p) e altura (h) dos indivíduos a cada ciclo da estrutura de repetição; o segundo problema, questão 02 (Figura 50b), a aplicação recebe informação de quantidade (q) e valor unitário (vu), demonstrando o valor parcial da compra (tp) e acumulando o total geral (tg=tg+tp), sendo que o sistema é encerrado quando o usuário informa 0 (zero) para quantidade ou quando o total parcial ultrapassa o valor de 1000 e, após qualquer uma destas 02 ocorrências é apresentado na tela o total geral (tg).

A avaliação desta atividade considerou alguns aspectos específicos, importantes para entender a forma de raciocínio do aluno e sua capacidade em construir uma aplicação, respeitando a formalidade da linguagem C. Para isto foi observada a habilidade do acadêmico, em: programar um código correto e sem erros, propor uma estrutura lógica efetiva, apresentar um raciocínio crítico, não cometer incorreções de grafia e compor uma sintaxe válida. Os fatores de fracasso foram pontuados e classificados como erros de lógica ou erros de sintaxe. Estes últimos agrupados de acordo com o número de incidências encontradas nos programas, o que originou as seguintes categorias:

- PV: caracterizado pela falta de ponto e vírgula em algum dos comandos;
- Chaves: detectáveis pela falta de chaves ao encerrar algum bloco de instruções, também apontado em situações que o estudante posicionou as chaves em posições inadequadas do código;
- Ortografia: situação observada pela escrita inadequada dos comandos da linguagem C, ou seja, incidência de erros de grafia;
- Est Cond: neste grupo são enquadrados erros na concepção de estruturas de condição no programa. Condições mal formuladas são os problemas observados na grande maioria dos casos, neste contexto.
- Est Rep: categoria de erros encontrados na construção de estruturas de repetição, apontados comumente por erros na composição do comando;
- Equação: item que aponta erros na composição de equações no programa. A falta da linearidade e símbolos incorretos são as principais encontradas.

Um mapa demonstrativo da avaliação das questões 01 e 02 (Caso09) é apresentado no Quadro 40, onde estão marcadas as observações atribuídas a acertos, erros de lógica e de sintaxe para cada participante.

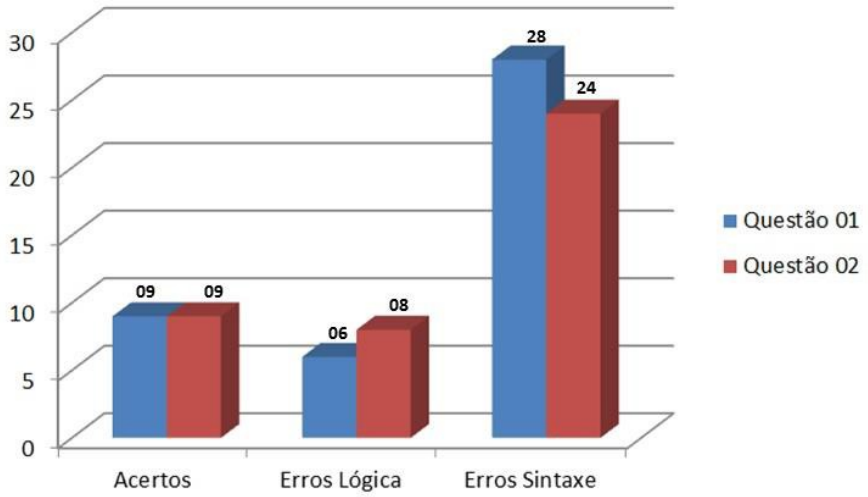
Quadro 40-Mapeamento dos acertos e erros das atividades – Caso09

Aluno	Questão 01								Questão 02							
			Erros de Sintaxe								Erros de Sintaxe					
	Acerto	Erro de Lógica	PV	Chaves	Ortografia	Est Cond	Est Rep	Equação	Acerto	Erro de Lógica	PV	Chaves	Ortografia	Est Cond	Est Rep	Equação
1			X	X						X	X	X			X	
2							X			X	X					
3		X		X	X	X		X			X	X	X		X	
4	X								X							
5			X	X						X	X					
6	X							X	X							
7							X			X			X	X		
8	X			X			X	X					X	X	X	
9	X							X		X						
10		X	X				X	X		X	X					X
11		X	X		X					X	X			X		
12		X	X					X	X		X					
13	X								X							
14	X		X						X		X					
15		X	X				X		X		X					
16	X								X							
17	X								X							
18		X					X	X		X	X				X	
19	X						X		X				X			

Fonte: Próprio autor.

Ao quantificar as observações realizadas é possível perceber que 09 alunos obtiveram êxito ao programar soluções para as duas questões, o que caracterizou um percentual de 47% de acertos para cada exercício, conforme demonstrado no Gráfico 46.

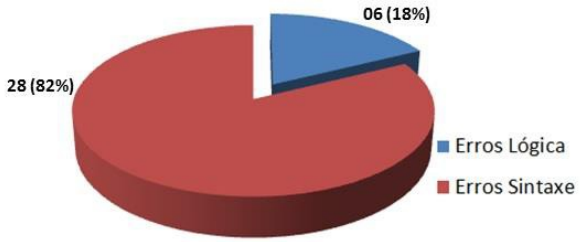
Gráfico 46-Avaliação de acertos/erros das questões 01 e 02 – Caso09



Fonte: Próprio autor.

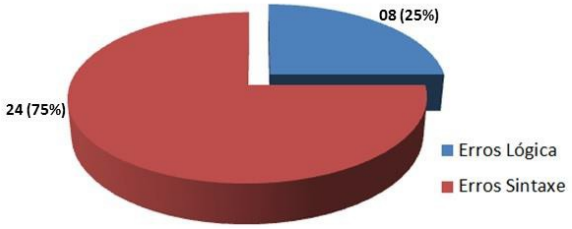
Os gráficos 47 e 48 apresentam os percentuais de erros de lógica e sintaxe das questões 01 e 02, respectivamente. É verificada nos dois exercícios uma grande diferença entre estes tipos de problemas. Na primeira questão constatou-se 18% de incoerências relacionadas à lógica utilizada para a programação e um percentual de 82% de imprecisões na sintaxe. Na questão 02 os valores não foram muito diferentes, com 25% de erros de lógica e 75% de sintaxe. A partir destes dados entende-se que os alunos apresentaram um maior grau de dificuldade na concepção da sintaxe para construção dos programas, do que na lógica proposta para a solução dos mesmos, obstáculos já reconhecidos por Sajaniemi & Kuittinen (2003) em sua pesquisa sobre o processo de aprendizagem de algoritmos.

Gráfico 47-Percentuais de erros da questão 01 – Caso09



Fonte: Próprio autor.

Gráfico 48-Percentuais de erros da questão 02 – Caso09



Fonte: Próprio autor.

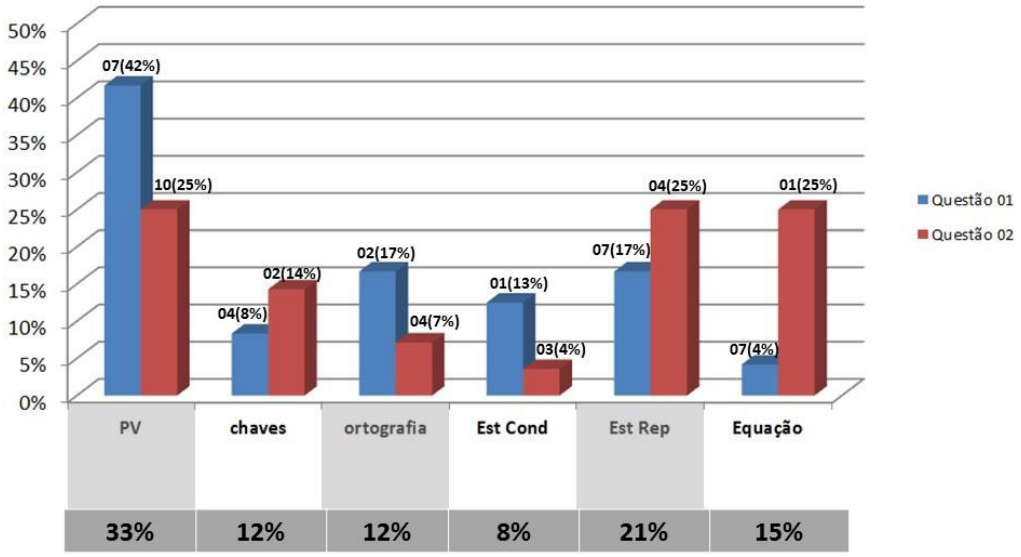
Com o intuito de evidenciar os principais obstáculos encontrados pelos estudantes na programação com a linguagem C, realizou-se um exame minucioso nos erros de sintaxe das soluções apresentadas para o Caso09 (Gráfico 49). Percebeu-se que a falta de ponto e vírgula

ao final dos comandos foi o erro com maior índice de ocorrência, questões, com 33% em ambas as questões, seguido da construção de estruturas de repetição com 21%, equações descritas de forma não linear ou com erro de variáveis (15%), incoerência à distribuição das chaves em blocos de comandos e erros de grafia computaram 12% e, em última posição estão às construções erradas das estruturas de condição (8%).

Destacaram-se, na observação individual de cada elemento do Gráfico 53, que o PV foi o principal erro na questão 01 e, a concepção de estruturas de repetição e equações incorretas caracterizaram-se como os principais problemas no segundo exercício.

Estas informações tornaram-se pertinentes, visto que durante a primeira questão os acadêmicos estavam pressionados a construir uma solução viável, com tempo definido, para o fluxograma, o que provavelmente induziu ao descuido e esquecimento do ponto e vírgula ao final dos comandos. Já o fluxograma da questão 02 estava calcado sobre duas estruturas de repetição, as quais definiam o funcionamento da aplicação, fato que aumentou a complexidade na interpretação e construção do algoritmo, agravado pela necessidade de equações para a execução de cálculos parciais e gerais, o que incrementou a dificuldade dos alunos em resolver o problema.

Gráfico 49-Classificação de erros das questões 01 e 02 – Caso09



Fonte: Próprio autor.

A partir da identificação da deficiência dos estudantes em implementar códigos, descrevendo a sintaxe correta da linguagem, adotou-se uma abordagem didática, defendida por Menezes *et al.* (2008) e Sirotheau (2011), a qual foi baseada na utilização de exemplos com estruturas e lógica aproximada aos exercícios de programação, possibilitando ao

estudante construir uma base de conhecimento para a proposição de soluções de novos problemas (que pode ser observada no Caso10), além da realização de consecutivas práticas de laboratório, como defendido por Prior (2003).

5.1.10. ANÁLISE DE RESULTADOS – CASO10

Por fim, o Caso10 contemplou duas atividades relacionadas à programação em linguagem C, visto que a tarefa de programar exige um esforço cognitivo alcançado por meio da prática e não apenas pela contemplação, segundo Reichert *et al.* (2001) e Al-Ymamy *et al.* (2006).

O primeiro exercício, aqui denominado atividade 01, buscou explorar a capacidade dos estudantes em interpretar um determinado código e esclarecer a condição do mesmo, definindo se o programa estaria correto, com erro de lógica ou erro de sintaxe. Esta metodologia visa propiciar ao discente a oportunidade de desenvolver competências para analisar algoritmos e reconhecer técnicas básicas para construção de aplicações, considerando a heterogeneidade na solução dos diferentes problemas, tarefa alinhada com a proposição de Baeza-Yates (1995).

Para isso foram disponibilizadas duas questões, com demandas distintas e três soluções implementadas para cada uma, sendo: uma proposta de código correto; um programa com vários erros de grafia e; um algoritmo desenvolvido sobre uma lógica que não atenderia a solução desejada. Visto a dificuldade demonstrada pelos discentes no Caso09 em construir programas respeitando a sintaxe da linguagem imperativa, buscou-se com esta atividade promover uma prática de observação e identificação dos erros mais comuns, cometidos por programadores iniciantes, a fim de formar uma consciência individual, estimulando assim a redução destes problemas.

Nos Quadros 41 e 42 são apresentados, respectivamente, os enunciados das questões da atividade 01, assim como a proposta de código correto para cada enunciado.

Quadro 41-Enunciados das questões 01 e 02 – Atividade 01/Caso10

Questão 01	Questão 02
Abaixo seguem 3 códigos desenvolvidos em linguagem C para a resolução de um determinado problema. Atenda as seguintes demandas: * Descreva a finalidade do código; * Um código esta correto, um código apresenta erros de lógica e um código apresenta erros de sintaxe. Identifique cada um deles. Nos códigos com erro, informe uma possível maneira de corrigir tais problemas.	Seguem 3 códigos, desenvolvidos em linguagem C, para o calculo do IMC (peso dividido pela altura ao quadrado). Atenda as seguintes demandas: * Um código esta correto, um código apresenta erros de lógica e um código apresenta erros de sintaxe. Identifique cada um deles. Nos códigos com erro, informe uma possível maneira de corrigir tais problemas. Lembro que o código deve respeitar os índices: Entre 17 e 18,49 - Abaixo do peso; Entre 18,5 e 24,99- Peso normal; Entre 25 e 29,99- Acima do peso.

Fonte: Próprio autor.

O grau de dificuldades, para ambos os exercícios, foi elaborado de maneira equivalente, visando um nível aceitável de comprometimento e concentração do aluno para identificar os problemas nos códigos disponibilizados.

Quadro 42-Código corretos para as questões 01 e 02 – Atividade 01/Caso10

Questão 1c

```
# include<stdio.h>
int main(void)
{
   int qp, qg, qo, qtot,x, mpg, mpo;
   float pg, po, vkg, vko, vg, vo, perg, pero, totv, mg, mo;
   qp = qg = qo = qtot = x = mpg = mpo = 0;
   pg = po = vkg = vko = vg = vo = perg = pero = totv = mg = mo = 0;
   printf ("Quantos Produtores: ");
   scanf("%d",&qp);
   for(x=0; x<qp; x++)
   {
      printf("\n *** PRODUTOR %i *** \n",x);
      printf ("Quantidade de Gado : ");
      scanf("%i",&qg);
      printf ("Peso medio do Gado : ");
      scanf("%f",&pg);
      printf ("Valor do Kg - Gado : ");
      scanf("%f",&vkg);
      printf ("Quantidade Ovinos : ");
      scanf("%i",&qo);
      printf ("Peso medio Ovino : ");
      scanf("%f",&po);
      printf ("Valor do Kg - Ovino : ");
      scanf("%f",&vko);
      vg = (qg*pg*vkg);
      vo = (qo*po*vko);
      totv = vg + vo;
      perg = vg / totv;
      pero = vo / totv;
      printf("Valor da venda de Gado foi %.2f \n", vg);
      printf("Valor da venda de Ovinos foi %.2f \n", vo);
      printf("Percentual da venda de Gado foi %.2f \n", perg);
      printf("Percentual da venda de Ovinos foi %.2f \n", pero);
      if (vg > mg)
      {
         mg = vg;
         mpg = x;
      }
      if (vo > mo)
      {
         mo = vo;
         mpo = x;
      } }
   printf("O Produtor %i vendeu mais gado, com uma renda %.2f \n", mpg, mg);
   printf("O Produtor %i vendeu mais ovinos, com uma renda %.2f \n", mpo, mg);
return 0;
}
```

Questão 2b

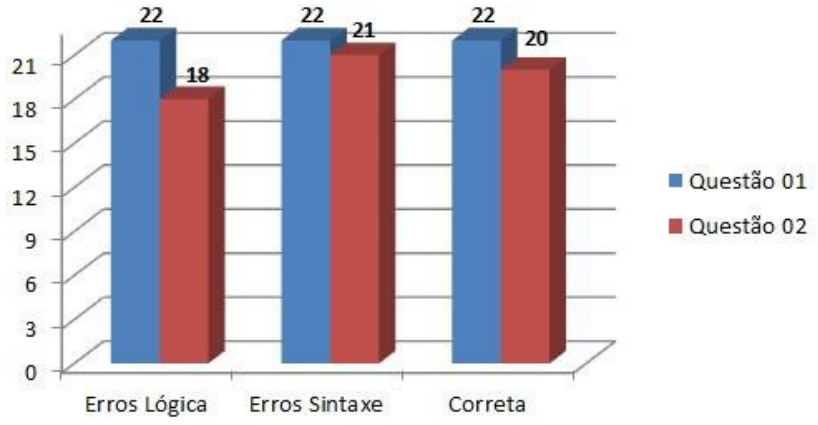
```
# include<stdio.h>
int main(void)
{
   int x;
   float p, a, i;
   x=0;
   while(p!=0 || x==10)
   {
      x++;
      printf("\n *** INDIVIDUO %i *** \n",x);
      printf("Peso: ");
      scanf("%f",&p);
      if(p==0)
      {
         printf("\n Encerrando programa... \n");

      }
      else
      {
         printf("Altura: ");
         scanf("%f",&a);
         i= p / (a*a);
         printf("Índice (IMC): %.1f \n",i);
         if(i>16.99 && i < 18.50)
         {
            printf("Abaixo do Peso... \n");
         }
         if(i>18.49 && i < 25)
         {
            printf("Bom... \n");
         }
         if(i>24.99 && i < 30)
         {
            printf("Acima do Peso... \n");
         }
      }
   }
return 0;
}
```

Fonte: Próprio autor.

A correção desta prática demonstrou que a maioria dos estudantes conseguiu avaliar corretamente os programas, apontando os tipos de erros em cada um dos casos, conforme é apresentado no Gráfico 50. Do total de 23 participantes na atividade, 22 indicaram de forma precisa o algoritmo correto e os códigos com erros de sintaxe e lógica na questão 01, enquanto na segunda questão, 18 alunos destacaram corretamente o programa com problemas de lógica, 21 os de sintaxe e, 20 estudantes encontraram o código correto.

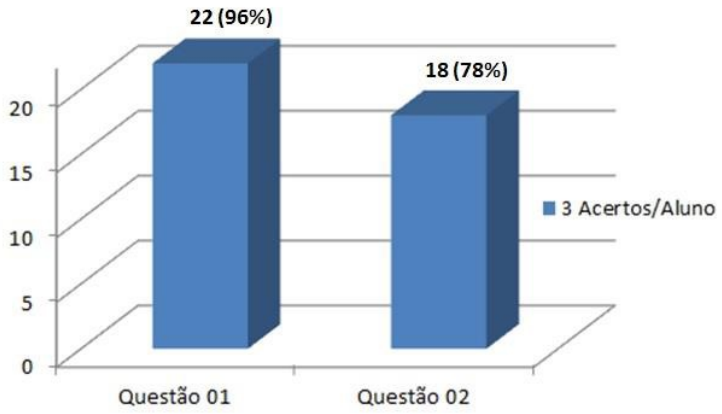
Gráfico 50-Acertos por item - Atividade 01/Caso10



Fonte: Próprio autor.

Por esta se caracterizar como uma atividade formativa, em que foi permitido aos alunos consultarem materiais didáticos durante o exercício, pode-se afirmar que o percentual de acertos verificado demonstra um nível de comprometimento dos discentes para composição de respostas corretas, o que diretamente direcionou a construção do conhecimento sobre este paradigma de programação. Esta observação é apoiada com a informação descrita no Gráfico 51, o qual evidencia que um montante de 96% dos alunos acertou completamente a primeira questão e 78% a segunda. A evidente diferença identificada no número de acertos na segunda prática foi apontada pelos alunos como decorrência da estrutura de repetição apresentada, que pela composição do comando, induzia ao erro.

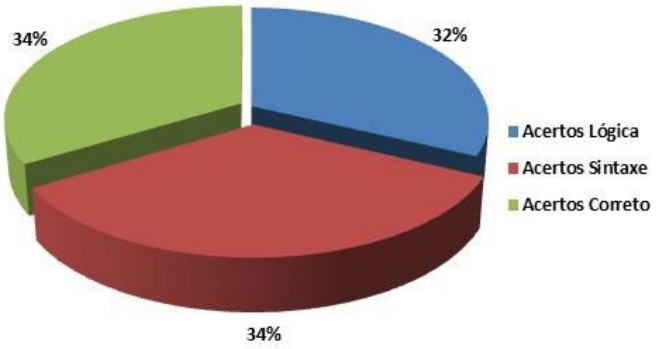
Gráfico 51-Alunos com 3 acertos - Atividade 01/Caso10



Fonte: Próprio autor.

Na última análise do exercício (Gráfico 52) constatou-se o nivelamento em relação às respostas corretas, ou seja, não foi identificada nenhuma tendência na resolução dos problemas, em ambas as questões os alunos demonstraram uma equivalência na dedução dos erros de lógica, sintaxe e também ao apontar os programas funcionais.

Gráfico 52-Percentuais de acertos totais - Atividade 01/Caso10



Fonte: Próprio autor.

Em complemento à primeira atividade, foi disponibilizada aos estudantes uma segunda tarefa com intuito de complementar o Caso10, denominada atividade 02, seguindo a linha da interpretação de algoritmos em linguagem C, contudo com o foco voltado para identificação da funcionalidade e precisão de programas, desenvolvidos para atender um determinado problema. Participaram desta prática o mesmo número de alunos da atividade 01, 23 estudantes, os quais responderam todos os itens da segunda atividade.

A atividade 02 foi composta de duas questões, com demandas específicas, para as quais foram construídas três soluções, cada uma com um conjunto de erros predefinidos e, que poderiam ser classificados da seguinte forma: (1) Mais atende, contemplando programas

corretos, ou com no máximo um erro simples; (2) Atenderia, classe definida para algoritmos com erros de sintaxe ou lógica, exclusivamente, em quantidades relevantes e; (3) Menos atende, grupo de programas com problemas de sintaxe e lógica no mesmo algoritmo. A incitação do raciocínio lógico, crítico e reconhecimento da formalidade, na programação em linha de comando, foi o objetivo desta prática, além de reconhecer padrões e erros, ações definidas como penosa para o processo de aprendizagem de programação (LAHTINEN *et al.* 2005).

Os enunciados da questão 01 e 02 são apresentados no Quadro 43. Buscou-se disponibilizar nos exercícios temas que permitissem ao aluno reconhecer situações reais, passíveis de programação, sendo o primeiro caso (questão 01) voltado a um sistema para o controle de compras e no segundo (questão 02), um *software* para a gerência de viagens.

Quadro 43-Enunciados das questões 01 e 02 – Atividade 02/Caso10

Questão 01	Questão 02
Implemente um programa que registre as compras de um determinado usuário. O sistema deverá computar o valor de todas as compras até o usuário digitar zero ou valor total das compras passar de R$ 1.000,00. Ao final o sistema devera apresentar o valor do maior item comprado, a média de gastos e o total da compra. * Classifique os códigos de acordo com o nível de acerto do algoritmos proposto. * (1) Mais atende; (2) Atenderia; (3) Menos atende.	Construa um programa para auxiliar um viajante a calcular o custo de suas viagens. Para isso saiba que o valor cobrado por viagem varia de acordo com um determinado trecho. O trecho 0 custa R$ 25,00/ km, o trecho 1 custa R$ 50,00/km e o trecho 2 custa R$ 100,00/km. O usuário deve informar a distância de cada trecho e o sistema irá informar: o trecho mais rentável, o trecho menos rentável, o valor total do trecho e a quilometragem percorrida. * Classifique os códigos de acordo com o nível de acerto do algoritmos proposto. * (1) Mais atende; (2) Atenderia; (3) Menos atende.

Fonte: Próprio autor.

No Quadro 44 são apresentados os algoritmos propostos para esta atividade, com a classificação (1) Mais atende, para as duas questões.

Quadro 44-Código corretos para as questões 01 e 02 – Atividade 02/Caso10

Questão 1b

```
# include<stdio.h>
int main(void)
{
    int c, val, tot, ma, me;
    me=0;
    tot=0;
    ma=0;
        c=0;
    while(val !=0 || tot < 1000)
    {
        printf("Digite valor da compra : ");
        scanf("%i",&val);
        if(val < ma)
        {
            me = val;
        }
        tot = tot + val;
    }
    c++;
    me=tot/c;
    printf("O valor do maior item é : %i \n",ma);
    printf("O valor médio dos itens é : %i \n",me);
    printf("O valor total da compra é : %i \n",tot);
    return 0;
}
```

Questão 2b

```
# include<stdio.h>
int main(void)
{
float x, km, kmt, vk1, vk2, vk3, vkt;
for(x=0; x<3; x++) {
  printf("Informe distância no trecho %i : ",x);
  scanf("%f",&km);
  if(x==0) {
    vk1=km*25.00;
  }
  if(x==1) {
    vk1=km*50.00;
  }
  if(x==2) {
    vk1=km*100.00;
  }
}
vkt = vk1 + vk2 + vk3;
if(vk1 == vk2 && vk2 == vk3) {
  printf("Todos os trechos tiveram a mesma rentabilidade, com
%.2f \n",vk1);
}
else
{
  if(vk1 > vk2 && vk1 > vk3) {
    printf("O trecho 01 foi mais rentável com %.2f \n",vk1);
  }
  if(vk2 > vk1 && vk2 > vk3) {
    printf("O trecho 02 foi mais rentável com %.2f \n",vk2);
  }
  if(vk3 > vk1 && vk3 > vk2) {
     printf("O trecho 03 foi mais rentável com %.2f \n",vk3);
  }
  if(vk1 < vk2 && vk1 < vk3) {
    printf("O trecho 01 foi menos rentável com %.2f \n",vk1);
  }
  if(vk2 < vk1 && vk2 < vk3) {
    printf("O trecho 02 foi menos rentável com %.2f \n",vk2);
  }
  if(vk3 < vk1 && vk3 < vk2) {
     printf("O trecho 03 foi menos rentável com %.2f \n",vk3);
  }
printf("O valor total dos trechos foi %.2f \n",vkt);
printf("A distância total percorrida foi %.2f \n",kmt);
return 0;
}
```

Fonte: Próprio autor.

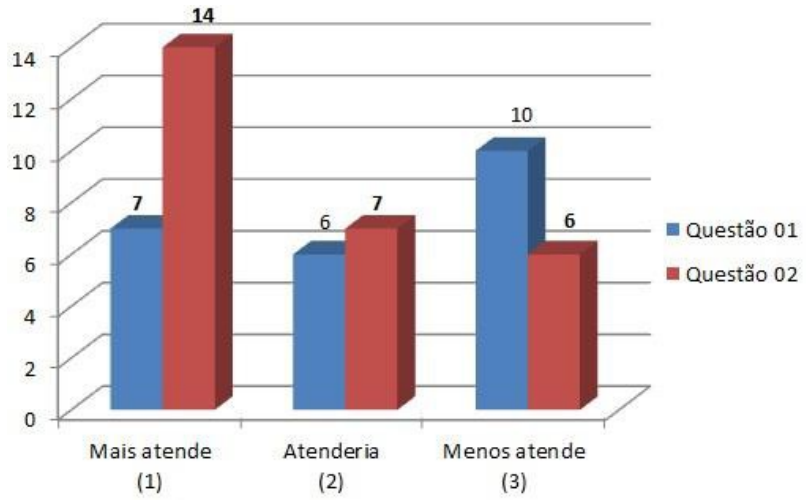
A partir da distinção de cada demanda neste exercício, atentou-se ao número de acertos para as situações abordadas (Gráficos 53), sendo que os alunos obtiveram um melhor desempenho ao analisar os códigos com mais erros, compreendidos como nível (3) Menos atende, na primeira questão, com 10 estudantes classificando corretamente os códigos nesta classe.

Na segunda questão, os estudantes destacaram-se em definir, de forma acertada, os algoritmos pertencentes ao grupo (1) Mais atende, com 14 estudantes enquadrando os códigos corretos desta forma.

O reconhecimento das diferenças entre os índices nas duas questões é passível de ser explicada pela demanda de cada programa, visto que o algoritmo proposto para primeira questão possuía um erro simples, um contador fora da estrutura de repetição, o que induziu os discentes ao erro. Já na segunda questão, o programa passível de ser definido como (1) Mais

atende, não possuía nenhuma inconformidade, ou seja, totalmente correto, o que facilitou a interpretação dos estudantes.

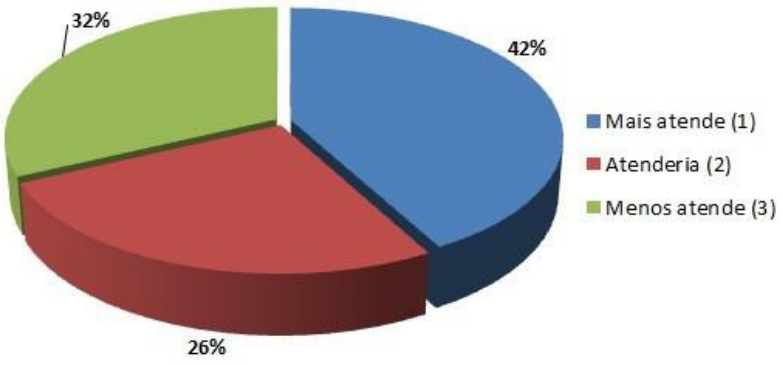
Gráfico 53-Acertos por classificação de código – Atividade 02/Caso10



Fonte: Próprio autor

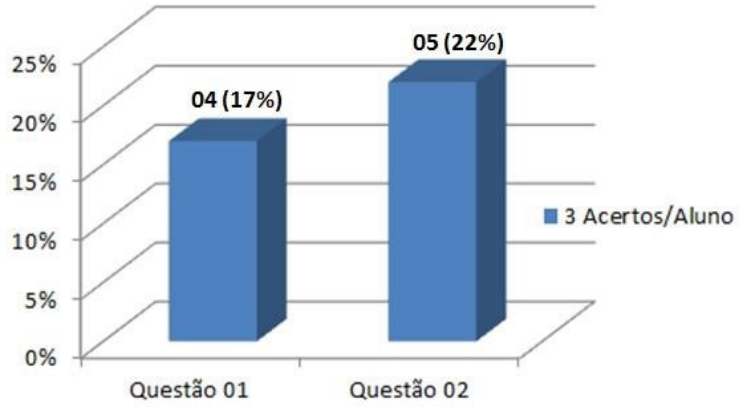
O montante de acertos, traduzidos como índice para cada item (Gráfico 54), destacou que 42% dos discentes conseguiram apontar corretamente os códigos com menos incidência de erros, os quais se enquadraram como respostas corretas para os requisitos. Os algoritmos, menos funcionais, caracterizados por erros de lógica e sintaxe, foram corretamente elencados por 32% dos estudantes, enquanto 26% classificaram de forma acertada códigos que atenderiam o enunciando, contudo com algumas ressalvas a serem corrigidas. A análise destes percentuais mostra que os acadêmicos, na sua maioria, demonstraram a habilidade de reconhecer um programa correto, que pode ser utilizado com resposta a uma demanda específica. Da mesma forma, com um percentual um pouco menor, o grupo exibiu competência de identificar os programas com erros, tanto de lógica quanto de sintaxe e, classificá-los no nível (3). O menor percentual de acertos foi referido aos códigos de nível intermediário (2 - Atenderia), os quais claramente denotaram maior complexidade na sua resolução, devido à necessidade de um pensamento crítico e aptidão em distinguir a quantidade e características dos conflitos nos algoritmos.

Gráfico 54-Percentuais de acertos por item - Atividade 02/Caso10



Fonte: Próprio autor

Gráfico 55-Alunos com 3 acertos - Atividade 02/Caso10



Fonte: Próprio autor

Em comparação a atividade 01, neste caso o quantitativo de acadêmicos que realizaram de forma inquestionável a classificação de todos os itens em cada exercício foi significativamente menor, com 17% na primeira questão e 22% na segunda. Esta tendência já era esperada, visto o aumento na complexidade do trabalho e dificuldade evidenciada pelos acadêmicos em associar as diretivas da linguagem natural com os procedimentos da linguagem de programação imperativa, já retratado por (CAMPOS, 2009) e aqui apresentado no Gráfico 55.

5.1.11. FECHAMENTO DOS CASOS

O eixo que norteia as ações deste trabalho pode ser definido como a intenção de propiciar uma metodologia, considerada efetiva, para o apoio ao processo de ensino e de aprendizagem da lógica, algoritmos e programação. A tese de utilizar diferentes paradigmas

como a programação baseada em blocos visuais, desenvolvimento de aplicações para mundos virtuais e por fim a utilização do paradigma imperativo (linguagem C) buscou estimular as habilidades necessárias à formação de indivíduos capazes de propor soluções computacionais baseadas no Pensamento Computacional e raciocínio lógico efetivo.

Contudo, a formação de um programador possui distintas peculiaridades, pois, para alcançar esta competência o indivíduo deve construir conhecimentos que ultrapassam o entendimento da sintaxe e semântica da linguagem de programação, conforme apontado por Takahashi (1998) e Gomes & Mendes (2000). O raciocínio crítico, lógico e capacidade de pensar computacionalmente são méritos primordiais que o estudante deve apresentar para se tornar um profissional em desenvolvimento.

A fim de identificar a evolução do *know-how* do aluno, relacionado à construção do raciocínio lógico durante a disciplina de algoritmos, esta pesquisa apoiou-se na técnica de RBC, para a implementação de um conjunto de 10 Casos, representados por 11 atividades, contemplando todos os conteúdos previstos para o semestre letivo. A análise dos resultados obtidos a partir de cada Caso ocorreu na etapa de retenção, com a correção e validação das repostas apresentadas pelos alunos e, observação do conhecimento exibido por estes para a composição de novas soluções, as quais se demonstraram fortemente calcadas sobre a estrutura de memória do acadêmico, concebida com as práticas realizadas.

Um conjunto de estratégias diferentes foram utilizadas, uma em cada Caso, com a avaliação de um instrumento de pesquisa (Caso01), a fim de identificar o nível de habilidade dos alunos no início das atividades de algoritmos, observação da capacidade na resolução de problemas de lógica e matemática (Caso02), construção de soluções com recursos gráficos de fluxogramas (Caso03), capacidade de construir programas com base em uma linguagem de programação por blocos visuais, entendendo a facilidade de utilização destes ambientes (Caso04), interpretação de problemas e proposição de soluções (Caso05), identificação de erros em aplicações implementadas com o Scratch (Caso06), migração entre paradigma de programação (do Scratch para o LSL) no Caso07, o Caso08 explorou as características de imersão e motivação, relacionadas com o uso de mundos virtuais para construção de aplicações para o OpenSim, reconhecimento da estrutura e programação simples com linguagem C foram o foco do Caso09 e, por fim, o Caso10 explorou a capacidade dos acadêmicos em interpretar programas escritos sobre um paradigma imperativo, além da capacidade de apontar construções inválidas tanto a nível de sintaxe quanto de lógica, em algoritmos desenvolvidos em C.

Uma evolução gradativa no potencial dos alunos para resolução de problemas foi destacada a cada Caso analisado, contudo a dificuldade dos alunos ao migrar para a linguagem C ficou explícita nas últimas atividades. A formalidade da sintaxe inerente a programação imperativa vai além do raciocínio lógico, percebida pelos índices observados nos Caso 09 e 10. Porém, durante as ações pedagógicas o desempenho demonstrado pelos acadêmicos apontou para uma capacidade diferenciada, em construir soluções para os problemas apresentados. Esta característica ficou exaltada, nos últimos períodos da disciplina, em que os estudantes embora cometessem falhas na descrição dos comandos, idealizaram estruturas algorítmicas factíveis, baseadas em raciocínios válidos, caracterizando desta forma a viabilidade do propósito destes experimentos.

Entendendo a subjetividade das observações e declarações, relacionadas à identificação da construção do raciocínio lógico, convém ressaltar o alinhamento das ações nesta etapa do estudo a ressalva de Queiroz (1988) sobre a importância em concentrar o foco em observações relacionadas diretamente ao objetivo da pesquisa, por meio do controle dos recursos teóricos e metodológicos adotados para análise dos Casos. Estas ações produziram uma experiência evidente (ROMANELLI, 1998), alcançada de forma lenta e gradual, garantindo assim a validação das colocações que foram apresentadas neste capítulo.

5.2. ANÁLISE ESTATÍSTICA

Entendo a pertinência de validar a proposta metodológica defendida por esta Tese, realizou-se uma análise formal dos dados obtidos durante a última etapa dos experimentos, buscando através de procedimentos estatísticos fundamentar as observações realizadas, visto que as técnicas estatísticas basearam-se na coleta, organização e interpretação de informações sobre modelos específicos, segundo Kachigan (1986). Para a execução destas atividades foi utilizado um conjunto de instrumentos, com o objetivo de interpretar os dados numéricos manipulados (MURTREIRA, 1993) e, desta forma obter uma coleção de resultados precisos, esclarecidos e devidamente fundamentados (SILVESTRE, 2007).

Como tarefa inicial para a análise do desempenho dos estudantes, com base nas notas resultantes de suas avaliações, realizou-se uma estatística descritiva, ou seja, uma síntese da série de valores coletados, a fim de proporcionar uma visão global sobre variação destes, utilizando como recursos tabelas, quadros, gráficos, indicadores numéricos e medidas descritivas (GUEDES *et al.* 2015; REIS, 1996). A aplicação e os resultados alcançados com a adoção destas técnicas são apresentados no subcapítulo 5.2.1, o qual, conforme define Morais

(2005), vislumbrou resumir os dados obtidos, disponibilizando um parecer sobre a dispersão e tendência central dos elementos manipulados. Desta maneira foi possível avaliar de forma apurada e formal a performance dos alunos nas diferentes atividades avaliativas propostas no Modelo 2, durante a disciplina de algoritmos no semestre de 2014/2.

No subcapitulo 5.2.2 é realizada uma inferência estatística, também sobre os dados de notas dos participantes do experimento final, em comparação com os demais estudantes que participaram da mesma disciplina com outros docentes, no período desta pesquisa. Segundo Huot (2001) esta análise amostral permite uma observação generalizada da população, estando assim alinhado ao foco desta etapa do estudo, o qual vislumbra um conhecimento científico metódico (ALMEIDA & FREIRE, 2000) sobre as médias dos discentes, almejado por meio de procedimentos e estratégias previamente definidas e cumpridas de acordo com um planejamento específico, constante na concepção do Modelo 2.

.

5.2.1. DESEMPENHO DOS ALUNOS NO EXPERIMENTO

O tempo de execução deste estudo perdurou por quatro semestres consecutivos 2013/1, 2013/2, 2014/1 e 2014/2. Com a realização dos projetos piloto nos primeiros três períodos e com o desenvolvimento do experimento final no segundo semestre de 2014. As atividades realizadas durante o período de 2013/1 não compuseram ações avaliativas, apenas formativas, com o intuito de observar a aceitação dos alunos, em relação à utilização de mundos virtuais em aulas de algoritmos. Nos demais semestres, as atividades constituíram-se por diferentes práticas, inclusive avaliativas.

O Quadro 45 apresenta o universo de participantes deste estudo, com a composição de cada semestre letivo, apontando os dados de matrícula, evasão, frequência e aprovações. Os quantitativos neste quadro se equivalem em todos os campos, contudo destaca-se a evolução no número de aprovados no último período (2014/2). Salienta-se que neste caso, observou-se pontualmente a amostra de estudantes ativos até o final do semestre, com presença acima de 75% nas aulas, em que o percentual de aprovações chegou a um índice de 91% em 2014/2, corroborando com as conjecturas supostas por esta tese na utilização do Modelo 2.

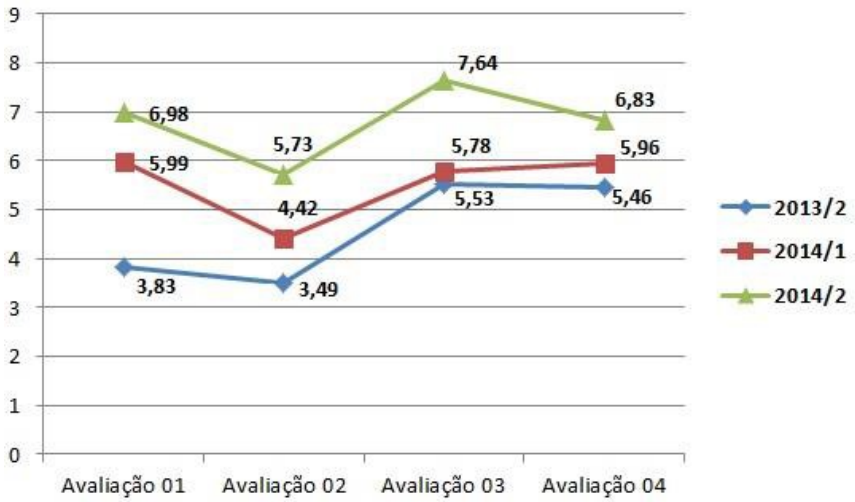
Quadro 45-Universo de alunos participantes de todas as etapas da pesquisa

Matriculas x Evasão x Aprovações (2013/2 - 2014/1 – 2014/2)					
Semestre	Matrículas	Nunca compareceram	Evasão Geral	Frequentes	Aprovados
2013/2	60	27	39	21	13
2014/1	60	30	35	25	15
2014/2	60	24	38	22	20

Fonte: Próprio autor.

Outra evidência encontrada da melhoria no desempenho dos estudantes é exposta ao analisar os resultados alcançados por estes, em atividades avaliativas realizadas nos respectivos semestres, conforme demonstrado no Gráfico 56. O progresso é percebido em todas as quatro avaliações, com um aumento de 0,96 pontos na média de 2014/1 em relação a 2013/2, 1,26 de 2014/2 para 2014/1 e, um avanço significativo de 2,22 pontos alcançados pelos alunos de 2014/2 sobre o grupo de 2013/1.

Gráfico 56-Comparação entre médias das avaliações (2013/2, 2014/1 e 2014/2)



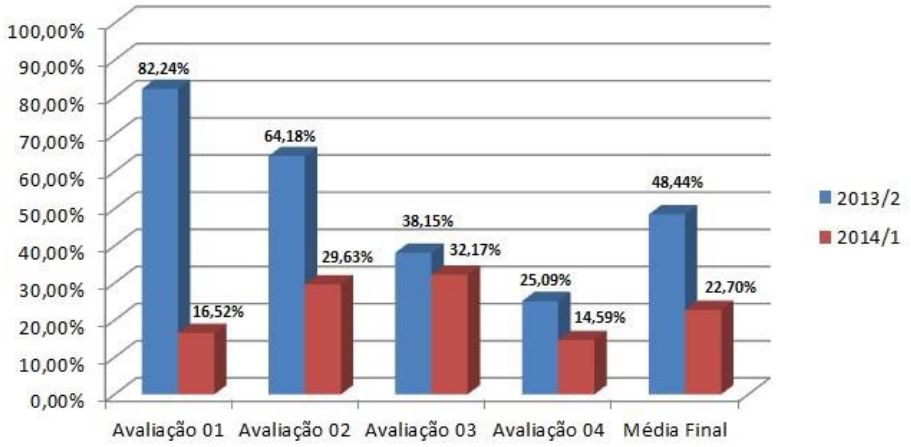
Fonte: Próprio autor

Os índices, relativos ao progresso descrito no Gráfico 56, estão frisados no Gráfico 57, a partir da apresentação do percentual de melhora nas notas do período 2014/2, em referência a 2013/2 (barras azuis) e 2014/1 (barras vermelhas). A distância observada nos dados das avaliações de 2013/2 pode ser descrita como decorrente da mudança de modelo didático, do Modelo 1 (com atividades relacionadas ao uso dos diferentes paradigmas alocadas no final do

semestre) para o Modelo 2 (em que a utilização do Scratch e OpenSim ocorreram no primeiro bimestre).

A variação positiva em relação a 2014/1, considerando a utilização da mesma prática (Modelo 2), é justificada com Moran (2004), pela melhora nos processos de organização dos conteúdos e provocação dos discentes, em busca de uma síntese coerente, resultante da experiência alcançada pelo docente ao questionar a compreensão do aluno e gerar um nível de tensão para superá-la. Também se aponta como fator positivo, a aprendizagem ativa (CHICKERING, 1991; DOS SANTOS, 2001; SHANG *et al*. 2001) inerente a este modelo, baseado em exercícios estruturados, trabalhos colaborativos e desafios, que estimularam o interesse dos acadêmicos com métodos vivenciais por meio do uso de exemplos e atividades em ações cotidianas e reais.

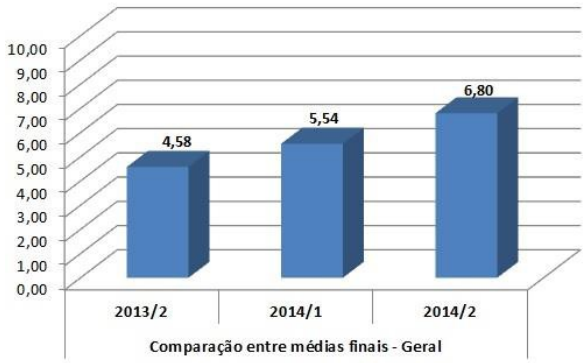
Gráfico 57-Evolução de desempenho dos estudantes (2013/2, 2014/1) em relação a (2014/2)



Fonte: Próprio autor

A diferença das médias finais de todos os alunos, nos semestres observados, é mostrada no Gráfico 58, enquanto as notas apenas dos aprovados são descritas no Gráfico 59. A partir destas, denota-se o gradual incremento dos resultados alcançados pelos estudantes nas disciplinas de algoritmos. A maior margem de intervalo de notas é percebida no grupo geral (Gráfico 58), exibindo um acréscimo médio entre os semestres de 1,11 pontos, enquanto a diferença dos alunos aprovados gira em torno de 21 décimos (0,21).

Gráfico 58-Médias finais - Geral



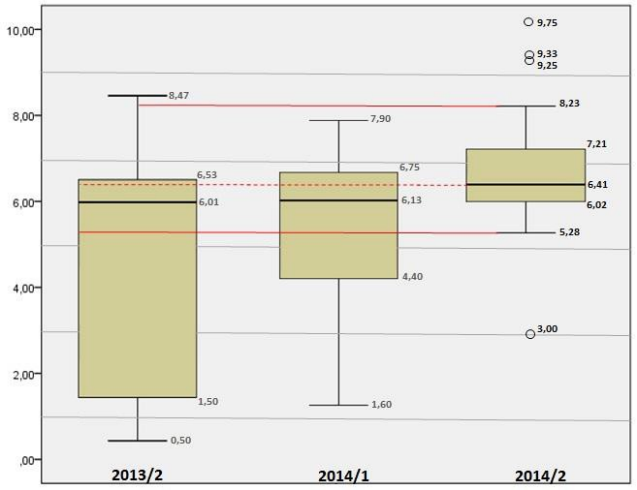
Fonte: Próprio autor

Gráfico 59-Médias finais - aprovados

Fonte: Próprio autor

Por fim, foi criado um gráfico de comparação entre as tendências das médias dos estudantes para o período (Gráfico 60), considerando as notas individuais. Destaca-se nesta representação dos *outliers* dos dados de 2014/2, referentes às notas de 3 acadêmicos (9,25/9,33/9,75), as quais apontaram o bom nível de habilidade de programação destes. O limite superior no BoxPlot de 2013/1 encontra-se acima dos demais semestres, contudo a mediana para este semestre é a menor observada, sobressaindo novamente 2014/2, com um valor de média 7,21 para o terceiro quartil e, 6,41 como mediana, sendo estes dados superiores aos demais períodos. Como informação relevante nesta observação, destaca-se o limite inferior denotado no segundo semestre de 2014, apontando um valor de 5,28, caracterizando que a predisposição para a melhora no desempenho ocorreu de forma linear e equânime, tanto pelos aprovados, quanto pelos discentes com notas reduzidas (abaixo da média).

Gráfico 60-BoxPlot comparativo entre médias (2013/2, 2014/1 e 2014/2)



Fonte: Próprio autor

A identificação do processo efetivo de ensino e de aprendizagem, neste ponto da pesquisa, é corroborada pela dinâmica observada em relação à evolução das notas dos alunos em atividades avaliativas, disponibilizadas durante o período dos experimentos. A clara ascensão no desempenho dos estudantes, tanto pela análise dos casos que compuseram as observações sobre a construção do raciocínio lógico na seção anterior e, pela comprovação quantitativa apontada nesta discussão, indicam a validade das ações e pressupostos esperados como resultados neste estudo.

A legítima conclusão sobre o progresso das notas dos acadêmicos, está calcada sobre a ampliação dos índices de aprovação e redução das reprovações verificadas, demonstrando que a proposta metodológica do Modelo 2 facilitou a compreensão da lógica envolvida na construção de soluções computacionais, assim como a própria ação de programar.

A gradual evolução das médias nas quatro avaliações do período 2014/2, mostrou a capacidade dos alunos em se apropriar de conceitos importantes da área de programação, os quais os possibilitaram executar as tarefas disponibilizadas, relacionando estas ações às suas estruturas prévias de conhecimento (princípios da aprendizagem significativa) e, desta forma, construindo uma memória de casos (teoria do raciocínio baseado em casos - RBC) utilizada como recursos para proposição de novas soluções para os problemas avaliados.

O entendimento destas relações corresponde a uma fração do montante das expectativas geradas durante os estudos desta tese, contudo a melhoria na evolução dos alunos, constatada em relação à amostra de estudantes que participaram das atividades, não são suficientes para a emissão de um parecer positivo sobre a validação do Modelo 2, como prática a ser utilizada em disciplinas de algoritmos e programação. Sendo assim, a fim de ratificar os resultados obtidos até o momento, foi proposta uma inferência estatística, considerando as médias dos acadêmicos no semestre 2014/2, em comparação as notas dos demais alunos que cursaram a mesma disciplina com outros docentes, servindo neste contexto como grupo de controle. Ressalta-se que esta informação poderá servir de base para uma reflexão final e emissão de um parecer sobre a utilização do método (Modelo 2) como uma solução a ser adotada para a melhoria no desempenho dos acadêmicos iniciantes em programação.

5.2.2. COMPARAÇÃO ENTRE MÉDIAS (EXPERIMENTO X GRUPO DE CONTROLE)

A fim de garantir um nível de formalidade e maior certeza da validade dos resultados alcançados com este experimento, esta seção é dedicada a uma inferência estatística sobre as

variáveis amostrais obtidas a partir das atividades avaliativas, realizadas durante a disciplina de algoritmos e programação no segundo semestre de 2014. A execução deste tipo de tarefa, segundo Reis *et al.* (1999), permite avaliar e tomar decisões sobre uma determinada população, sem a necessidade de análise da totalidade de elementos da mesma. Desta maneira, foram utilizados recursos para comparação das notas médias, obtidas pelos alunos participantes dos experimentos em relação ao grupo de controle, com o intuito de identificar o desempenho dos discentes e a validade da proposta desta Tese.

Os testes adotados para inferência estatística buscaram em diferentes momentos confirmar a validade de determinada hipótese, com base no conjunto de dados coletados e procedimentos objetivos, a fim de aceitar ou rejeitar a mesma (SIEGEL & CASTELLAN, 2006). Para alcançar um resultado efetivo, sobre a observação das hipóteses, foram utilizados testes unilaterais (BRIGER, 1946; MARÔCO, 2011), em que a hipótese alternativa (H1) recebe um sinal oposto à hipótese nula (H0). Este tipo de operação é valida para confirmar ou não a diferença entre as médias.

No caso específico, das amostras aqui analisadas, foi possível a realização de três tipos de testes: o teste ou análise de variância, o qual possibilitou identificar a existência de uma diferença significativa entre as médias observadas, ou seja, avaliar as afirmações sobre as médias população relacionada ao estudo (MILONE, 2004); os testes paramétricos, indicados para avaliação de intervalos sobre uma distribuição normal, com variâncias iguais assumidas e com escalas de mensuração que possibilitam operações aritméticas (MATAR, 1998; COOPER & SCHINDLER, 2003); e, os testes não paramétricos, os quais não exigem requisitos tão específicos, como a normalidade da distribuição, contudo esta técnica apresenta resultados menos pontuais em relação aos testes paramétricos correspondentes (MONTGOMERY & RUNGER, 2009), sendo utilizada como complemento as análises realizadas.

Com o intuito de inferir indutivamente as propriedades do universo observado, vislumbrou-se a obtenção de uma decisão estatística que apoie a afirmação da superioridade no desempenho acadêmico, constatado nesta Tese, em relação às notas dos alunos participantes das ações didáticas do Modelo 2 (denominados de Grupo Experimento) sobre os demais discentes das disciplinas de algoritmos nos anos de 2013 e 2014, os quais compuseram o chamado Grupo de Controle.

O Grupo de Controle foi composto pela relação dos alunos que cursaram algoritmos nos semestres de 2013/1, 2013/2, 2014/1 e 2014/2, excluindo as turmas que participaram dos projetos piloto desta tese. A amostra inicialmente estava organizada em 12 turmas, totalizando

505 alunos matriculados, contudo a fim de manter a conformidade dos dados e garantir a uniformidade nos padrões analisados, foram descartados deste universo todos os alunos com registro de infrequência ou trancamento, reduzindo a amostra para 284 estudantes os quais efetivamente participaram de toda disciplina de algoritmos, no semestre matriculado (Quadro 40). A última coluna do Quadro 46 é composta pelas médias dos alunos que integraram o Grupo Experimento.

Quadro 46-Notas do Grupo de Controle e Grupo Experimento

Grupo de Controle												Exp
1	2	3	4	5	6	7	8	9	10	11	12	13
6,60	6,10	6,00	7,50	6,00	0,00	7,80	8,70	3,20	6,70	0,00	5,20	3,00
0,00	6,80	2,10	2,80	0,00	4,90	7,60	7,30	5,10	6,40	0,10	2,20	5,28
6,50	2,70	6,30	2,60	3,60	0,00	0,00	7,50	6,30	6,00	8,10	8,00	6,00
7,50	6,70	9,10	0,80	6,85	9,80	6,90	0,00	0,00	4,00	6,10	0,00	6,00
6,50	8,30	6,00	6,40	8,47	7,60	8,30	3,40	0,00	8,00	6,50	0,20	6,00
0,00	7,10	6,10	7,90	7,28	8,50	9,20	0,00	6,10	6,80	8,20	6,70	6,01
1,10	2,50	6,00	3,10	6,04	8,10	1,60	7,20	6,70	6,90	8,10	1,20	6,05
0,00	2,50	8,10	6,90	0,00	4,20	9,40	0,40	6,30	6,60	9,00	7,80	6,10
0,20	7,40	6,50	6,30	7,66	8,80	6,50	9,80	6,00	8,00	6,30	8,10	6,30
5,60	2,30	0,70	6,80	0,00	6,80	6,00	0,00	9,20	6,00	0,20	6,60	6,35
4,40	6,90	6,30	7,10	7,34	7,20	8,70	6,00	7,40	6,20	0,80	1,50	6,36
6,80	6,80	6,30	6,30	4,17	6,00	7,70	1,90	6,90	6,50	1,70	6,60	6,45
0,00	7,60	8,80	10,00	0,00	0,00	8,00	6,80	4,50	7,40	0,50	1,40	6,74
0,00	7,40	0,00	7,30	6,02	8,70	9,20	9,50	7,30	6,60	3,40	0,80	6,84
9,30	7,70	9,40	8,60	6,53	6,80	0,00	0,00	0,00	7,00	7,10	4,80	6,88
1,50	4,30	6,40	8,00	6,04	7,60	8,30	6,10	0,00	7,40	6,30	6,80	7,15
0,60	6,00	9,50	6,10	6,03	0,00	8,10	6,20	9,30	6,00	6,10	7,70	7,23
0,00	6,40	8,30	1,00	0,00	6,00	0,00	5,60	0,00	6,00	7,60	7,00	7,88
0,00	2,40	4,90	7,50	6,03	0,00	6,50	2,50	1,70		5,40	0,10	8,23
6,50	7,60	8,00	1,80	6,01	6,10		0,00	0,00				9,25
2,60	0,00	6,80	8,00	3,97	6,70		2,70	7,00				9,33
0,00	8,70	8,10	0,90	4,49	0,70		6,80	5,30				9,75
2,00	7,40	8,70	0,70	2,18	6,40		6,70	9,50				
6,30	2,60	10,00	1,20	4,54	9,30		4,40	4,40				
0,00	4,40	3,90	0,00		8,90		6,40	6,20				
5,00	6,00	6,30	3,50		8,60		0,00	7,00				
			0,00					4,30				
								5,20				

Fonte: Próprio autor.

Para qualquer inferência estatística o fundamental para obtenção de resultados válidos, é que o pesquisador reconheça a estrutura de dados amostrais a serem analisados, a fim de

definir os métodos a serem aplicados. Desta forma, a primeira ação realizada neste estudo contemplou a verificação da distribuição dos valores do Grupo de Controle e Grupo de Experimento, a fim de reconhecer a normalidade ou não destes.

Ainda com a utilização do *software* SPSS, buscou-se determinar se os dados das variáveis foram modelados sobre uma distribuição normal, para isso definindo duas hipóteses para ambos dos grupos:

H0: Os dados da variável analisada se originam de uma distribuição normal;

H1: Os dados não se originam de uma distribuição normal;

Utilizou-se o teste de Shapiro-Wilk (SHAPIRO & WILK, 1965; ROYSTON, 1992; RAZALI & WAH, 2011; DOUFOUR *et al*. 1998) para verificação da normalidade do Grupo Experimento, visto que este teste apresenta resultados mais precisos para amostras menores que 50 elementos. O Quadro 47 apresenta a saída da aplicação, em que se destaca o grau de significância observado para a variável, com um $p < 0,05$, ou seja, para o $df < 50$ e um *sig* de 0,41 rejeita-se a hipótese nula, apontando que a relação dos dados não segue uma distribuição normal.

Quadro 47-Análise de Normalidade do Grupo Experimento

Testes de Normalidade

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estatística	df	Sig.	Estatística	df	Sig.
Experimento	,207	22	,015	,907	22	,041

a. Correlação de Significância de Lilliefors

Fonte: Próprio autor.

A observação da normalidade da variável Grupo de Controle foi realizada com a adoção do teste Kolmogorov-Smirnov (MASSEY, 1951; LILLIFORS, 1967; RAZALI & WAH, 2011), visto que este é indicado para análise de amostras maiores que 50 elementos. O resultado alcançado, também reconheceu que os dados do Grupo de Controle não respeitavam uma distribuição normal, com um p (0,000) < 0,05, aceitando a hipótese H1, conforme o Quadro 48.

Quadro 48-Análise de Normalidade do Grupo Experimento

Testes de Normalidade

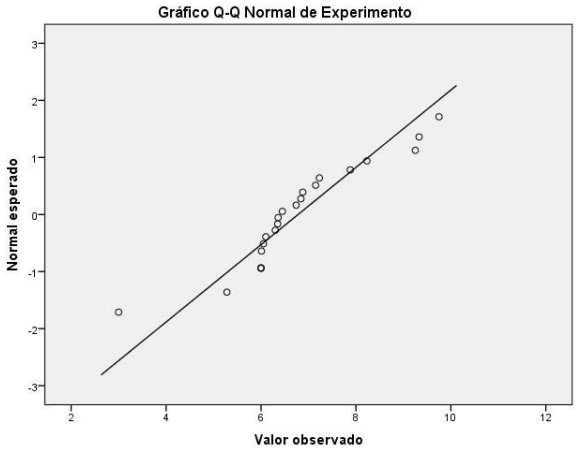
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estatística	df	Sig.	Estatística	df	Sig.
GrupoControle	,217	284	,000	,890	284	,000

a. Correlação de Significância de Lilliefors

Fonte: Próprio autor

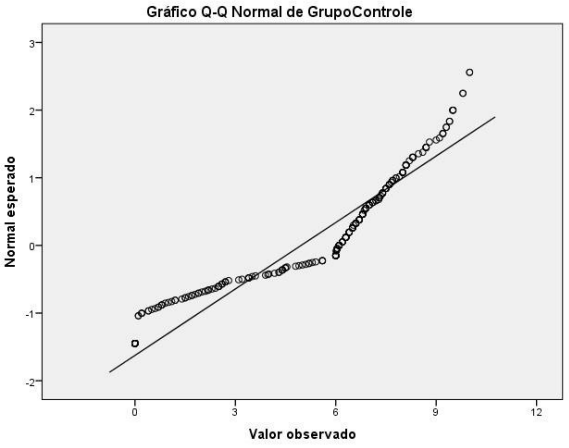
Os testes diagnosticaram que ambas as variáveis não seguem uma distribuição normal, contudo ao observar os Gráficos Quantil-Quantil (BARBETTA *et al.* 2004) para os Grupos, percebe-se que as tendências são próximas a normal (Gráfico 61 e 62).

Gráfico 61-Gráfico QQ para Grupo Experimento



Fonte: Próprio autor

Gráfico 62-Gráfico QQ para Grupo de Controle



Fonte: Próprio autor

Neste caso específico, utilizou-se o conceito de Leech *et al.* (2005), para comprovar a normalidade das amostras. O autor afirma que a observação do índice de assimetria dos dados possibilita validar a distribuição como normal, para isso estes quantitativos devem estar em um intervalo maior que -1 e menor que 1. Ao observar os valores de assimetria constatou-se os índices 0,032 (Quadro 49) e 0,145 (Quadro 50) para o Grupo Experimento e Grupo de Controle, respectivamente.

Quadro 49-Estatística descritiva Grupo Experimento

			Estatística	Erro Padrão
Experimento	Média		6,7809	,31466
	95% Intervalo de Confiança para Média	Limite inferior	6,1265	
		Limite superior	7,4353	
	5% da média aparada		6,8166	
	Mediana		6,4050	
	Variância		2,178	
	Desvio Padrão		1,47590	
	Mínimo		3,00	
	Máximo		9,75	
	Intervalo		6,75	
	Intervalo interquartil		1,38	
	Assimetria		,032	,491
	Curtose		1,524	,953

Fonte: Próprio autor

Quadro 50-Estatística descritiva Grupo de Controle

			Estatística	Erro Padrão
GrupoControle	Média		4,9639	,18123
	95% Intervalo de Confiança para Média	Limite inferior	4,6072	
		Limite superior	5,3206	
	5% da média aparada		4,9863	
	Mediana		6,1000	
	Variância		9,328	
	Desvio Padrão		3,05422	
	Mínimo		,00	
	Máximo		10,00	
	Intervalo		10,00	
	Intervalo interquartil		5,14	
	Assimetria		-,482	,145
	Curtose		-1,126	,288

Fonte: Próprio autor

As constatações alcançadas determinam que para a comparação das médias é possível utilizar tanto testes paramétricos (distribuições normais) quanto testes não paramétricos (distribuições diferentes de normais).

A partir do reconhecimento formalizado dos padrões de distribuição das notas, referentes aos grupos, foram elencados três métodos para a comparação entre as médias dos alunos: um teste de variância (ANOVA), um teste paramétrico (Teste t) e um teste não paramétrico (Mann-Whitney). A redundância nas observações permitirá comprovar se realmente as médias alcançadas no semestre, pelos alunos que participaram do experimento, foram maiores que as médias demonstradas pelo Grupo de Controle.

5.2.2.1. ANÁLISE DE VARIÂNCIA (ANOVA)

O teste de variância ANOVA, do acrônimo *Analysis of Variance* (MARTINEZ & FERREIRA, 2007) demonstra a existência ou não de diferenças internas em uma determinada amostra, assim como variabilidade existente em um grupo amostral. O método calcula a distribuição Z e verifica se este valor é superior ao Z crítico. A validade desta afirmação leva a rejeição da hipótese nula, ou seja, não existem diferenças entre as médias.

As hipóteses para o teste entre médias, aqui apresentado, seguem a seguinte definição:

H0: $\mu_1 = \mu_2 = \mu_{3...} = \mu_{12}$

H1: $\mu_1 \neq \mu_2 \neq \mu_{3...} \neq \mu_{12}$

Justifica-se a adoção deste método, como diagnóstico inicial, para a verificação da existência de variação entre os dados, permitindo a aplicação de testes mais precisos para a identificação da ordem de tais diferenças, caso existam.

Como primeira ação, nesta técnica, realiza-se o teste de Levene (LEVENE, 1960; SCHULTZ, 1985; LIM & LOH, 1996), com o intuito de identificar o nível de homogeneidade nas variâncias do universo das 13 turmas avaliadas. O Quadro 51 demonstra a saída para este teste, em que se percebe uma significância (*sig* = 0,000), assumindo p<0,05, determinando assim a não existência de igualdade de variância entre os elementos, rejeitando assim H0.

Quadro 51-Teste de Levene – Análise de Homogeneidade

Medias

Estatística de Levene	df1	df2	Sig.
6,283	12	293	,000

Fonte: Próprio autor

O Quadro 52 apresenta o nome da variável independente (Médias), a soma dos quadrados, quadrados médios, valor do teste Z e a significância (*sig*), a qual define ou não a rejeição da hipótese nula. O resultado apontou para uma significância baixa (*sig* = 0,000) o que indica p<0,05. Esta afirmação foi positiva para este estudo, visto que comprova a existência de pelo menos uma diferença entre os grupos observados, rechaçando a hipótese H0, mostrando assim que a variabilidade entre os grupos demonstrou-se suficientemente ampla em relação à variabilidade interna dos mesmos.

Quadro 52-ANOVA

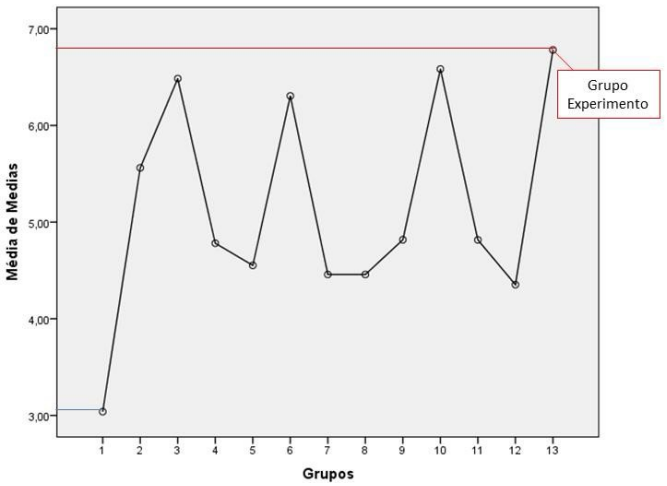
Medias

	Soma dos Quadrados	df	Quadrado Médio	Z	Sig.
Entre Grupos	341,013	12	28,418	3,452	,000
Nos grupos	2412,035	293	8,232		
Total	2753,048	305			

Fonte: Próprio autor

Ao examinar o gráfico referente às médias da amostra (Gráfico 63), destaca-se a primeira turma (1) com o menor valor e, a turma do experimento (13) com a maior nota válida, próximo a 7.

Gráfico 63-Gráfico de Médias - ANOVA



Fonte: Próprio autor

A conclusão deste processo de análise caracteriza a existência de uma diferença inquestionável entre as notas de médias tratadas, assim como evidencia indícios sobre os aspectos positivos inerentes a proposta do Modelo 2, exposto pela posição dos elementos no gráfico de parcelas de médias (Gráfico 67), referentes ao Grupo Experimento.

5.2.2.2. COMPARAÇÃO DE MÉDIAS - TESTE T DE STUDENT (PARAMÉTRICO)

Reconhecendo que existe algum nível de diferença entre as amostras utilizadas neste estudo, evidenciou-se a necessidade da utilização de um teste para validação da diferença, de forma pontual, entre as notas do Grupo de Controle com as do Grupo Experimento. Para isso adotou-se o Teste t de Student (BOX, 1987) para amostras independentes, como solução para esta tarefa, devido à exatidão e formalidade deste método, além do fato dos dados analisados seguirem uma distribuição normal.

Para a execução do teste definiu-se como hipótese nula, a inexistência de diferença entre as médias dos grupos e, como hipótese 1, médias distintas, como segue:

H0: $\mu_{GrupoExperimento} = \mu_{GrupoControle}$

H1: $\mu_{GrupoExperimento} \neq \mu_{GrupoControle}$

O início da avaliação por este método permitiu reconhecer as médias aritméticas de cada grupo, conforme Quadro 53. O Grupo Experimento com 22 itens apresentou uma média

de 6,78, enquanto o Grupo de Controle com 284 elementos alcançou uma média geral de 4,96.

Quadro 53-Análise de Médias

	Grupos	N	Média	Desvio Padrão	Erro padrão da média
Medias	Experimento	22	6,7809	1,47590	,31466
	Grupo de Controle	284	4,9639	3,05422	,18123

Fonte: Próprio autor

A avaliação de homogeneidade de variância (Teste de Levene) disponibilizou um grau de significância (*sig*=0,000), ou seja, p<0,05, revelando que as médias possuem variâncias iguais não assumidas. Observando o *p-value*, reconhecendo as variâncias, encontra-se um *sig*=0,000<0,05. A partir destes resultados é possível descartar a hipótese nula e, assumir H1 como válida, formalizando a afirmação de que realmente existe uma diferença entre as médias do Grupo Experimento e do Grupo de Controle (Quadro 54).

Quadro 54-Teste t para amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Medias	Variâncias iguais assumidas	25,475	,000	2,762	304	,006	1,81700	,65777	,52263	3,11137
	Variâncias iguais não assumidas			5,004	36,942	,000	1,81700	,36312	1,08120	2,55280

Fonte: Próprio autor

A realização deste teste permitiu reconhecer a diferença nas médias, entre as amostras independentes, contudo por se tratar de um método bilateral não foi possível identificar se as médias do Grupo Experimento realmente superaram as médias do Grupo de Controle, apenas que estas são distintas, o que é um fato positivo, contudo não satisfaz as demandas deste estudo. A verificação da média aritmética dos valores das amostras evidenciou que as notas dos alunos do experimento ficaram acima dos demais, contudo estatisticamente não foi possível chegar a esta resolução.

5.2.2.3. COMPARAÇÃO DE MÉDIAS - TESTE DE MANN-WHITNEY (NÃO PARAMÉTRICO)

Este é um teste adequado para realizar a comparação entre médias de amostras independentes, semelhante ao Teste t, porém não tão robusto (WILCOXON, 1945; MANN & WHITNEY, 1947). Normalmente o teste de Mann-Whitney (MANN & WHITNEY, 1947) é adotado como alternativa ao Teste t, contudo especificamente neste trabalho é utilizado como complemento, pois busca-se por meio deste identificar se as médias do Grupo Experimento são realmente maiores que as do Grupo de Controle.

Para a verificação dos resultados esperados realizou-se um teste unilateral à esquerda, definindo como hipóteses iniciais:

H0: $\mu_{GrupoControle} \geq \mu_{GrupoExperimento}$

H1: $\mu_{GrupoControle} < \mu_{GrupoExperimento}$

Para o nível de significância α a regra geral é rechaçar a hipótese nula se *p-value* for menor ou igual a 0,05, para um intervalo de confiança de 95%. O Quadro 55 apresenta as estatísticas do teste de Mann-Whitney. Visto a natureza desta comparação, é necessário avaliar a significância exata (1 extremidade), a qual apresenta um p=0,018, assumindo assim p<0,05 e pelas regras do teste rejeitando a hipótese nula (H0). Baseado nestes dados conclui-se que as médias do Grupo de Controle realmente são inferiores as notas do Grupo Experimento.

Quadro 55-Teste Mann-Whitney para amostras independentes

	Medias
U de Mann-Whitney	2288,000
Wilcoxon W	42758,000
Z	-2,094
Significância Sig. (2 extremidades)	,036
Sig exata (2 extremidades)	,036
Sig exata (1 extremidade)	,018
Probabilidade de ponto	,000

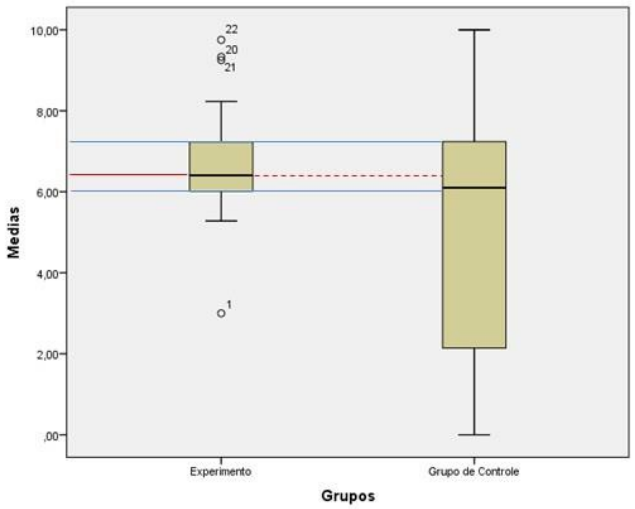
a. Variável de Agrupamento: Grupos

Fonte: Próprio autor

A representação em BoxPlot (Gráfico 64) de ambos os grupos reforça alegação sobre o melhor desempenho dos alunos do Grupo Experimento, a partir da identificação de que a mediana e último quartil encontra-se acima do Grupo de Controle, além da percepção do

grupamento das notas estarem acima de 5,00 o que demonstra uma maior homogeneidade na evolução dos alunos com as práticas calcadas no Modelo 2.

Gráfico 64-BoxPlot comparativo entre Grupo Experimento e Grupo de Controle



Fonte: Próprio autor

Portanto, os alunos do Experimento apresentaram notas mais elevadas do que os discentes das aulas tradicionais no mesmo período, as diferenças observadas foram estatisticamente significativas com U=2288,000; W=42758,000 e p=0,018 (Quadro 55).

5.3. FECHAMENTO DO EXPERIMENTO FINAL

O experimento final foi elaborado com o intuito de validar uma proposta de prática didática para o ensino de algoritmos, denominada Modelo 2, constituída pela integração dos paradigmas de blocos de programação visual (Scratch), programação para mundos virtuais (LSL) e desenvolvimento com uma linguagem imperativa (Linguagem C).

O estudo buscou identificar a evolução na capacidade de desenvolvimento de algoritmos e programas dos alunos, ao interagir com a nova organização dos conteúdos na disciplina, considerando o fato deste modelo ter sido organizado em sincronia com as necessidades dos cursos para os quais ele foi implementado, respeitando a carga horária semestral (60h) e a utilização da Linguagem C, questões estas definidas como requisitos para a disciplina de algoritmos e programação.

Os resultados apontados pelos Projetos Pilotos 1, 2 e 3 viabilizaram esta última etapa da pesquisa, a qual teve como objetivo principal observar e identificar o desenvolvimento do

raciocínio lógico, construção do Pensamento Computacional e desempenho acadêmico nas atividades avaliativas.

5.3.1. CONSTRUÇÃO DO PENSAMENTO COMPUTACIONAL

O raciocínio lógico e a capacidade para desenvolver soluções algorítmicas efetivas, para diferentes problemas, foram analisadas a partir de um conjunto de ações, conteúdos e atividades organizados na forma de casos. Estes casos foram amplamente estudados, conforme já apresentado, em que os alunos participantes dos experimentos demonstraram um progresso gradual, desde o entendimento das noções básicas, necessárias para organizar pensamento, até a composição de aplicações em três diferentes paradigmas.

Contudo, sendo esta uma pesquisa formal buscou-se um método para validar efetivamente as observações sobre as competências relacionadas ao Pensamento Computacional, adquiridas pelos estudantes durante os experimentos desta tese de Doutorado. Para isso propôs-se uma taxonomia para o reconhecimento de tais aptidões, a qual foi fundamentalmente calcada nesta área de estudo e, aqui denominada (TAPC) Taxonomia para Avaliação do Pensamento Computacional.

Wing (2006) definiu o termo Pensamento Computacional como um processo que pode ser realizado tanto por máquinas quanto por seres humanos, descrevendo-o como uma habilidade fundamental a todos os indivíduos, ou seja, este se resume a capacidade analítica utilizada para resolução de problemas pelo indivíduo, a qual pode ser empregada na concepção de sistemas e também para o entendimento do comportamento humano, considerando suas ações de maneira intrínseca e subjetiva.

No ano de 2009 um grupo de educadores, com interesses em comum sobre o PC, foi reunido com a coordenação da Associação de Professores da Ciência da Computação (CSTA) e da Sociedade Internacional de Tecnologia na Educação (ISTE) a fim de definir uma linguagem padrão para o Pensamento Computacional e estratégias para a sua implementação em sala de aula, a fim de integrá-lo à área da educação (PK-12, 2009). Como resultado deste projeto foi estabelecido um conjunto de elementos essenciais para o PC e, a caracterização destes, como objeto de aprendizagem para estudantes de todos os níveis. Visto a aplicabilidade do PC, vislumbrou-se a internalização destas habilidades, com o intuito de que as mesmas pudessem ser transferidas para novos ambientes e utilizadas como recursos válidos para a resolução de problemas em diferentes situações (BARR *et al.* 2011). Como resultados apontados pelo projeto PK-12 o Pensamento Computacional (PC) foi definido como um

processo para solução de problemas que inclui as capacidades de (PK-12, 2009; BARR *et al.* 2011):

• formular problemas factíveis de serem solucionados com computador;

• organizar e analisar dados de forma lógica;

• representar dados por meio de abstrações;

• automatizar ações de forma algorítmica (composição ordenada de passos);

• otimizar soluções através da combinação eficiente de passos e recursos;

• generalizar o processo de solução de um problema para diferentes situações.

Em relação à dimensão de habilidades (disposições e atitudes) necessárias ao PC, destacaram-se:

• confiança em lidar com a complexidade;

• persistência em resolver problemas difíceis;

• tolerância à imprecisão;

• faculdade para resolver problemas abertos;

• competência para atividades colaborativas, a fim de atender objetivos e soluções comuns.

Baseado nestes processos e habilidades foi proposta a TAPC, a qual está definida por duas dimensões: (DC) Dimensão de Competências e (DH) Dimensão de Habilidades. Cada uma destas, composta por quatro elementos de avaliação, seguindo as proposições do projeto PK-12. Estes elementos estão descritos no Quadro 56, sendo que durante a sua aplicação, cada um, pode ser pontuado de acordo com 3 níveis diferentes, em uma escala de 1 a 3 (1-baixo, 2-regular e 3-alto).

Quadro 56-Composição da taxonomia (Dimensões)

Dimensão de Competências (DC)		Dimensão de Habilidades (DH)	
DC1	Organizar dados de forma lógica	**DH1**	Confiança em lidar com a complexidade
DC2	Representar dados através de abstrações	**DH2**	Persistência em resolver problemas
DC3	Automatizar ações de forma algorítmica	**DH3**	Faculdade de resolver problemas abertos
DC4	Generalizar processos para solução de problemas	**DH4**	Potencial para atividades colaborativas

Fonte: Próprio autor

Além das dimensões, vislumbrando validar a execução das atividades, inseriu-se na estrutura desta taxonomia um componente denominado (SRC) Status da Resolução dos Casos, o qual serve para indicar o nível de desempenho dos alunos, podendo assumir os seguintes valores: 1-Caso não realizado; 2-Caso não finalizado; 3-Caso resolvido.

A fórmula definida para este método de avaliação do Pensamento Computacional é apresentada na Equação 05, caracterizada pelo produto das médias das dimensões multiplicado pelo *status* de resolução dos Casos.

(5)

$$TAPC = \left(\frac{\mu DC + \mu DH}{2} \right) X\ SRC$$

Devido ao fato do conjunto de elementos, que compõem as dimensões, possuir um caráter muito amplo, não sendo possível a sua aplicação em todos os Casos, optou-se por caracterizar como Não Aplicável (NA) as dimensões enquadradas nestas situações. Desta forma, o valor da mesma não é computado e não influencia no cálculo realizado. Esta solução busca um equilíbrio para o processo classificação da TAPC.

Os resultados apontados por esta taxonomia devem respeitar uma escala de 9 pontos, para classificação do nível de utilização do Pensamento Computacional, estando organizado em baixo, regular e alto, conforme o Quadro 57.

Quadro 57-Níveis de classificação da TAPC

Intervalo Valores	Nível PC
1,00 – 2,99	Baixo
3,00 – 5,99	Regular
6,00 – 9,00	Alto

Fonte: Próprio autor

A aplicação da TAPC sobre os 10 Casos descritos no capitulo 5.1, permitiu a implementação do Quadro 58, onde são apresentados os valores médios das dimensões, o *status* de cada atividade e o grau alcançado.

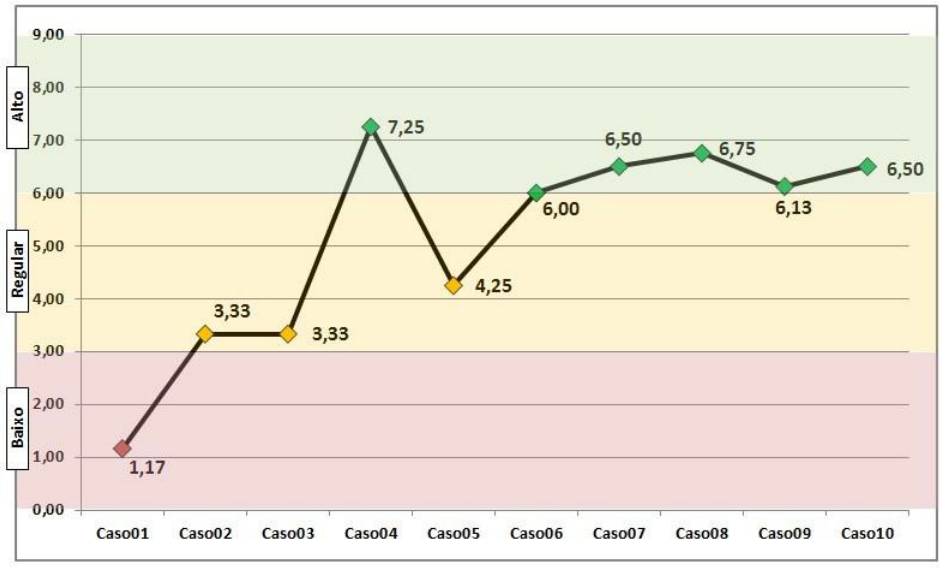
Quadro 58-Inferência da TAPC

Relação de Casos	μDC	μDH	SRC	Grau TAPC
Caso01	1,00	1,33	1	1,17
Caso02	1,33	2,00	2	3,33
Caso03	1,67	1,67	2	3,33
Caso04	2,50	2,33	3	7,25
Caso05	2,00	2,25	2	4,25
Caso06	2,33	1,67	3	6,00
Caso07	2,00	2,33	3	6,50
Caso08	2,00	2,50	3	6,75
Caso09	1,75	2,33	3	6,13
Caso10	2,00	2,33	3	6,50

Fonte: Próprio autor

O Gráfico 65 demonstra a representação dos dados inferidos pela TAPC, através do qual foi possível observar a evolução dos discentes na utilização do Pensamento Computacional, ao realizar as atividades propostas durante o experimento com o Modelo 2.

Gráfico 65-Representação da TAPC para os Casos observados



Fonte: Próprio autor

A utilização da TAPC permitiu validar o Experimento Final desta Tese, no que consente a construção do Pensamento Computacional, visto que ao observar os resultados alcançados com a análise dos Casos desenvolvidos pelos alunos (Gráfico 65), denota-se uma clara ascensão no que tange as capacidades e habilidades demonstradas durante a execução das tarefas propostas.

O Caso01, no início da disciplina de algoritmos, foi o único a resultar em um nível baixo em relação à utilização do PC. Este resultado pode ser atribuído ao fato de ser uma das primeiras atividades da disciplina, considerando que os alunos neste momento, não demonstraram interesse pelos assuntos abordados. Os Casos 02 e 03 obtiveram o mesmo índice (regular), enquanto o quarto Caso caracterizou-se como um *outlier* nas observações (nível alto), o que é justificado por este ser uma atividade de programação livre utilizando o Scratch, o que comprova o caráter motivacional e a facilidade no desenvolvimento com o paradigma de programação com blocos visuais. No Caso05 os alunos demonstraram dificuldade na construção da lógica para interpretar os códigos (nível regular), contudo nos demais Casos (06, 07, 08, 09 e 10) os estudantes demonstram competências pertinentes à aptidão para o uso do PC como intuito de propor soluções viáveis para os problemas apresentados, ficando enquadrados em um nível alto na taxonomia.

Desta forma, é factível afirmar, que o método de aprendizagem denominado Modelo 2 pode ser utilizado como uma ferramenta efetiva para o estímulo à construção do Pensamento Computacional por alunos em uma disciplina de algoritmos e programação.

5.3.2. DESEMPENHO ACADÊMICO

A última ação deste estudo foi provar a hipótese que os discentes que participaram das práticas propostas com o Modelo 2, apresentaram melhor desempenho acadêmico do que os demais estudantes de outras disciplinas de algoritmos. Para alcançar este objetivo foi utilizado um conjunto de três técnicas estatísticas para a comparação de médias (Teste de variância, Teste t e Teste Mann-Whitney). Todas as inferências realizadas apontaram para uma mesma conclusão, que o desempenho das médias observadas foram estatisticamente superiores para os alunos que compuseram o Grupo Experimento em relação aos estudantes do Grupo de Controle.

Sendo assim, neste ponto da pesquisa é possível garantir que a construção de uma disciplina para o ensino de iniciantes em programação pode seguir um arranjo distinto, baseado na adoção do Scratch, Scratch4OS, Open Simulator (LSL) e Linguagem C. Todos estes paradigmas demonstram-se viáveis de serem integrados em um mesmo período e, devido as suas características particulares como aspectos visuais, imersão e motivação estimularam o estudante a construir o conhecimento sobre lógica e programação.

6. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A reconhecida dificuldade dos estudantes em construir o conhecimento sobre a lógica e programação de computadores é tema de diferentes estudos, publicadas ao longo dos tempos. A incapacidade de desenvolver um raciocínio lógico para a proposição de soluções algorítmicas, assim como o reduzido potencial para abstrair a problemática envolvida na concepção de programas são fatores amplamente divulgados.

O reconhecimento destes aspectos serviu como elemento motivador para nortear os estudos desta Tese, a fim de entender a raiz do problema e propor uma solução factível, que pudesse ser adotada como prática pedagógica para o ensino de algoritmos e programação para os indivíduos, cidadãos do mundo atual. Esta proposição se torna válida, ao reconhecer que o contato do ser humano com distintas tecnologias exige a capacidade para utilização de elementos computacionais visando à resolução de problemas, aumentando desta forma o poder cognitivo, operacional, possibilitando um aumento das capacidades criativas e produtivas do ser humano (BLIKSTEIN, 2001; SICA, 2011; FRANÇA *et al.* 2012).

A partir do levantamento de um amplo referencial teórico foram observados importantes aspectos relacionados a esta temática, partindo da avaliação das práticas pedagógicas no ensino de algoritmos, desde o modelo tradicional, até a utilização de jogos e outras tecnologias como a realidade aumentada.

Uma questão pertinente foi a percepção na flexibilidade das orientações contidas nas Diretrizes Curriculares Nacionais, para os cursos na área de computação, a qual proporciona um grau de independência, para a instituições, em relação aos conteúdos pertinentes aos Projetos Políticos Pedagógicos, das disciplinas de algoritmos e programação. Esta característica foi comprovada a partir da observação da organização didática e linguagens programação adotadas nas 34 disciplinas, relacionadas à área de algoritmos, das 25 instituições ensino pesquisadas. Com base neste contexto, durante este estudo, diferentes paradigmas de programação foram avaliados, com o intuito de reconhecer a sua dinâmica de

utilização e, desta forma, identificar a viabilidade de utilização das mesmas como ferramentas efetivas no processo de ensino e aprendizagem da lógica e programação.

Além destes recursos, foram investigadas as teorias de aprendizagem passíveis de serem utilizadas para subsidiar a construção de uma estratégia pedagógica.

O estudo de trabalhos e pesquisas correlatas permitiu uma análise dos diversos métodos e proposições, focados na concepção de auxiliar o estudante a se apropriar do conhecimento pertinente à atividade de programação e, desta forma construir o conhecimento e a capacitação necessária para a implementação de soluções computacionais para os problemas que se apresentam no decorrer da atuação dos futuros profissionais de tecnologia da informação.

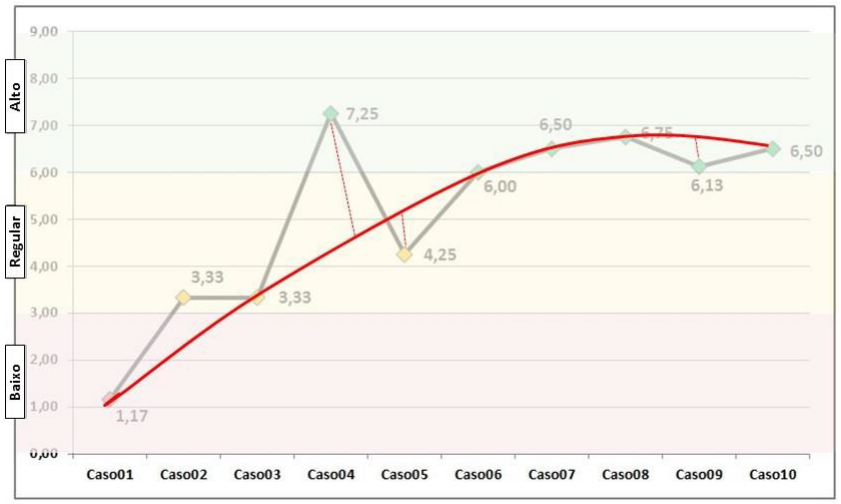
Toda a base estabelecida a partir deste levantamento prévio contribuiu para a definição do modelo a ser adotado nesta Tese, o qual envolveu a utilização de blocos de programação visual, devido à facilidade que ensejam para criar um programa uma vez que possibilitam trabalhar em um processo de selecionar, arrastar/soltar os blocos, compondo assim a solução final. Também envolveu a adoção de mundos virtuais, com os quais foi possível explorar aspectos motivacionais, de imersão e cognitivos comuns neste tipo de aplicação. Culminado com a capacitação para o desenvolvimento da programação com uma linguagem imperativa, neste caso específico a linguagem C, o que não poderia deixar de ser feito uma vez que a aprendizagem desta linguagem é um dos principais objetivos da disciplina de algoritmos e programação.

Com intuito de validar este modelo, diferentes sistemáticas foram organizadas e testadas em disciplinas de algoritmos, a fim de identificar o nível de eficiência de cada uma das abordagens. Ao final de 18 meses foi possível apontar a ordem dos conteúdos, através da qual se obtinha um melhor desempenho dos estudantes, resultado alcançado com o Modelo 2. Este método estava calcado na reorganização do plano de ensino da disciplina, sem alterar a sua carga horária, sendo inserido conteúdos de programação com blocos visuais (usando as linguagens Scratch e Scratch4OS) e programação para mundos virtuais (LSL) e, por fim os assuntos pertinentes à programação com a linguagem C. Além destas mudanças na estrutura dos assuntos abordados, houve um planejamento detalhado das atividades a serem realizadas em cada período, o qual se apoiou nas teorias de aprendizagem selecionadas.

Contudo, embora as práticas deste modelo se demonstrassem efetivas, não havia registros da construção do Pensamento Computacional por parte dos alunos, durante as interações com o sistema de trabalho proposto e também era necessária uma avaliação do desempenho acadêmico dos estudantes participantes do estudo.

Uma das últimas etapas desta tese foi voltada para a investigação e busca de evidências da construção das habilidades relacionadas ao Pensamento Computacional pelos estudantes, durante todas as fases da disciplina. Para isso foi estabelecida uma taxonomia específica com o intuito identificar e elicitar tais competências, com vistas a averiguar se as mesmas foram desenvolvidas. Isto permitiu obter evidências sobre o desenvolvimento do PC pelos estudantes, conforme apresentado na Figura 56, onde tem-se na vertical a identificação dos níveis de Pensamento Computacional, propostos na taxonomia e, na horizontal o conjunto de atividades realizadas pelos discentes ao longo da disciplina. A linha vermelha demonstra a evolução das capacidades evidenciadas nos estudantes, partindo de um nível baixo, não satisfatório, no início do semestre e alcançando um patamar alto nos estudos relacionados à linguagem C.

Figura 56-Evolução das competências relacionadas ao Pensamento Computacional



Fonte: Próprio autor

Por fim, a última atividade deste estudo consistiu na validação das observações sobre o desempenho acadêmico de cada indivíduo, desenvolvida a partir de um conjunto de inferências estatísticas, que agregaram segurança às conclusões apontadas, sobre a predominância e efetividade, como processo de aprendizagem, no Modelo 2.

A efetividade do Modelo 2 pode ser comprovada pelas diferentes ações executadas, permitindo assim concluir que um sistema de ensino, voltado para área de algoritmos e programação, calcado sobre a seguinte sequência específica: lógica de programação, utilização de blocos de programação visual com Scratch e Scratch4OS, interação com o

mundo virtual Open Simulator, migração da programação visual para construção de *scripts* LSL para o metaverso e, por fim, a mudança no paradigma para uma linguagem estruturada e imperativa (Linguagem C) estimulam a construção do raciocínio lógico e das habilidades e competências necessárias para que o aluno iniciante consiga implementar soluções computacionais para problemas reais.

Outro ponto pertinente a ser destacado se refere à experiência obtida pelos participantes do experimento, ao migrar entre diferentes paradigmas de programação, o que possibilitou ao discente reconhecer que a linguagem manipulada é apenas um meio, pois o verdadeiro recurso, a lógica de programação, é a mesma desenvolvida ao longo da disciplina e utilizada na construção de soluções com outras linguagens já estudadas.

Como trabalho futuro, pretende-se implementar um portal para o apoio ao processo de ensino de algoritmos, construído com base nos estudos desenvolvidos nesta pesquisa. O ambiente deverá contar com diferentes recursos, alinhados com as teorias de aprendizagem aqui discutidas e disponibilizará os conteúdos e paradigmas de programação na mesma sequência definida para o Modelo 2.

REFERÊNCIAS

AAMODT, A; PLAZA, E. **Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches**. AI Communications, [S.l.], v.7, n.1, p.39-59, Mar. 1994

ABELSON, H. & DISESSA, A. **Turtle geometry**. Massachussets, Mit Press, 1981.

ABELSON, H.; ABELSON, A. **Logo for Macintosh. An Introduction Through Object Logo**. MIT Press Classics Series, 1993.

ABELSON, H.; FRIEDMAN, M. **App Inventor – A view into learning about computers through building mobile applications**. ACM - Special Interest Group on Computer Science Education, 2010.

AL-IMAMY, S.; ALIZADEH, J.; NOUR, M.A. **On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process**. In: Journal of Information Technology Education 5, 2006.

ALMEIDA, L.; FREIRE, T. **Metodologia da investigação em psicologia e educação.** (2ºed.). Braga: Psiquilíbrios, 2000.

AMARAL, Érico; AVILA, Barbara G.; TAROUCO, Liane MR. **Aspectos teóricos e práticos da implantação de um laboratório virtual no OpenSim**. In:Anais do Simpósio Brasileiro de Informática na Educação. 2012.

ARNÁIZ, Pilar; HERRERO, Ana; DE HARO, Remedios. Trabajo colaborativo entre profesores y atención a la diversidad. Comunidad Educativa, n. 262, p. 29-35, 1999.

AURÉLIO, **Dicionário do Aurélio Online**, 2008. Disponível em: < http://www.dicionariodoaurelio.com/interpretar>. Acesso em: 01 maio 2015.

AUSUBEL, D. P. **Some psychological aspects of the structure of knowledge**. In: ELAM, S. (Ed.) Education and the structure of knowledge. Illinois: Rand MacNally, 1964.

AUSUBEL D. P. ; NOVAK J. D.; HANESIAN H., **Educational Psychology: A Cognitive View**. In New York: Holt, Rinehart& Winston, 1978.

AUSUBEL, D. P. **Aquisição e retenção de conhecimentos: Uma perspectiva cognitiva**. Lisboa: Editora Plátano, 2003.

ÁVILA, Bárbara; AMARAL, Erico; TAROUCO, Liane. **Implementação de Laboratórios Virtuais no metaverso OpenSim**. RENOTE, v. 11, n. 1, 2013.

AYELDEEN, H.; SHAKER, O.; HEGAZY, O.; HASSANIEN, A. E. Case-Based Reasoning: A Knowledge Extraction Tool to Use. In: Information Systems Design and Intelligent Applications. Springer India, 2015.

AZEVEDO, Breno Fabrício Terra; REATEGUI, Eliseo; BEHAR, Patricia Alejandra. **Estudo de análise qualitativa em fórum de discussão**. RENOTE, v. 7, n. 3, p. 135-145, 2009.

AZUMA, R. T. **A Survey of Augmented Reality**, Presence: Teleoperators an Virtual Environments, v.6, n.4, p. 355-385, 1997.

BAEZA-YATES, Ricardo A. **Teaching algorithms**. ACM SIGACT News, v. 26, n. 4, 1995.

BAINBRIDGE, W. S. **The Scientific Research Potential of Virtual Worlds**. Science, vol. 317, n. 5837, p. 472-476, 2007.

BACKER, Paul de. **Gestão ambiental: A administração verde**. Rio de Janeiro: Qualitymark, 1995.

BARAB, S.; SQUIRE, K. **Design-based research: putting a stake in the ground.** The Journal of the Learning Science, v. 13, n. 1, p. 1-14, 2004.

BARANAUSKAS, Maria Cecília Calani. **Procedimento, função, objeto ou lógica? Linguagens de programação vistas pelos seus paradigmas.** Computadores e Conhecimento: Repensando a Educação. Campinas, SP, Gráfica Central da Unicamp, 1993.

BARBAS, Maria; LOPES, Nuno. **Introdução à programação:(re) construção de espaços educacionais em realidade aumentada**. REVISTA DA UIIPS, p. 40, 2013.

BARBETTA, Pedro Alberto; REIS, Marcelo Menezes; BORNIA, Antonio Cezar. **Estatística: para cursos de engenharia e informática**. Atlas, 2004.

BARNES D.J.; FINCHER S.; THOMPSON S. **Introductory problem solving in computer science**. In G. Daughton, and P. Magee, (eds.) 5th Annual Conference on the Teaching of Computing, Dublin City University, 36–39, 1997.

BARCELOS, Ricardo. **O processo de construção do conhecimento de algoritmos com o uso de dispositivos móveis considerando estilos preferenciais de aprendizagem.** Tese de Doutorado, Programa de Pós-Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul, 2012.

BARCELOS, Ricardo; TAROUCO, Liane; BERCHT, Magda. **O uso de mobile learning no ensino de algoritmos**. RENOTE, v. 7, n. 3, p. 327-337, 2009.

BARCELOS, Ricardo. **Ambiente Virtual de Aprendizagem 3D: proposta de Objeto de Aprendizagem para o Ensino de Algoritmos**. Encontro de Educação à Distância, p. 47-61, 2011.

BARCELOS, T.; SILVEIRA, Ismar Frango. **Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica**. In: XX Workshop sobre Educação em Computação, Curitiba. Anais do XXXII CSBC. 2012.

BARKER-PLUMMER, David. (2011) **"Turing Machines", The Stanford Encyclopedia of Philosophy** (Spring of 2011 Edition), Edward N. Zalta (ed.). Disponível em: <http://plato.stanford.edu/archives/spr2011/entries/turing-machine>. Acesso em: 15 setembro 2014.

BATTISTELLA, P. E.; NETO, A. C. R.; CAMPOS, R. L.; INÁCIO, A. S.; JUNIOR, D. I. R.; SILVEIRA, R. A.; VON WANGENHEIM, A. **Classificação de Objetos de Aprendizagem e análise de Ferramentas de Autoria**. XX Simpósio Brasileiro de Informática na Educação, 2009.

BATTISTELLA, Paulo Eduardo; VON WANGENHEIM, Aldo; VON WANGENHEIM, Christiane Gresse. **SORTIA-Um Jogo para Ensino de Algoritmo de Ordenação: Estudo de caso na Disciplina de Estrutura de Dados**. In: Anais do Simpósio Brasileiro de Informática na Educação. 2012.

BARR, David; HARRISON, John; CONERY, Leslie. **Computational Thinking: A Digital Age Skill for Everyone**. Learning & Leading with Technology, v. 38, n. 6, 2011.

BECK BRONDANI, Matheus; MOZZAQUATRO, Patricia Mariotto; ANTONIAZZI, Rodrigo Luiz. **Ambiente de simulação e animação para o ensino de programação**. Revista Interdisciplinar de Ensino, Pesquisa e Extensão, v. 1, n. 1, 2014.

BELL, P., SANDOVAL, W. A. **Design-Based Research Methods for Studying Learning** in Context: Introduction. Educational Psychologist, v. 39, n.4, p. 199–201, 2004.

BEN-ARI, M. **Constructivism in Computer Science Education**, In: Journal of Computers in Mathematics and Science Teaching, v. 20, n. 1, 2001.

BENTLY, J. L.; KERNINGHAN, B. W. **A system for algorithm animation**. Computing Systems. Vol 4. No. 1, 1991.

BERCTH, Magda; DE FRANÇA FERRREIRA, Luís; SILVEIRA, Sidnei Renato. **Aprendendo a construir algoritmos através da mediação digital**. RENOTE, v. 3, n. 1, 2005.

BEREITER, C. and NG., E. **Three Levels of Goal Orientation in Learning.** In Journal of the Learning Sciences, nº 3, (vol. 1), 243-271, 1991.

BERLINSK, D. **O Advento do Algoritmo: A Idéia que Governa o Mundo.** São Paulo: Globo, 2002.

BERVIAN, Pedro Alcino; CERVO, Amado Luiz; SILVA, Roberto da. **Metodologia científica**. São Paulo: Pretence Hall, 2002.

BETT, **Powering Learning. The learning technology event**, London, Jan 2013.

BLOOM, B. S. *et al.* **Taxonomy of educational objectives**. New York: David Mckay, . 262 p. (v. 1), 1956.

BLOOM, B. S. **What we are learning about teaching and learning: a summary of recent research.** Principal, v. 66, n. 2, p. 6-10, 1986.

BORGES, M.A.F. **Uma nova abordagem para o ensino de banco de dados.** Anais do Workshop de Ensino de Informática WEI98/CSBC98, Volume 1, pp.445-453, 1998.

BORGES, M. Augusto F. **Avaliação de uma metodologia alternativa para a aprendizagem de programação**. VIII Workshop de Educação em Computação – WEI. Curitiba, 2000.

BORGES, K. S.; FILHO, H. B. R. **A Importância dos Grupos de Estudos na Formação Acadêmica.** XIII Workshop de Educação em Computação - WEI. São Leopoldo, RS, Brasil, 2005.

BOTELHO, Carlos Alberto. **Sistemas Tutores no domínio da programação**. Revista de Informática Aplicada/Journal of Applied Computing, v. 4, n. 1, 2010.

BOX, Joan Fisher. **Guinness, Gosset, Fisher, and small samples**. Statistical Science, p. 45-52, 1987.

BRAGA, Mariluci. **Realidade virtual e educação**. Revista de biologia e ciências da terra, v. 1, n. 1, p. 1-13, 2001.

BRIEGER, Friedrisch Gustav. **Limites unilaterais e bilaterais na análise estatística.** Bragantia, v. 6, n. 10, 1946.

BROWN, A. **Design experiments: theoretical and methodological challenges in creating complex interventions in classroom settings**. The Journal of the Learning Science, v.2, n.2, p.141-178, 1992.

BRUSILOVSKY, P.. **Program visualization as a debugging tool for novices**. In: Proc. of INTERCHI'93 .Pp. 29-30. Amsterdam, 1994.

BYRNE, P. and LYONS, G. **The Effect of Student Attributes on Success in Programming**. In Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE 2001, United Kingdom, p.49-52, 2001.

CAMARGO, Sandro C.; CECHINEL, Cristian; VASCONCELLOS, Bruno C. **Desenvolvimento de Um Aplicativo Integrado ao Facebook e ao Sistema do Projeto IGUAL para Compartilhamento e Recomendação de Objetos de Aprendizagem.** XVIII TISE – Conferência Internacional sobre Informática na Educação, Porto Alegre, Brasil, 2013.

CAMPOS, Geraldo Maia. **Estatística prática para docentes e pós-graduandos.** São Paulo: Faculdade de Odontologia de Ribeirão Preto, 2001.

CAMPOS, E. A. V; ASCENCIO, A. F. G. **Fundamentos da Programação de Computadores**. Editora Prentice Hall, 2003.

CAMPOS, R. L. B. L. **ERMC2: Uma proposta de metodologia para melhoria do ensino-aprendizado de lógica de programação**. In: XI Congresso Chileno de Educación Superior en Computación (CCESC), 2009b. Santiago, Chile, Jornadas Chilenas de Computación. 2009.

CASTRO, C. T.; CASTRO Junior, A.; MENESES, C., B. M.; RAUBER, M. **Utilizando Programação Funcional em Disciplinas Introdutórias de Computação**, In: XI Workshop de Educação em Computação – WEI, Campinas/SP, 2003.

CASTRO, Thaís; FUCKS, Hugo. **Sistematização da Aprendizagem de Programação em Grupo. Instituto de Computação** – Universidade Federal do Amazonas (UFAM), Programa de Pós-Graduação em Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2011. Disponível em <http://groupware.les.inf.puc-rio.br/public/papers /SistematizacaoGPL.pdf>. Acesso em 22 junho 2014.

CHASE, Jared A.; HOUMANFAR, Ramona. **The differential effects of elaborate feedback and basic feedback on student performance in a modified, personalized system of instruction course**. Journal of Behavioral Education, v. 18, n. 3, 2009.

CHAVES, José O.; CASTRO, Angélica; LIMA, Rommel; LIMA, Marcos V.; FERREIRA, Karl. **Mojo: uma ferramenta para auxiliar o Professor em disciplinas de programação**. XXI Workshop sobre Educação em Computação**,** XIII Congresso Brasileiro da Sociedade Brasileira de Computação, Maceió/AL, 2013.

CERVO, Arnaldo L.; BERVIAN, Pedro A.; DA SILVA, Roberto. **Metodologia científica**. Editora Pearson Prentice Hall, 6ª Ed. 2007.

CHOI, J. e HANNAFIN, M. **Situated Cognition and Learning Environments: Roles, Structures and Implications for Design**. Educational Technology Research and Development, 43(2), pp 53-69, 1995.

CHICKERING, Arthur W. **Applying the Seven Principles for Good Practice in Under graduate Education**. New directions for teaching and learning, v. 47, 1991.

COLLINS, A., JOSEPH, D., & BIELACZYC, K. **Design research: Theoretical and methodological issues.** Journal of the Learning Sciences, v.13, n.1, p.15–42, 2004.

CONWAY, M., J.; PAUSCH, R. "**Alice: Easy to Learn Interactive 3D Graphics**", Computer & Graphics, Volume 31, Issue 3, pages 58- 59, 1997.

COOPER, D. R.; SCHINDLER, P. S. **Métodos de pesquisa em administração**, 7ª ed. Porto Alegre: Bookman, 2003

CORREIA, N.; CHAMBEL, T. "**Integração Multimédia em Meios e Ambientes Aumentados nos Contextos Educativos e Culturais**", In: Revista Interdisciplinar dos Centros e Núcleos da Unicamp, Campinas, 2004.

CORREIA, Carlos H; DOMINGUES, Maria J. C. de S. **Práticas inovadoras de ensino: uso de brinquedos no ensino de algoritmos**. XIII SIMPEP, Bauru, SP, 2006.

COSTELLOE, E**. The Use of a Software Enabled Scaffolding Environment to Aid Novice Programers**. Submited to the University of Dublin for the Degree of Master of Science, 2004.

CRAIG, S. D.; GHOLSON, B.; DRISCOLL, D. M.. Animated Pedagogical Agents in Multimedia Educational Environments. Journal of Educational Psychology, Vol. 94, No. 2, 428-434, 2002.

CREWS T.; ZIEGLER U. **The Flowchart Interpreter for Introductory Programming Courses.** Department of Computer Science Western Kentuchy University Bowling Green, 1998.

CRISTÓVÃO, H. M. **Aprendizagem de Algoritmos num Contexto Significativo e Motivador: Um Relato de Experiência**. In: Anais do XXVII Congresso da Sociedade Brasileira de Computação. Belém do Pará: PA, 2008.

DAMIANI, Magda F. **Entendendo o trabalho colaborativo em educação e revelando seus benefícios.** Understanding collaborative work in education and revealling its benefits. Educar em Revista, n. 31, 2008.

DANTSIN, E.;EITER, T.;GOTTLOB, G.; VORONKOV, A. **Complexity and Expressive Power of Logic Programming**. In Computational Complexity. Proceedings., Twelfth Annual IEEE Conference on (Formerly: Structure in Complexity Theory Conference), 1997.

DAZZI, R. L. S.; SANTIAGO, Rafael de ; JESUS, Elieser Ademir de. **Construtor e Interpretador de Fluxogramas - Uma Ferramenta de Ensino. In: Construtor e Interpretador de Fluxogramas - Uma Ferramenta de Ensino**, 2004, Caceres-Espanha. VI Simposio Internacional de Informática Educativa (SIIE 2004), 2004.

DBRC **(Design-Based Research Collective).** Design-based research: An emerging paradigm for educational inquiry. Educational Researcher, v. 32, n. 1, p. 5-8, 2003.

DCN. **Diretrizes Curriculares Nacionais**, Ministério da Educação (MEC), Brasil. Disponível em: < http://portal.mec.gov.br/index.php?option=com_content&id=12991:diretrizes-curriculares-cursos-de-graduacao>. Acesso em: 05 junho 2014.

DCN-C. **Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação**, Ministério da Educação (MEC), Brasil. Disponível em: < http://portal.mec.gov.br/index.php?option=com_docman&task=doc_download&gid=11205&Itemid=>. Acesso em: 05 junho 2014.

DCN-EC. **Diretrizes Curriculares Nacionais para os Cursos de Graduação em Engenharia de Computação**, Ministério da Educação (MEC), Brasil. Disponível em: < http://portal.mec.gov.br/cne/arquivos/pdf/2008/pces153_08.pdf>. Acesso em: 05 junho 2014.

DCN-Tec. **Diretrizes Curriculares Nacionais para para a Educação Profissional de Nível Tecnológico**, Ministério da Educação (MEC), Brasil. Disponível em: < http://portal.mec.gov.br/cne/arquivos/pdf/cp29.pdf>. Acesso em: 05 junho 2014.

DE ARAÚJO, Fabrício Viero; FALKEMBACH, Gilse Antoninha Morgental. **Experiências no Aprendizado de Algoritmos Utilizando um Ambiente de Aprendizagem**. RENOTE, v. 4, n. 1, 2006.

DEDE, Chris. **EcoMUVE. Advancing Ecosystems Science Education via Situated Collaborative Learning in Multi-User Virtual Environments**. Disponível em: http://ecomuve.gse.harvard.edu/. Acesso em: 20 de outubro 2013.

DE JESUS, Andreia; BRITO, Gláucia Silva. **Concepção de ensino-aprendizagem de algoritmos e programação de computadores: a prática docente**. VARIA SCIENTIA, v. 9, n. 16, 2009.

DELGADO, C.; XEXEO, J. A.; SOUZA, I. F.; CAMPOS, M.; RAPKIEWICZ, C. E. **Uma abordagem pedagógica para a iniciação ao estudo de algoritmos.** In: XII Workshop de Educação em Computação. 2004.

DELGADO, C.; XEXEO, J. A.; SOUZA, I. F.; RAPKIEWICZ, C. E.; PEREIRA Jr, J. **Identificando competências associadas ao aprendizado de leitura e construção de algoritmos.** In: XIII Workshop sobre Educação em Computação. 2005.

DE OLIVEIRA, Anselmo E.; SOARES, Márlon HFB; DE SOUZA, Aparecido R. **Comparação entre média geométrica e médias ponderadas no cálculo de notas em disciplinas conjugadas de química**. Educ. quím, v. 17, n. 4, 2006.

DE BARROS, L. N.; ANA, Delgado, K. V.; MATSUMOTO, P. M. **A tool for programming learning with pedagogical patterns.** In eclipse ’05: Proceedings of the OOPSLA workshop on Eclipse technology eXchange, New York, NY, USA. ACM, 2005.

DETERS, J. I., SILVA, J. D., MIRANDA, E. M., FERNANDES, A. M. R. O **Desafio de Trabalhar com Alunos Repetentes na Disciplina de Algoritmos e Programação.** In: Workshop de Ambientes de apoio à Aprendizagem de Algoritmos e Programação. 2008.

DIJKSTRA, Edsger W. **A Discipline of Programming**. Prentice Hall, 1976.

DIJKSTRA, Edsger W. **On the teaching of programming, i.e. on the teaching of thinking.** In: Selected Writings on Computing: A Personal Perpective. Springer-Verlag, NY. 1982

DIJKSTRA, Edsger W. **On the Cruelty of Really Teaching Computing Science**. In Communications of ACM, Issue 12, (vol.32), 1398-1404, 1989.

DIM, C. A.; ROCHA, F. D. **Uma Ferramenta Para Aprendizagem de Lógicas e Estímulo do Raciocínio e da Habilidade de Resolução de Problemas em um Contexto Computacional no Ensino Médio**. In: XIX Workshop sobre Educação em Computação. Anais do XXI CSBC–2011. Natal, RN. 2011.

DOS SANTOS, Sandra Carvalho. **O processo de ensino-aprendizagem e a relação professor-aluno: Aplicação dos "sete princípios para a boa prática na educação de Ensino superior"**. Caderno de pesquisas em administração, v. 8, n. 1, p. 69-82, 2001.

DOS SANTOS, Rodrigo P.; COSTA, Heitor A. **Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dado e Programação aos iniciantes em Computação e Informática.** Infocoomp, Journal of Computer Sciense, v. 5, n. 1, 2006.

DOUFOUR J. M.; GARDIOL, L.; KHALAF, L. **Simulation-basead Finite Sample Normality Tests in Linear Regressions**. Econometrics Journal, vol1, 1998.

DUARTE, José B. **Estudos de caso em educação. Investigação em profundidade com recursos reduzidos e outro modo de generalização**. Revista Lusófona de Educação, 2008.

DUVAL, R. **Registros de representações semióticas e funcionamento cognitivo da compreensão em matemática**. In: Machado, S. D. A. (org.). Aprendizagem em matemática: Registros de representação semiótica. Editora Papirus, 2003.

EDELSON, D.C. **Design research: what we learn when we engage in design**. The Journal of the Learning Science, v. 11, n. 1, p. 105-121, 2002.

EDUSCRATCH. Computação Criativa uma introdução ao Pensamento Computacional baseada no conceito de *design*. MIT, 2011. Disponível em: http://scratched.gse.harvard.edu/resources/scratch-curriculum-guide-draft. Acesso em: 10 agosto 2014.

EDUSUMMIT. **International Summit on ICT un Education,** Wrking Group 6 Advancing computational thinking in 21ST century learning**,** 2013. Disponível em: http://www.edusummit.nl/theme-edusummit-2013/programme/. Acesso em: 10 setembro 2014.

ELLIS, S.; WHALEN, S. F. **Cooperative Learning: getting strated**. Scholastic, New York, 1990.

ESTEVES, M.; MENDES, A.J. **A Simulation Tool to Help Learning of Object Oriented Programming Basics,** in Proceeding of 34th Frontiers in Education Conference, Savannah, Estados Unidos, Outubro de 2004.

ESTEVES, M.; ANTUNES, R.; MORGADO, L.; MARTINS, P.; FONSECA, B. **Contextualização da aprendizagem da programação: estudo exploratório no Second Life.** Proceedings of IADIS Ibero-Americana WWW/Internet, p. 7-8, 2007.

ESTRELA, T.; ESTEVE, M.; RODRIGUES, A. **Síntese da investigação sobre formação inicial de professores em Portugal**. Porto: Porto Editora, INAFOP, Caderno de Formação de Professores, 2002.

FRANÇA, RS de; SILVA, WC da; AMARAL, HJC do. **Ensino de ciência da computaçao na educaçao básica: Experiências, desafios e possibilidades**. In: XX Workshop sobre Educação em Computação. 2012.

FROZZA, Rejane; SILVA, Andréa K.; LUX, Beatriz; CRUZ, Marcia E. J. K.; BORIN, Mirceia. **Proposta de uma Aplicação de Mundos Virtuais na Educação usando o Open Simulator com diferentes requisitos tecnológicos**. In: Simpósio Brasileiro de Informática na Educação. n 20, 2009, Florianópolis. Anais, Florianópolis, 2009.

HEARST, M. **Untangling Text Mining**. In: ANNUAL MEETING OF THE ASSOCIATION OF COMPUTATIONAL LINGUISTICS, 37th, 1999. University of Maryland, MD, Association of Computational Linguistics, 1999.

HEINZEN, LUIZ ANGELO. **Módulo de raciocínio baseado em casos em uma ferramenta de apoio ao ensino de lógica de programação**. Blumenau: FURB, 2002.

FAGIN, B.; MERKLE, L. D.; EGGERS, T. W. **Teaching computer science with robotics using Ada/Mindstorms 2.0.** In Proceedings of the 2001, Annual ACM SIGAda International Conference on Ada (pp. 73-78). New York: ACM Press, 2001.

FAL, M.; CAGILTAYC, Nergiz E. **How scratch programming may enrich engineering education**. In: Proceedings of the 2nd International Engineering Education Conference. Mourtos, NJ. 2013.

FALKEMBACH, G. A. M. **Uma experiência de resolução de problemas através da estratégia ascendente: Ambiente de Aprendizagem Adaptado para Algoritmos (A4).** Tese de Doutorado. Porto Alegre: PGIE/UFRGS, 2003.

FALKEMBACH, G. A. M., AMORETTI, M. S. M., TAROUCO, L. R., & VIERO, F. **Aprendizagem de Algoritmos: Uso da Estratégia Ascendente de Resolução de Problemas**. 8º Taller Internacional de Software Educativo. Santiago, Chile, 2003.

FALKEMBACH, G. A. M. ; ARAÚJO, Fabrício Viero de . **Ensino de Algorítmos utilizando a estratégia ascendente de resolução de problemas com o apoio do ambiente A4**. In: VII Congreso Iberoamericano de Informática Educativa, 2004, México, 2004.

FALKEMBACH, Gilse; VIERO DE ARAÚJO, Fabrício. **Aprendizagem de Algoritmos: Dificuldades na Resolução de Problemas.** Anais SULCOMP, v. 2, n. 2, 2013.

FARIA, Eustáquio SJ; ADÁN COELLO, Juan M. **Detectando diferenças significativas entre programas como auxílio ao aprendizado colaborativo de programação**. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO, 2004.

FARIA, E. Marques; LIMA, V. Cirino. **Aprendizagem de Algoritmos – Dificuldades e soluções**. IV Encontro Estadual de Didática e Prática de Ensino (EDIPE), Pontifícia Universidade Católica de Goiás, 2011.

FARIA, E. Marques. **A contribuição da teoria histórico-cultural de vygotsky para o ensino ea aprendizagem de algoritmo**. Tese de Doutorado, Programa de Pós-Graduação Stricto Sensu em Educação, Pontifícia Universidade Católica de Goiás, 2013. Disponível em:

http://tede.biblioteca.ucg.br/tde_arquivos/23/TDE-2013-10-16T151523Z-1447/Publico/ELIEZER%20MARQUES%20FARIA.pdf. Acesso em: 11 Setembro 2014.

FARIAS, M. A. Freitas; JÚNIORII, Gilson P. S.; ANDRADE, Elisângela M. A. O. R. **Um Estudo Preliminar Sobre as Dificuldades no Processo de Ensino e Aprendizagem das Disciplinas Básicas de Programação no Ifs-Campus Lagarto**. VI Colóquio Internacional "Educação e Contemporaneidade", São Cristovão, SE, 2012.

FELDMAN, R.; SANGER, J. **The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.** Cambridge, MA: Cambridge University Press, 2007.

FERRANDIN, M. & STEPHANI, S. **Ferramenta para o ensino de programação via Internet**. I Congresso Sul Catarinense de Computação: UNESC. Criciúma, 2005.

FERRAZ, A. P. C. M.; BELHOT, Renato Vairo. **Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais**. Gest. Prod., São Carlos, v. 17, n. 2, p. 421-431, 2010.

FERREIRA, Cláudia; GONZAGA, Flávio; SANTOS, Rodrigo. **Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração.** In: Anais do XVIII Workshop sobre Educação em Computação, XXX CSBC, Belo Horizonte, MG, Brasil. 2010.

FERREIRA, Cláudia; GONZAGA, Flávio; SANTOS, Rodrigo. **Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração.** In: Anais do XVIII Workshop sobre Educação em Computação, XXX CSBC, Belo Horizonte, MG, Brasil. p. 981-990, 2010.

FERREIRA, FANI S. S.; BALDUINO, JEFFERSON O. **Uma proposta de resolução de problemas aplicada ao ensino de introdução à lógica e algoritmos de programação.** II CONINTER – Congresso Internacional Interdisciplinar em Sociais e Humanidades, Belo Horizonte, MG, 2013.

FILHO, Cléuzio F. (2007) **História da computação: O Caminho do Pensamento e da Tecnologia**. Porto Alegre: EDIPUCRS, 2007. Disponível em: <http://www.pucrs.br/edipucrs/online/historiadacomputacao.pdf>. Acesso em: 15 Setembro 2014.

FINKE, M.; HOMMEL, G.; SCHEFFER, T.; WYSOTZKI, F. **Aerial robotics in computer science education**. Computer Science Education, 7(2), 239-246, 1996.

FISCHER, M.; BAIRD, D. Making **M-learning work: Utilizing Mobile Technology for active exploration, collaboration, assessment, and reflection in higher education**. Journal of Educational Technology systems, 35 (1), 3-30. 2007.

FLEURY, A.; FLEURY, M. T. L. **Estratégias empresariais e formação de competências: um quebra-cabeça caleidoscópio da indústria brasileira.** 2. ed. São Paulo, Atlas, 2001.

FRANCO, J. F.; DA CRUZ, S. R. R.; FRANCO, N. F.; DEUS LOPES, R. **Apresentando uma Arquitetura Pedagógica e Técnica Usada em Sinergia com Recursos Multimídia na Construção Cooperativa de Saberes**. RENOTE, v. 4, n. 1, 2006.

FUKS, Hugo. **Sistemas Colaborativos**. Ed. Elsevier, São Paulo, Brasil, 2012.

GALVEIAS, Maria de Fátima Cid. **Prática Pedagógica: Cenário de Formação Profissional**. Journal Interacções, N. 8, PP. 6-17, ISSN 1646-2335, 2008.

GARCIA, Islene C.; REZENDE, Pedro J. de; CALHEIROS, Felipe C. **Astral: um ambiente para ensino de estruturas de dados através de animações de algoritmos**. Revista Brasileira de Informática na Educação, v. 1, n. 1, p. 71-80, 1997.

GARDNER, H. **Frames of mind.** New York, Basic Books Inc., 1985.

GARDNER, H. **Frames of Mind: The theory of multiple intelligences**. New York: Basic Book, 1993.

GERHARD W., BRUSILOVSKY P. **Elm-art: An adaptive versatile system for web based instruction**. International Journal of Artificial Intelligence in Education, 12:351–384, 2001.

GIRAFFA, Lucia Maria Martins; VICCARI, Rosa Maria. **Estratégias de Ensino em Sistemas Tutores Inteligentes modelados através da tecnologia de agentes**. Revista Brasileira de Informática na Educação, v. 5, n. 1, p. 9-18, 1999.

GONDIM, H. W.; AMBROSIO, A. P. L.; COSTA, Fábio Moreira. **Uma Experiência no Ensino de Algoritmos utilizando Ambientes Visuais de Programação 3D.** In: Workshop sobre Educação em Computação (WEI), Bento Gonçalves-RS. 2009.

GOMES, A. J.; MENDES, A. J. **Suporte à aprendizagem da programação com o ambiente SICAS**, in RIBIE, 2000.

GOMES, A., HENRIQUES, J., MENDES, A. J. "**Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores**". "In Educação, Formação & Tecnologias; vol.1(1), pp. 93-103, 2008. Disponível em: < http://eft.educom.pt> Acesso em: 15 maio 2014.

GOMES, Tancicleide; DE MELO, Jeane CB. **App Inventor for Android: Uma Nova Possibilidade para o Ensino de Lógica de Programação**. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2013.

GONDIM, H. W. A. S.; AMBRÓSIO, Ana Paula. **Esboço de fluxogramas no ensino de algoritmos**. In: WEI-Workshop sobre Educação em Computação. 2008.

GONZÁLEZ, Sahudy M.; TAMARIZ, Annabell R. **Uma Experiencia no Ensino de Programação para Cursos de Engenharia**, XXVIII Encontro Nacional de Engenharia de Produção, Rio de Janeiro, 2008

GOUVEIA, L., "**Ambientes Virtuais Colaborativos: a procura de formas alternativas de interacção**", Revista Politécnica, nº2, 2000.

GREGOLIN, Vanderlei R. **Conceitos Matemáticos em Ambiente Logo.** Dissertação de Mestrado, Programa de Pós-Graduação em Educação, Universidade Federal de São Carlos. 1994.

GRÜBEL, M. J; BEZ. F. **Jogos educativos.** Revista RENOTE Novas Tecnologias na Educação, v. 4 n. 2, 2006.Disponível em: <http://seer.ufrgs.br/renote/article/view/14270>. Acesso em: 13 Setembro 2013.

HAAJANEN, J.; PESONIUS, M.; SUTINEN, E.; TARHIO, J.; TERASVIRTA, T.; VANNINEN, P. **Animation of user algorithms on the Web**. Proc. VL'97, IEEE Symposium on Visual Languages, 356-363.EUA, 1997.

HARASIM, L.; CALVERT, T.; GROENEBER, C. **Virtual-U: a Web-Based System to Support Collaborative Learning**. In B. H. KHAN (Ed.)*Web-Based Instruction*. Englewood Cliffs, N.J.:Educational Technology Publications, 1997.

HARRIS, J. **CS1: Where is Visual Basic? Consortium for Computing Sciences in Colleges**. Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference, EUA, 2000.

HAYS (2014), **Recruting Experts Worldwide**. Disponível em: <http://www.hays.pt/.>. Acesso em: 10 Setembro 2014.

HEARST M.A. **The Debate on Automated Essay Grading**, IEEE Intelligent Systems, vol. 15, nº. 5, p. 22-37. 2000.

HERREID, Clyde Freeman; SCHILLER, Nancy A. **Case studies and the flipped classroom**. Journal of College Science Teaching, v. 42, n. 5, p. 62-66, 2013.

HOSTINS, Higor; RAABE, André. **Auxiliando a aprendizagem de algoritmos com a ferramenta webportugol**. XV WEI, 2007.

HUNDHAUSEN C. et al., "**Integrating pedagogical code review into a CS1 course: An empirical study**," in Proc. 40th ACM Tech. Symp. Comp. Sci. Educ., Chattanooga, pp. 291–295, TN, 2003.

IDC Latin. IDC Analize the Future, 2014. Disponível em: <http://br.idclatin.com/prodserv/mktanalysis.aspx>. Acesso em: 10 Setembro 2014.

IDEONE. **Ideone powered by Sphere Engine**. Disponível em: < http://ideone.com/>. Acesso em: 25 setembro 2013.

IEPSEN, Edécio Fernando; BERCHT, Magda; REATEGUI, Eliseo Berni. **Avaliando a Dimensão Afetiva para Apoio ao Processo de Aprendizagem na Disciplina de Algoritmos: um Estudo de Caso**. Revista Latinoamericana de Tecnología Educativa-RELATEC, v. 12, n. 2, p. 55-66, 2013.

IEPSEN, Edécio Fernando. **Ensino de algoritmos: detecção do estado afetivo de frustração para apoio ao processo de aprendizagem**. Tese de Doutorado/Programa de Pós Graduação em Informática na Educação – UFRGS, 2013a.

IEPSEN, Edécio Fernando; BERCHT, Magda; REATEGUI, Eliseo. **Detecção e Tratamento do Estado Afetivo Frustração do Aluno na Disciplina de Algoritmos**. In: Anais do Simpósio Brasileiro de Informática na Educação. 2011.

IFF, **Projeto Pedagógico do Curso de Licenciatura em Computação, Instituto Federal de Educação Ciência e Tecnologia Farroupilha**, Campus Santo Augusto. Disponível em: < http://www.sa.iffarroupilha.edu.br/site/midias/arquivos/20122116105189ppc_lic_computacao_2012.pdf>. Acesso em 14 junho 2014.

IFSE, **Projeto Pedagógico do Curso de Sistemas de Informação, Instituto Federal de Educação, Ciência e Tecnologia do Sergipe**. Disponível em: < http://www.ifs.edu.br/images/DAA/ppc/bacharelado%20sistemas%20da%20informacao%20-%20lagarto.pdf>. Acesso em: 15 Junho 2014.

IFTO, **Projeto Pedagógico do Curso de Licenciatura em Computação, Instituto Federal de Educação Ciência e Tecnologia do Tocantins**. Disponível em: < http://www.porto.ifto.edu.br/noticias/computacao.pdf >. Acesso em: 14 junho 2014.

IGUAL Project. **Innovation for Equality in Latin American Universities**. Disponível em: < http://www.igualproject.org/>. Acesso em: 20 junho 2014.

JENKINS, T. **On the difficulty of learning to program.** In Proceedings of 3rd Annual LTSN_ICS Conference (Loughborough University, United Kingdom, August 27-29, 2002). The Higher Education Academy, p.53-58, 2002.

JERMANN, P.; SOLLER, A.; MUEHLENBROCK, M. **From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning**. In: Proc. of Euro-CSCL, 2001.

JONASSEN, D. H. **Evaluating Constructivistic Learning in Duffy**, T. M. & Jonassen, D. H. Construtivism and the Technology of Instruction - A Conversation. LEA Publishers. 1992.

JONASSEN D. H. **The Future of Hypermedia Based Learning Environments: Problems, Prospects, and Entailments**. Proceedings of the EDMedia. 1993.

JOHNSON, D. W.; JOHNSON, R. T. **Learning together and alone, cooperative, competitive and individualistic learning.** Allyn and Bacon, Paramount, 1994.

JOHNSON, W. L.; SOLOWAY, E. **PROUST: Knowledge-based program understanding**. In: IEEE, Transactions on Software Engineering, IEEE, 1985.

JÚNIOR, J. C. R. P.; RAPKIEWICZ, C. E. **O Processo de Ensino e Aprendizagem de Algoritmos e Programação: Uma Visão Crítica da Literatura**. III Workshop de Educação em Computação e Informática do estado de Minas Gerais (WEIMIG). Belo Horizonte, 2004.

KACHIGAN, Sam Kash. **Statistical analysis: An interdisciplinary introduction to univariate & multivariate methods**. New York, NY, 1986.

KAMEI, Fernando Kenji; DE MELO FERREIRA, Fernanda Josirene; DA COSTA FERRO, Márcio Robério. **Vídeo-Monitoria: aumentando o desempenho dos alunos de programação.** 10ª ERBASE – Escola Regional de Computação, Bahia-Alagoas-Sergipe, Maceió, Brasil, 2010.

KAMPFF, Adriana Justin Cerveira; DIAS, Márcia Gladis Cantelli. **Reflexões sobre a construção do conhecimento em ambientes de pesquisa e de autoria multimídia: uma tarefa compartilhada por alunos e professores**. RENOTE, v. 1, n. 2, 2003.

KAMPSTRA, Peter. **Beanplot: A boxplot alternative for visual comparison of distributions**. Journal of Statistical Software, v. 28, n. 1, 2008.

KELLEHER, C.; PAUSCH, R. **Lowering the Barriers to Programming: A Taxonomy of Programming Environments and languages for Novice Programmers**. In ACM Computing Surveys, v. 37, n. 2, 83-137, 2005.

KELLY, A. **Design research in education: yes, but is it methodological?** The Journal of the Learning Science, v.13, n.1, p.115-128, 2004.

KELLY, A. **Is Design-based research a method on its own?** In Convenção Internacional AERA, George Manson University, 2006.

KNUTH, D. E. **The Art of Computer Programming**. Vol 1 – Fundamental Algorithms. Addison-Wesley, 1968.

KOLB, D. A. **Experential learning: experience as the source of learning and development.** EUA: Prentice-Hall, 1984.

KOLB, D. A. **A gestão e o processo de aprendizagem**. In: STARKEY, K. Como as organizações aprendem. São Paulo: Futura, . p. 321-341, 1997.

KOLODNER, Janet. **Case-Based Reasoning**. San Mateo: Morgan Kaufmann, 1993.

KOORSSE, Melisa; CILLIERS, Charmain; CALITZ, André. **Programming assistance tools to support the learning of IT programming in South African secondary schools**. Computers & Education, v. 82, p. 162-178, 2015.

KOSTER, R. **Theory of fun for game design. Scottsdale: Paraglyph**, p. 80-99, 2004.

KOZMA, R. **Reflections on the state of educational technology research and development.** ETD&R, v. 48, n.1, p.5-15, 2000.

KOWALSKI, R. **Predicate Logicas a Programming Language**. In Information Processing ‘74, pp. 569-574. North-Holland,Amsterdam, 1974.

KURNIA, A.; LIM, A.; CHEANG, B. **Online Judge. Computer & Education**, v. 36, n.4, p. 299-315, 2011.

LAHTINEN, E.; ALA-MUTKA, K.; JÄRVINEN, H.-M. **A Study of the Difficulties of Novice Programmers**. Proceedings do 10th ITiCSE, Monte de Caparica, Portugal, 2005.

LAKATOS, Eva M.; MARCONI, Marina de A. **Metodologia do Trabalho científico**. 7. ed. São Paulo: Atlas, 2007.

LATTA, J. N.; OBERG, D. J. **A conceptual virtual reality model**. IEEE Computer Graphics & Applications, pp. 23-29, 1994.

LEAKE, David B. **Case-Based Reasoning: experiences, lessons, and future directions**. 420 fl. Menlo Park: AAAI Press / MIT Press, 1996.

LEAVE, J. & WENGER, E. **Situated learning: legitimate peripherial participation**. Cambridge, UK: Cambridge University Press, 1991.

LEC. **LSL Editor Community Edition**. Disponível em: <http://sourceforge.net/projects/lsleditor/postdownload?source=dlp> Acesso em: 05 julho 2014.

LEECH, Nancy L.; BARRETT, Karen Caplovitz; MORGAN, George Arthur. **SPSS for intermediate statistics: Use and interpretation**. Psychology Press, 2005.

LEVENE, Howard. **Robust tests for equality of variances**. Contributions to probability and statistics: Essays in honor of Harold Hotelling, v. 2, p. 278-292, 1960.

LÉVY, P. Cibercultura. Rio de Janeiro: Editora 34, 1999.

LILLIEFORS, Hubert W. **On the Kolmogorov-Smirnov test for normality with mean and variance unknown.** Journal of the American Statistical Association, v. 62, n. 318, 1967.

LIM, Tjen-Sien; LOH, Wei-Yin. **A comparison of tests of equality of variances.** Computational Statistics & Data Analysis, v. 22, n. 3, p. 287-301, 1996.

LIMA, Mark Renato Campos. **Algoritmos Genéticos na Formação de Grupos para Aprendizagem Cooperativa Apoiada por Computador**. PPGEE/UFMA, 2006.

LINDER, S. P.; NESTRICK, B. E.; MULDERS, S.; LAVALLE, C. L. **Facilitating Active learning with inexpensive Móbile Robots**. The journal of Computing in small colleges. Anais do 6º CCSC. Northeastern conference on the journal of computing in small coleges, Vol. 16, 2001.

LINDERMANN, Helena. **Os sistemas do futuro e seus aspectos ergonômicos**. In: Boletim Dc. Porto Alegre vol. 4, n. 1, 1983.

LINO, A.D.P; HARB, M.P.A.; BRITO, S.R.; SILVA, A.S.; FAVERO, E. **Avaliação automática de consultas SQL em ambiente virtual de ensino-aprendizagem**. 2ª Conferência Ibérica de Sistemas e Tecnologias de Informação. Porto, Portugal, 2007.

LISTER, R.; ADAMS, E. S.; FITZGERALD, S.; FONE, W.; HAMER, J.; LINDHOLM, M.; THOMAS, L. **A multi-national study of reading and tracing skills in novice programmers**, In: ACM SIGCSE Bulletin, v. 36, n. 4, 2004.

LKG, **Lifelong Kindergarten Group**. Disponível em: <http://llk.media.mit.edu/>. Acessado em: 14 setembro 2014.

MA, Linxiao; FERGUSON, John; ROPER, Marc; ROSS, Isla; WOOD, Murray. **Improving the mental models held by novice programmers using cognitive conflict and jeliot visualisations.** ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Pages 166-170, New York, NY, USA 2009.

MACEDO, A. A.; PIMENTEL M. da G.; FORTES R. P. de M. **Studyconf: infraestrutura de suporte ao aprendizado cooperativo na www**. Revista Brasileira de Informática na Educação, nº5, 77-99, Setembro, 1999.

MACHADO, Alex F. **Uma metodologia de aprendizagem baseada em classificador numérico aplicada a jogos eletrônicos**. Tese de Doutorado, Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, 2012. Disponível em: < http://www2.ic.uff.br/PosGraduacao/Teses/550.pdf>. Acessado em: 10 abril 2013.

MACHADO, Rosele F. S.; RAABE, André L. A. . **Modelagem Cognitiva dos Problemas de Aprendizagem de Algoritmos**. Universidade do Vale do Itajaí. Disponível em:
< http://200.169.53.89/download/CD%20congressos/2008/SBIE/sbie_posters/Modelagem%20cognitiva%20dos%20problemas%20de%20aprendizagem.pdf > Acesso em: 20 abril 2014.

MAIA, L. D. O.; DA SILVA, V. J.; ROSA, R. E. D. S.; DE LUCENA JUNIOR, V. F.; QUEIROZ NETO, J. P. **A Robótica como Ambiente de Programação Utilizando o Kit Lego Mindstorms**. Simpósio Brasileiro de Informática na Educação, 2008.

MALAN, David J.; LEITNER, Henry H. **Scratch for budding computer scientists.** In: ACM SIGCSE Bulletin. ACM, 2007.

MANN, H. B.; WHITNEY, D. R. **On a test whether one of two random variables is stochastically larger than the other**. Ann. Math. Statist., 18, 1947.

MANSO, António; OLIVEIRA, Luís; MARQUES, C. **Portugol IDE–Uma ferramenta para o ensino de programação**. 2009. Disponível em: < http://orion.ipt.pt/~manso/papers/2009/Portugol_IDE_PAEE2009.pdf >. Acesso em: 21 junho 2014.

MARÔCO, João. **Análise estatística com o SPSS Statistics.** Report Number, Lda, 2011.

MASSEY JR, Frank J. **The Kolmogorov-Smirnov test for goodness of fit.** Journal of the American statistical Association, v. 46, n. 253, 1951.

MATTAR, F. N. **Pesquisa de Marketing – execução, análise.** 2ª ed. São Paulo: Atlas, 1998.

MATTOS, Mauro M.; WANGENHEIM, C. G. V.; PACHECO, Roberto C. **Aplicação de Raciocínio Baseado em Casos na Fase de Análise de Requisitos para Construção de Abstrações em Lógica de Programação**. SEMINCO–Seminário de Computação, Blumenau, 1999.

MARQUES, Diego L.; COSTA, Luís F. S.; SILVA, Max A. A.; REBOUÇAS, Ayla D. D. S. **Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução a Programação utilizando Jogos e Python**, XVII WIE – Workshop de Informática na Escola, XXII SBIE - Simpósio Brasileiro de Informática na Educação, UFS, Aracaju, 2011.

MARTINEZ, Luís; FERREIRA, Aristides. **Análise de Dados com SPSS.** Escolar Editora, 2007.

MATTOS, Mauro M.; VAHLDICK, Adilson. **Relato de uma experiência no ensino de algoritmos e programaçao utilizando um framework lúdico**. In: Anais do II Workshop de Ambientes de apoio à Aprendizagem de Algoritmos e Programação. 2008.

MAYER, Richard. **Multimedia Learning**. Cambridge: Cambridge University Press. 2001.

MAYER, Richard. **Cognitive Constraints on Multimedia Learning: When Presenting More Material Results in Less Understanding**. Journal of Educational Psychology. Vol. 93, Nº 1, 187-198. 2001b.

MAYER, R.; MORENO, R. **Animation as on Aid to Multimedia Learning.** Educational Psychology Review, Vol 14, No 1, March 2002.

MEC. "**Diretrizes Gerais do Programa de Apoio a Planos de Reestruturação e Expansão das Universidades Federais – REUNI**", Ministério da Educação, 2007. Disponível em: < http://portal.mec.gov.br/sesu/arquivos/pdf/diretrizesreuni.pdf>. Acesso em: 18 janeiro 2014.

MEDINA M.; FERTIG C. **Algoritmos e Programação: Teoria e Prática** São Paulo: Novatec, 2005.

MEERBAUM-SALANT, Orni; ARMONI, Michal; BEN-ARI, Mordechai. **Learning computer science concepts with scratch.** Computer Science Education, v. 23, n. 3, p. 239-264, 2013.

MELCHIORS, C.; TAROUCO, L. M. R. **Raciocínio baseado em casos aplicado ao gerenciamento de falhas em redes de computadores**. Tese de Doutorado. UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, 1999.

MÉLO, F., CUNHA, R., SCOLARO, D.; CAMPOS, J. **Do Scratch ao Arduino: Uma proposta para o ensino introdutório de programação para cursos superiores de tecnologia.** In: XXXIX Congresso Brasileiro de Educação em Engenharia, Blumenau, SC. Disponível em:< www. abenge. org. br/CobengeAnteriores/2011/sessoestec/art1886. pdf>. Acesso em: 21 junho 2014.

MENDES, Thiago G. **Games e Educação: Diretrizes de Projetos para Jogos Digitais voltados à Aprendizagem.** Dissertação de Mestrado. Programa de Pós-Graduação em *Design*, Faculdade de Engenharia, UFRGS, 2012. Disponível em: < http://www.lume.ufrgs.br/bitstream/handle/10183/61009/000860539.pdf?sequence=1>. Acesso em: 01 junho 2014.

MENEZES, C. S.; TAVARES, O.L.; NEVADO, R.A.; CURY, D. **Computer Supported Co-operative Systems to support the problem solving - a case study of learning computer programming**. In: The 38th Annual Frontiers in Education (FIE) Conference, 2008, New York. v. 1, 2008.

MICHAELIS. **Michaelis: moderno dicionário da língua portuguesa**, Editora Melhoramentos, 2009. Disponível em: < http://michaelis.uol.com.br/moderno/portugues>. Acesso em: 02 maio 2015.

MILLER, G. A. **The magical number seven, plus or minus two: Some limits on our capacity for processing information**. Psychological Review 63 (2): 81–97.doi:10.1037/h0043158. PMID 13310704, 1956.

MILONE, Giuseppe. **Estatística: geral e aplicada.** Pioneira Thomson Learning, 2004.

MIRANDA, Fabio Neves. "**Facebook e Algoritmos: Um Estudo de Caso do Uso do Facebook como Suporte Pedagógico na Disciplina de Algoritmos e Técnicas de Programação do Curso de Sistemas de Informação em IES.** Disponível em: < http://www.faminasbh.edu.br/upload/downloads/20130627151026_83686.pdf>. Acesso em: 20 junho 2014.

MIT, **Scratch**, Disponível em: <http://info.scratch.mit.edu/Scratch_1.4>. Acesso em: 10 setembro 2014.

MONDSHEIN, Lee; SATTAR, Abdul; LORENZEN, Torben. **Visualizing Prolog: a jigsaw puzzle approach**. ACM Inroads, v. 1, n. 4, p. 43-48, 2010.

MONTGOMERY, D. C.; RUNGER, G. C. **Estatística aplicada e probabilidade para engenheiros**. 4ª ed. Rio de Janeiro: LTC, 2009.

MORAN, José M. **Novas tecnologias e mediação pedagógica**. 6. ed. Campinas: Papirus, 2000.

MORAN. José M. **Novos desafios na educação: a Internet na educação presencial e virtual**. In: Porto, Tania M. E. (Org.). Saberes e linguagens de educação e comunicação. Pelotas: Ed. Universitária, 2001.

MORAN, José Manuel. **Os novos espaços de atuação do educador com as tecnologias. Conhecimento local e conhecimento universal: diversidade, mídias e tecnologias na educação**. Curitiba: Champagnat, p. 245-253, 2004.

MORAIS, C. M. **Escalas de medida, estatística descritiva e inferência estatística.** Sebenta de Estatística, Instituto Politécnico de Bragança, 2005.

MOREIRA, Mireille Pinheiro; FAVERO, Eloi Luiz. **Um Ambiente Para Ensino de Programação com *Feedback* Automático de Exercícios**. In: Workshop sobre Educação em Computação (WEI 2009). 2009.

MOREIRA, Marco Antônio. **Teorias de Aprendizagem**. São Paulo: EPU, 1999.

MORELATO, L.A.; BORGES, M.A.F. **Alternativas de Baixo Custo para o uso da Robótica Educacional: Construção e Avaliação do Framework GoGo Board**. Simpósio Brasileiro de Informática na Educação, 2008.

MOTA, M. P.; BRITO, S. R.; MOREIRA, M. P.; FAVERO, E. L. "**Ambiente Integrado à Plataforma Moodle para Apoio ao Desenvolvimento das Habilidades Iniciais de Programação**" In: Anais do XX Simpósio Brasileiro de Informática na Educação, 2009.

MOREIRA M. A. **A teoria da aprendizagem significativa e sua implementação em sala de aula**. Brasília: Editora da UnB, 2006.

MOREIRA, Mireille Pinheiro; FAVERO, Eloi Luiz. **Um Ambiente Para Ensino de Programação com *Feedback* Automático de Exercícios.** In: Workshop Sobre Educação em Computação. 2009.

MOODLE. **Open-source learning platform.** Disponível em: < https://moodle.org/>. Acesso em: 28 setembro 2013.

MOTIL, J.; EPSTEIN (2000), D. **JJ: a Language Designed for Beginners (Less Is More).** Disponível em: http://www.ecs.csun.edu/jmotil/TeachingWithJJ.pdf. Acessado em: 15 Set 2014.

MÜLLER, T. J.; DO CANTO FILHO, A. B.; AMARAL, É. M. H.; DE LIMA, J. V.; TAROUCO, L. M. R. **Classificação de Objetos de Aprendizagem Segundo o Grau de Multimodalidade**. RENOTE, 11(1), 2013.

MURTEIRA, B. J. F. **Estatística descritiva-análise exploratória de dados.** Ed. McGraw-Hill, 1993.

NASCIMENTO, Mariana R.; GUERRERO, Dalton D. S.; FIGUEIREDO, Jorge C. A. **Um Estudo sobre a Eficácia do Ensino à Distância de Programação para Alunos Iniciantes**. Dissertação de Mestrado, Centro de Engenharia Elétrica e Informática, UFCG, 2011.

NAACE (2014) **Computing in the National Curriculum, A Guide for Primary Teachers.** COMPUTING AT SCHOOL, EDUCATE-ENGAGE-ENCOURAGE . Disponível em: <http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf>. Acesso em: 15 setembro 2014.

NETO, Valter dos Santos Mendonça. **A utilização da ferramenta scratch como auxílio na aprendizagem de lógica de programação**. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2013.

NETS. **National Educational Technology Standards for Students**, Second Edition. NETS Project, International Society For Technology In Education (ISTE), 2007.

NIKOU, Stavros A.; ECONOMIDES, Anastasios A. **Transition in Student Motivation during a Scratch and an App Inventor course**. In: Global Engineering Education Conference (EDUCON), 2014 IEEE. IEEE, 2014.

NMC. **New Media Consortium, Technology Outlook for UK Tertiary Education 2011-2016**. Disponível em: < http://www.nmc.org/pdf/2011-Technology-Outlook-UK.pdf>. Acesso em: 25 junho 2014.

NOBRE, Isaura A. M.; MENEZES, Crediné Silva de. **Suporte à Cooperação em um Ambiente de aprendizagem para Programação (SAmbA).** In: Anais do Simpósio Brasileiro de Informática na Educação. 2002.

NUNES, Marcelo; GIRAFFA, Lúcia. **A educação na ecologia digital**. PPGCC/FACIN, PUCRS, 2003.

NUNES, F. B.; AMARAL, É. M. H.; VOSS, G. B.; MEDINA, R. D.; HERPICH, F.; TAROUCO, L. M. **Integrating Virtual Worlds and Virtual Learning Environments through Sloodle: from theory to practice in a case of study for teaching of algorithms**. XVIII TISE – Conferência Internacional sobre Informática na Educação, Porto Alegre, Brasil, 2013.

OCHOA, X.; SUPERIOR, E.; CARRILLO, G.; ORTEGA, A.; VILLAVICENCIO, C. **Large-scale storage and retrieval of educational metadata using an rdf store**. D-Lib Magazine, 2012.

OEIRAS, J.; VAHAL, J.; NETO, M. ROCHA, H. **Modalidades Síncronas de Comunicação e Elementos de Percepção em Ambientes de EaD**. Simpósio Brasileiro de Informática na Educação. São Leopoldo: 2002.

OPENSIM. **OpenSimulator Project.** Disponível em: < http://opensimulator.org/>. Ac sso em: 25 setembro 2013.

OZORAN, D.; CAGILTAY, N.E.; TOPALLI, D. **Using Scratch in introduction to programming Course for Engineering Students**, MEUK2012, Antalya, Turkey, 2012.

PAPERT, S. **Mindstorms: Children, Computers, and Powerful Ideas**. Basic Books, New York, 1980.

PAPERT, S. **LOGO: Computadores e Educação**. São Paulo, Brasiliense, 1985.

PAPERT, Seymour. **An exploration in the space of mathematics educations.International Journal of Computers for Mathematical Learning**, v. 1, n. 1, 1996.

PEREIRA, Jean-Louis do Canto. **Implementação de OpenSimulator com funcionalidade de voz em ambiente empresarial**, 2010.

PEREIRA JÚNIOR, J.; RAPKIEWICZ, C.E.; DELGADO, C.; XEXEO, J.A.M. **Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio**. XIII WEI - Workshop de Educação em Computação. São Leopoldo, RS, Brasil, 2005.

PERKINS, D. N., SCHWARTZ, S. and SIMMONS, R. **Instructional Strategies for the Problems of Novice Programmers**. In R. E. Mayer (ed.), Teaching and Learning Computer Programming, p.153-178. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

PETRY, Patrícia G. **Um sistema para o ensino e aprendizagem de algoritmos utilizando um companheiro de aprendizagem colaborativo**. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, UFSC, 2005.

PFANNKUCH, Maxine. **Comparing box plot distributions: A teacher's reasoning**. Statistics Education Research Journal, v. 5, n. 2, 2006.

PIAGET, J. **Development and Learning. Journal of Research in Science Teaching**, New York, n. 2, v. 3, p. 176-86, 1964.

PIAGET, J. Psicologia e Pedagogia. Rio de Janeiro: Forense Universitária, 1982. [Psychologie et Pédagogie, 1969]

PÍCCOLO, Homero L.; SILVA, Marcus O.; RODRIGUES, Vinícius A.; MAIA, Yuri A. N. **Um Ambiente Interativo de Aprendizagem para o ensino de programação na Universidade de Brasília.**

PIMENTEL, K.; TEIXEIRA, K. **Virtual Reality – through the new looking glass**. 2.ed. New York, McGraw-Hill, 1995.

PIMENTEL Edson; FRANÇA, Vilma; NORONHA, Robinson; OMAR, Nizam. **Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores**. IX Workshop de Informática na Escola, 2003. Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/viewFile/819/805> Acesso em: 07 abril 2014.

PK-12, Leveraging Thought Leadership for Computational Thinking in the PK-12 Curriculum. National Science Foudation (NSF), 2009. Disponível em: <http://www.nsf.gov/awardsearch/showAward?AWD_ID=0964217>. Acesso em: 15 Julho 2015.

POZZEBON, E.; MACHADO, G.; MINATTO, S.G.; JUSTO IZÉ M. da C.; Frigo, L.B. **Programação de Computadores no Ensino Médio**, International Conference ICBL – Interactive Computer Aided Blended Learning, 2013.

PROULX, V.K. **Programming Patterns and Design Patterns in the Introductory Computer Science Course**, proc. of the 31st SIGCSE, Austin, TX, USA, march, p. 7-12, 2000.

QUEIROZ, M. I. P. de. **Relatos orais: do indizível ao dizível**. In: VON SIMSON, O. (Org.). Experimentos com histórias de vida (Itália-Brasil). São Paulo: Vértice; Editora Revista dos Tribunais. Enciclopédia Aberta de Ciências Sociais, v. 5, 1988.

RAABE, André Luís Alice; GIRAFFA, Lúcia Maria Martins; ORTH, Afonso Inácio. **Ambiente para Produção de Material Didático baseado na utilização de Vídeos e Internet.** In: Congresso Latino-americano de Informática - CLEI99, Assunción-Paraguay, setembro, 1999.

RAABE, André Luís Alice, **Uma proposta de arquitetura de Sistema Tutor Inteligente baseada Teoria das Experiências de Aprendizagens Medianas** – Tese de Doutorado, 2005.

RAMOS, Paula; GIANNELLA, T. Rabetti , STRUCHINER, Miriam. **A Pesquisa Baseada em Design em Artigos Científicos sobre o Uso de Ambientes de Aprendizagem Mediados pelas Tecnologias da Informação e da Comunicação no Ensino de Ciencias: Uma Análise Preliminar**. VII Encontro Nacional de Pesquisa em Educação em Ciências – Enpec, ISSN 21766940, Florianópolis, 2009.

RAMOS, P. **Ambiente virtual Vivências: análise do processo de desenvolvimento na perspectiva da pesquisa baseada em design**. 240 f. 2010. Tese de Doutorado. Tese (Doutorado em Educação em Ciências e Saúde)–Núcleo de Tecnologia Educacional para a Saúde, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

RAPKIEWICZ, C. Elena; FALKEMBACH, Gilse. A. M. SEIXAS, L. M. Jeanty ; SANTOS, N. S. R. Santana; CUNHA, V. Vieira; KLEMANN, Miriam. **Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais**. Revista RENOTE Novas Tecnologias na Educação, v. 4 n. 6 2006.

RAZALI, Nornadiah Mohd; WAH, Yap Bee. **Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests**. Journal of Statistical Modeling and Analytics, v. 2, n. 1, p. 21-33, 2011.

REATEGUI, E.; CAMPBELL, J. A.; TORRES, R. **Using Item Descriptors in Recommender Systems**. The Nineteenth National Conference on Artificial Intelligence (AAAI 2004), Workshop on Semantic Web Personalization, San Jose, USA, Julho 2004.

REATEGUI, Eliseo; NOTARE, Marcia. **A3-Ambiente de Aprendizagem de Algoritmos.** In: Anais do Simpósio Brasileiro de Informática na Educação. 2004.

REATEGUI, Eliseo B.; CERON, R. F.; BOFF, E.; VICARI, R. M. **Um agente animado para ambientes de aprendizagem colaborativos**. Revista Brasileira de Informática na Educação, v. 14, n. 3, 2006.

REATEGUI, E; EPSTEIN, D; LORENZATTI, A.; KLEMANN, M.. **Sobek: a Text Mining Tool for Educational Applications.** In: International Conference on Data Mining, 2011, Las Vegas, Estados Unidos. Anais do DMIN '11, 2011

RESNICK, Mitchel; MALONEY, John; ANDRÉS, Monroy-Hernández ; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay;SILVERMAN, Brian; KAFAI, Yasmin. **Scratch: programming for all**. Communications of the ACM, v. 52, n. 11, p. 60-67, 2009.

RESNICK, R.; ROSEMBAUN, E. **Design for thinkerability.** Design, Make, Play: Growing the next generation of STEM Innovatiors, 163-181. Routledge, 2013.

REDONDO, M., BRAVO, C., ORTEGA, M., "**Relacionando los espacios de discusión y elaboración en un entorno para el aprendizaje en grupo del diseño**", Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, No.20, 2001.

REBELO, B.; MENDES, A.; MARCELINO, M.; REDONDO, M. **Sistema colaborativo de suporte à aprendizagem em grupo da programação—SICAS-COL**. Proceedings of the VII Simpósio Internacional de Informática Educativa, Leiria, Portugal, 2005.

REBOUÇAS, Ayla D. D. S.; MARQUES, Diego L.; COSTA, Luís F. S.; SILVA, Max A. A. **Aprendendo a Ensinar Programação Combinando Jogos e Python.** XXI SBIE - Simpósio Brasileiro de Informática na Educação, UFPB, João Pessoa, 2010.

REICHERT, R.; NIEVERGELT, J.; HARTMANN, W. **Programming in Schools - Why, and How**. In: Pellegrini, C., Jacquesson, A. (eds.) Enseigner l'informatique, pp. 143-152. Georg Editeur Verlag, 2001.

REIS, E. **Estatística Descritiva.** Lisboa: Edições Sílabo, 1996.

RIBEIRO, Romenig S.; BRANDÃO, Leônidas O.; BRANDÃO, Anarosa A. F. **Uma visãodo cenário Nacional do Ensino de Algoritmos e Programação: uma proposta baseada no Paradigma de Programação Visual**. XVIII WIE – Workshop de Informática na Escola, XXIII SBIE - Simpósio Brasileiro de Informática na Educação, UFRJ, Rio de Janeiro, 2012.

ROCHA, P. S., FERREIRA, B., MONTEIRO, D., NUNES, D. D. S. C., & do Nascimento Góes, H. C. **Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino**. RENOTE, 8, 2010.

ROCHA, Rogério. **Utilização da robótica pedagógica no processo de ensino-aprendizagem de programação de computadores**. Dissertação de mestrado. CEFET, MG, 2006.

RODRIGUES, P. A., BRANDÃO, L. O., BRANDÃO, A. A. F., "**Interactive Assignment: a Moodle component to enrich the learning process**". In: Proceedings of Frontiers in Education, páginas T4F-1 - T4F-6, 2010.

ROGERS, J. **Communities of Practice: A framework for fostering coherence** in virtual learning communities. *Educational Technology & Society.* 3 (3), 384-392, 2000.

ROMANELLI, G. **A entrevista antropológica: troca e alteridade.** Revista do Programa de Pós-Graduação em Psicologia da Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Ribeirão Preto, 1998.

ROMEIKE, R. **Applying creativity in CS high school education - criteria, teaching example and evaluation.** In Proc. Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland. CRPIT, 88. Lister, R. and Simon, Eds. ACS. 87-96, 2007.

ROSA, N.; RAPKIEWICZ, C**. Ensinando princípios básicos de programação utilizando jogos educativos em um programa de inclusão digital**. VI *Brazilian Symposium on Computer Games and Digital Entertainment*. São Leopoldo, 2007.

RUIZ, Alvaro J. **Metodologia Científica, Um guia para eficiência nos estudos**, Editora Atlas, 6ª Ed. 2010.

RUST, C., PRICE, M.; O'DONOVAN, B. "**Improving students' learning by developing their understanding of assessment criteria and processes**", In: Assessment and Evaluation in Higher Education, 28(2), 147-164, 2003.

ROYSTON, Patrick. **Approximating the Shapiro-Wilk W-Test for non-normality**. Statistics and Computing, v. 2, n. 3, p. 117-119, 1992.

SADLER, D.R**. Formative assessment: revisiting the territory**. In: Assessment in Education, 5(1), 77-84, 1998.

SAJANIEMI, J.; KUITTINEN, M. **Program Animation Based On The Roles Of Variables**". Proceedings of The 2003 ACM Symposium on Software Visualization, 2003.

SALGADO, Nilmara; CASTRO, Thaís; CASTRO, Alberto. **Aprendizagem Colaborativa de Programação com Scratch e OpenSimulator**. In:Proceedings of Brazilian Symposium on Collaborative Systems. Sociedade Brasileira de Computação, 2013.

SANTOS, R. P., COSTA, H. A., RESENDE, A. M., SOUZA, J. M. . **O uso de ambientes gráficos para ensino e aprendizagem de estruturas de dados e de algoritmos em grafos**. In: Anais do XXVIII Congresso da SBC-WEI. 2008. p. 157-166.

SANTOS, Rodrigo Pereira; COSTA, Heitor Augustus Xavier. **TBC-AED e TBC-AED/Web: Um desafio no ensino de algoritmos, estruturas de dados e programação.** In: IV Workshop Em Educação Em Computação E Informática Do Estado De Minas Gerais. 2005.

SANTOS, Rodrigo Pereira; COSTA, Heitor Augustus Xavier. **Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática**. Infocoomp, Journal of Computer Sciense, v. 5, n. 1, 2006.

SANTOS, Leila M. A.; TAROUCO, Liane M. R. **A importância do estudo da teoria da carga cognitiva em uma educação tecnológica**. RENOTE, v. 5, n. 1, 2007.

SANTOS, R. P. dos; COSTA, H. A. X. **Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos Iniciantes em Computação e Informática**. In: INFOCOMP, Volume 5, nº.1, ISSN 1807-4545, 2006.

SANTOS, Elci Alcione; FERMÉ, Eduardo; FERNANDES, Elsa. **Utilização de robots no ensino de programação: o projecto droide**. In: Actas do IX Congresso da SPCE "Educação para o sucesso: políticas e actores, 2007.

SBC, **Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática, Sociedade Brasileira de Computação.** Disponível em: < http://bcc.ime.usp.br/principal/documentos/sbc2003.pdf>. Acesso em: 15 Junho 2014.

SCAICO, Pasqueline D.; MARQUES, Diego L.; AZEVÊDO, Max A.; SILVA, Jarbele C.; NETO, Sinval V. M. **Combinando Diversão e Educação: Castelo dos Enigmas, um Jogo Sério para o Ensino de Algoritmos** XVII WIE – Workshop de Informática na Escola, XXII SBIE - Simpósio Brasileiro de Informática na Educação, UFS, Aracaju, 2011.

SCAICO, P. D.; DE LIMA, A. A.; DA SILVA, J. B. B.; AZEVEDO, S.; PAIVA, L. F.; RAPOSO, E. H. S.; MENDES, J. P. **Programação no Ensino Médio: Uma Abordagem de Ensino Orientado ao *Design* com Scratch**. In Anais do Workshop de Informática na Escola (Vol. 1, No. 1), 2012.

SCHLEMMER, E. TREIN, D. **Criação de Identidades Digitais Virtuais para Interação em Mundos Digitais Virtuais em 3D**. Congresso Internacional de EaD – ABED, 2008.

SCHMITT, Marcelo A. R.; TAROUCO, Liane M. R. **Metaversos e laboratórios virtuais – possibilidades e dificuldades**. Revista de Novas Tecnologias na Educação, Porto Alegre, v. 6, n. 1, p 1-12, 2008.

SCHULTZ, Brian B. **Levene's test for relative variation.** Systematic Biology, v. 34, n. 4, p. 449-456, 1985.

SCANLAN, A. D. **Structured Flowcharts Outperform Pseudocode: An Experimental Comparisons**. California States University at Sacramento, 1989.

SCRATCH. **About Scratch (Scratch Documentation Site)**. Disponível em: <http://info.scratch.mit.edu/ About_Scratch>. Acesso em: 18 julho 2014.

SENAY, Hikmet; LAZZERI, Santos Gerardo. **Graphical representation of logic programs and their behaviour.** In: Visual Languages, 1991., Proceedings. 1991 IEEE Workshop on. IEEE, 1991.

SETTI, Mariangela de Oliveira Gomes**. O Processo de Discretização do Raciocínio Matemático na Tradução para o Raciocínio Computacional: Um Estudo de Caso no Ensino/Aprendizagem de Algoritmos.** Tese de Doutorado. Tese) Universidade Federal do Paraná. Curitiba, Brasil, 2009.

SETUBAL, J. C. **Uma proposta de Plano Pedagógico para a Matéria de Computação e Algoritmos**, In: II Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática (WEI'2000). Editora Universitária Champagnat, 2000.

SHANG, Y., SHI, H., CHEN, S. **An intelligent distributed environment for active learning**. In: ACM Journal of Educational Resources in Computing (JERIC). Vol. 1, No. 2, Summer 2001, Article 4, 17 pages. ISSN:1531-4278. New York: ACM Press, 2001.

SHAPIRO, S. S.; WILK, M. B. **An Analysis of Variance Test for Normality (Complete Samples)**. Biometrika, 1965.

SIEGEL, S.; CASTELLAN, N. J. **Estatística não-paramétrica para ciências do comportamento.** 2nd. ed. Porto Alegre: Artmed, 2006.

SILVA, I. F. A.; SILVA, I. M. M.; SANTOS, M. S. (2013). **Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação**. XIII Jornada de Ensino, Pesquisa e Extensão, Universidade Federal Rural de Pernambuco, Recife, 2013.

SILVA, Ítalo F. A. ; SILVA, Ivanda M. M.; SANTOS, Marizete S.. **Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação**. Universidade Fed. Rural de Pernambuco, Recife – PE. 03 pp. Disponível em: < http://www.eventosufrpe.com.br/jepex2009/cd/resumos/R1479-1.pdf>. Acesso em: 18 abril 2014.

SILVESTRE, António. **Análise de dados e estatística descritiva**. Escolar editora, 2007.

SINAES. **Sistema Nacional de Avaliação da Educação Superior, Ministério da Educação**, Ministério da Educação (MEC) Brasil. Disponível em: < http://portal.mec.gov.br/index.php/?id=12303&option=com_content>. Acesso em: 05 junho 2014.

SIROTHEAU, Silvério; BRITO, Silvana Rossy; SILVA, Aleksandra do Socorro; ELIASQUEVICI, Marianne Kogut; FAVERO, Marianne Kogut; TAVARES, Orivaldo de Lira. **Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação.** XVII WIE – Workshop de Informática na Escola, XXII SBIE - Simpósio Brasileiro de Informática na Educação, UFS, Aracaju, 2011.

SLEEMAN, D. **The challenges of teaching computer programming**, In Communication. ACM Volume 29, Issue 9, pages 840–841, 1986.

SLOANE, K. D.; LINN, M. C. **Instructional Conditions in Pascal Programming Classes**. In R. E. Mayer (ed.), Teaching and Learning Computer Programming, p.207-235. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

SLOODLE. Simulation-Linked object Oriented Dynamic Learning Environment Project. Disponível em: < https://www.sloodle.org/>. Acesso em: 25 setembro 2013.

SOFTEX (2009), **Softwares e Serviços de TI – A Industria Brasileira em Perspectiva**, Observatório SOFTEX, N.1, Vol 1. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/2009-Observatorio-Softex-Industria-em-perspectiva-1.pdf>. Acesso em: 10 setembro 2014.

SOLOWAY, E.; EHRLICH, K. **Empirical Studies of Programming Knowledge**. In: IEEE Transactions on Software Engineering, vol. SE-10, n. 5, setembro, 1984.

SOUTO, Aletéia Vanessa M.; DUDUCHI, Marcelo. **Um processo de avaliação baseado em ferramenta computadorizada para o apoio ao ensino de programação de computadores**. Disponível em: <http://bibliotecadigital.sbc.org.br/download.php?paper=1363>. Acesso em: 01 abril 2011.

SOUZA, Marcelo; JAEGER, Eliana V.; CARDOSO Brigiane M. S**. Ensino de algoritmos apoiado pelo uso de jogos digitais educativos.** Revista RENOTE Novas Tecnologias na Educação, v. 11 n. 3, 2013.

STRUCHINER, M. **Apreciação analítica de ambientes construtivistas de aprendizagem baseados em novas tecnologias de informação e de comunicação para a educação na Área das Ciências da Saúde.** Projeto submetido ao CNPq, 2006.

SUKKARIEH, Jana Z.; PULMAN, Stephen G.; RAIKES, Nicholas. **Auto-Marking - Using Computational Linguistics to Score Short, Free Text Responses**. International Alliance for Learning Conference. 2003.

SUO, Xiaoyuan. **Toward more effective strategies in teaching programming for novice students.** Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on, ISBN 978-1-4673-2417-5, Pages T2A-1 - T2A-3 Hong Kong, 2012.

SWELLER, John; MERRIENBOER, Jeroen; PAAS, Fred. **Cognitive Load Theory and Instructional Design**. Educational Psychology Review, Vol. 10, No. 3, 1998

SWELLER, John. "**Cognitive Load Theory: A Special Issue of educational Psychologist**". LEA, Inc, 2003.

TAKAHASHI, T. **Introdução a Programação Orientada aos Objectos**. III EBAI (Escola Brasileiro-Argentina de Informática), 1998.

TAN, A. **Text Mining: The State of the Art and the Challenges**. In: Workshop on Knowledge Discovery from Advanced Databases (PKDAD'99), 1999, Beijing. Proceedings. Beijing, 1999.

TANRIKULU, Elif; SCHAEFER, Björn Christian. **The users who touched the ceiling of scratch**. Procedia-Social and Behavioral Sciences, v. 28, 2011.

TAROUCO, L. M. R.; CUNHA, S. **Aplicação de teorias cognitivas ao projeto de objetos de aprendizagem**. Revista RENOTE Novas Tecnologias na Educação, v. 4 n. 2, Dezembro, 2006.

TAROUCO, L. M. R.; DOS SANTOS, P. M. E.; ÁVILA, B.; GRANDO, A. R.; SOUZA ABREU, C. **Multimídia Interativa: princípios e ferramentas**. RENOTE, v. 7, n. 1, 2009.

TAROUCO, Liane. **Alfabetização visual para a redução da sobrecarga cognitiva em material educativo digital.** Ambientes hipermidiáticos, p. 37-48, 2006.

TAROUCO, L. M. R., KONRATH, M. L. P., CARVALHO, M. J. S., AVILA, B. G. **Formação de professores para produção e uso de objetos de aprendizagem**. RENOTE, v. 4, n. 1, 2006.

TAVARES, Leonardo G.; CARSTENS, Luciano; DESCHAMPS, Ana C. F.; FANCHIN, Bruna; FERREIRA, Eduardo B.; CARMO, Jéssica S. **A Importância das Disciplinas de Algoritmos e Programação no Desenvolvimento dos Trabalhos de Conclusão de Curso**

na Engenharia Elétrica da Universidade Positivo. XL COBENGE – Congresso Brasileiro de Educação em Engenharia, Belém, 2012.

TODOROV, J. C.;TRISTÃO, G., "Sistema personalizado de ensino: bases psicológicas e abordagem administrativa" Cadernos de Psicologia Aplicada, 3, 65-71. São Paulo, 1975.

TODOROV, J. C., MOREIRA, M. B., MARTONE, R. C., "**Sistema Personalizado de Ensino, Educação a Distância e Aprendizagem Centrada no Aluno**". Psicologia: Teoria e Pesquisa. São Paulo/SP, 2009.

TORI, Romero; MAINENTE, A. C. **Aprendendo lógica de programação via Web**. 2011. Disponível em http://e-spacio.uned.es/fez/eserv.php?pid=bibliuned:1121&dsID=n09tori01.pdf>. Acesso em 23 Jun 2014.

UCB, **Projeto Pedagógico do Curso de Bacharelado em Ciência da Computação, Universidade Católica de Brasilia.** Disponível em: < http://www.ucb.br/sites/000/3/00001486.pdf>. Acesso em: 14 junho 2014.

UFAL, **Projeto Pedagógico do Curso de Ciência da Computação, Universidade Federal de Alagoas.** Disponível em: < http://www.ufal.edu.br/arapiraca/graduacao/ciencia-da-computacao/projetos-pedagogicos/projeto-pedagogico-2009>. Acesso em: 14 junho 2014.

UFAM, **Projeto Pedagógico do Curso de Engenharia da Computação, Universidade Federal do Amazonas.** Disponível em: < http://ft.ufam.edu.br/attachments/article/112/PPC-ENGENHARIA-COMPUTACAO.pdf>. Acesso em: 14 junho 2014.

UFC, **Projeto Pedagógico do Curso de Engenharia de Software, Universidade Federal do Ceará, Campus Quixadá.** Disponível em: < http://www.si3.ufc.br/sigaa/public/curso/ppp.jsf?lc=pt_BR&id=657515>. Acesso em: 15 Junho 2014.

UFES, **Projeto Pedagógico do Curso de Engenharia de Computação, Universidade Federal do Espirito Santo.** Disponível em: < http://www.ceunes.ufes.br/downloads/PP/15.pdf>. Acesso em: 14 junho 2014.

UFG, **Projeto Pedagógico do Curso de Engenharia de Computação, Universidade Federal de Goias.** Disponível em: < http://www2.emc.ufg.br/uploads/467/original_03._EC_Inicio_-_Projeto_Pol%C3%ADtico_Pedag%C3%B3gico_-_Curso_EC.pdf>. Acesso em: 14 unho 2014.

UFG-1, **Projeto Pedagógico do Curso de Engenharia de Software, Universidade Federal de Goias.** Disponível em: < http://www.inf.ufg.br/sites/default/files/projeto-pedagogico-es.pdf>. Acesso em: 15 junho 2014.

UFPA, **Projeto Pedagógico do Curso de Engenharia da Computação, Universidade Federal do Pará.** Disponível em: < http://www.laps.ufpa.br/aldebaro/engcomp/novoppc/Engcomp_projeto_pedagogico_2009_v7_20dez.pdf>. Acesso em: 14 junho 2014.

UFPB, **Projeto Pedagógico do Curso de Licenciatura em Computação EAD, Universidade Federal da Paraíba.** Disponível em: < http://portal.virtual.ufpb.br/wordpress/wp-content/uploads/2013/01/PPC-Licenciatura-Computa%C3%A7%C3%A3o-UFPB-Versao-Final-06082012.pdf>. Acesso em: 14 junho 2014.

UFPI, **Projeto Pedagógico do Curso de Sistemas de Informação, Universidade Federal do Piauí.** Disponível em: < http://www.ufpi.br/subsiteFiles/cc/arquivos/files/sistemas_infor_cshnb.pdf>. Acesso em: 14 junho 2014.

UFRA, **Projeto Pedagógico do Curso de Sistemas de Informação, Universidade Federal Rural da Amazonia.** Disponível em: < http://www.portal.ufra.edu.br/attachments/2038_06%20Projeto%20Pedago%CC%81gico%20de%20Sistemas%20de%20Informac%CC%A7a%CC%83o%20(Versa%CC%83o%202013).pdf>. Acesso em: 15 junho 2014.

UFRN, **Projeto Pedagógico do Curso de Engenharia de Software, Universidade Federal do Rio Grande do Norte.** Disponível em: < http://www.sigaa.ufrn.br/sigaa/verProducao?idProducao=297853&key=b35017b90c922029cf9f6ad62c0e5984>. Acesso em: 15 junho 2014.

UFSC, **Projeto Pedagógico do Curso de Engenharia de Computação, Universidade Federal de Santa Catarina.** Disponível em: < http://enc.ufsc.br/files/2012/04/PPC-Engenharia-de-Computa%C3%A7%C3%A3o.pdf>. Acesso em: 14 junho 2014.

UFSJ, **Projeto Pedagógico do Curso de Ciência da Computação, Universidade Federal de São João Del-Rei.** Disponível em: < http://www.ufsj.edu.br/portal2-repositorio/File/soces/Res050CONEP2011PPCCienciasComutacao_Anexo.pdf>. Acesso em 14 junho 2014.

UFSM, **Projeto Pedagógico do Curso de Ciência da Computação, Universidade Federal de Santa Maria.** Disponível em: < http://www.inf.ufsm.br/index/graduacao/cc/estrutura-do-curso>. Acesso em: 14 junho 2014.

UFVJM, **Projeto Pedagógico do Curso de Sistemas de Informação, Universidade Federal de Viçosa.** Disponível em: < http://cpd.crp.ufv.br/SisPaginaUFV/files/file77603SIP-PPC-Sistemas_de_Informacao-site.pdf>. Acesso em: 15 junho 2014.

UFVJM, **Projeto Pedagógico do Curso de Sistemas de Informação, Universidade Federal dos Vales do Jequitinhonha e Mucuri.** Disponível em: < http://www.ufvjm.edu.br/prograd/regulamento-dos-cursos/doc_view/431-sistemas-de-informacao-projeto-pedagogico.html>. Acesso em: 15 junho 2014.

UK. **Statutory Guidance, National Curriculum in England: computing programmes of study**. Published 11 Seteptember 2013. Disponível em: <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study>. Acesso em: 10 setembro 2014.

ULUDAG, S.; KARAKUS, M.; TURNER, S.W. **Implementation IT0/CS0 with Scratch, App Inventor for Android and Lego Mindstorms**, SIGITE, New York, USA, 2011.

UNESCO. Padrões de competência em TIC para professores. Organização das Nações Unidas para a Educação, a Ciência e a Cultura, 2009.

UNESCO. **ICT in Primary Education, Analytical Survey – Exploring the origins, settings and initiatives**, Vol 1 – United Nations Educational Scientific and Cultural Organization, 2012. Disponível em: <http://iite.unesco.org/pics/publications/en/files/3214707.pdf>. Acesso em: 10 agosto 2014.

UNICAMP, **Projeto Pedagógico do Curso de Engenharia de Computação, Universidade Estadual de Campinas.** Disponível em: < http://www0.fee.unicamp.br/cg/Proj.Pedag.EC/ProjPedEC2012-Final2.pdf>. Acesso em: 14 junho 2014.

UNIPAMPA-CA, **Projeto Pedagógico do Curso de Engenharia de Software, Universidade Federal do Pampa, Campus Alegrete.** Disponível em: < http://cursos.unipampa.edu.br/cursos/engenhariadesoftware/files/2013/04/PPC-ES-UNIPAMPA.pdf>. Acesso em: 15 junho 2014.

UNIPAMPA-CB, **Projeto Pedagógico do Curso de Engenharia de Computação, Universidade Federal do Pampa, Campus Bagé.** Disponível em: < http://cursos.unipampa.edu.br/cursos/engenhariadecomputacao/projeto-pedagogico-de-curso/>. Acesso em: 14 junho 2014.

USP, **Projeto Pedagógico do Curso de Ciências de Computação, Universidade de São Paulo.** Disponível em: < http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/SECAO-GRAD_90_Projeto%20Pol%C3%ADtico%20Pedag%C3%B3gico%202014%20-%20BCC%20-%2011%20abr%202014.pdf>. Acesso em: 14 Junho 2014.

USS, **Projeto Pedagógico do Curso de Engenharia de *Software*, Universidade Severino Sombra.** Disponível em: < http://www.uss.br/arquivos%3Bjsessionid%3D0DFF8CBC11E69B6C33956641EF04CAFC/graduacao/vassouras/engenhariaComputacao/PPC_ENG_COMPUTACAO_2012_2_DOC.pdf >. Acesso em: 15 junho 2014.

VAHLDICK, A.; BENITTI, F. B. V.; URBAN, D. L.; KRUEGER, M. L.; HALMA, A. **O uso do Lego Mindstorms no apoio ao Ensino de Programação de Computadores.** In anais do XX WEI Workshop de Educação Em Computação, Bento Gonçalves (pp. 523-526), 2009.

VALENTE, José Armando. **Informática na educação: instrucionismo x construcionismo**. Manuscrito não publicado, NIED: UNICAMP, 1997.

VALENTE, José Armando. **O uso inteligente do computador na educação**. Revista Pátio, ano I, n. 1, p. 19-21, mai/jul, 1997.

VASSILEV, Tzvetomir I. **An Approach to Teaching Introductory Programming for IT Professionals Using Games**. International Journal of Human Capital and Information Technology Professionals (IJHCITP), v. 6, n. 1, p. 26-38, 2015.

VALENTE J. A**. O Computador na Sociedade do Conhecimento**. UNI-CAMP/NIED, 1999.

VEIGA, Ilma Passos Alencastro. **A prática pedagógica do professor de Didática**. 2. Ed. Campinas, Papirus, 1992.

VOELCKER, Marta Dieterich; DA CRUZ FAGUNDES, Léa; SEIDEL, Susana. **Fluência digital e ambientes de autoria multimídia**. RENOTE, v. 6, n. 2, 2008.

VON WANGENHEIM, Christiane A. Gresse; VON WANGENHEIM, Aldo v Aldo. **Raciocínio baseado em casos**. Editora Manole Ltda, 2003.

VOSINAKIS, Spyros; KOUTSABASIS, Panayiotis; ANASTASSAKIS, George. **A Platform for Teaching Logic Programming Using Virtual Worlds.** In:Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on. IEEE, 2014.

VYGOTSKY L. S. **Mind in Society – The Development of Higher Psychological Processes**. Cambridge MA: Harvard University Press, 1978.

WAGNER, R.; MOURA, A.; RIGODANZO K. S.; PASSERINO, L. M.; DUTRA Piovesan, S. **VirtualTche–Mundo Imersivo do Instituto Federal Farroupilha–Campus Panambi.** In Anais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 1, No. 1), 2012.

WALD, A.; WOLFOWITZ, J. **An Exact Test for Randomness in the Non-Parametric Case Based on Serial Correlation**. The Annals of Mathematical Statistics. Dec, 1943.

WANG, F.; HANNAFIN, M.J. **Using design-based research in design and research of technology-enhanced learning environments**. In Annual Meeting of the American Educational Research Association, San Diego, CA, 2004.

WANG, F.; HANNAFIN, M.J. **Design-based research and technology-enhanced learning environments.** Education Technology Research and Development, v. 53, n. 4, p. 5-23, 2005.

WENGER, E.; Mcdermott, R.; SNYDER, W. M. **Cultivating communities of practice: A guide to managing knowledge**. Boston, MA: Harvard Business School Press, 2002.

WHALLEY, J. L.; LISTER, R.; THOMPSON, E.; CLEAR, T.; ROBBINS, P.; KUMAR, P. K.; PRASAD, C. "**An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies**", In: VIII Australasian Computing Education Conference (ACE2006), Computer Society, 2006.

WILCOXON, F. **Individual comparisons by ranking methods**. Biometrics, Alexandria, v.1, 1945.

WILEY, David. **Learning Object Design and Sequencing Theory**. Dissertation. Brigham Young University, 2000.

WILEY, D. **The instructional use of learning objects**. 2002. Disponível em: <http://www.reusability.org/read/>. Acesso em: 14 março 2014.

WILEY, David A. (Ed.). **The Instrucional Use of Learning Objects: Online Version**. Disponível em: <http://reusability.org/read>. Acesso em: 22 junho 2014.

WINSLOW, L. E. **Programming Pedagogy – A Psycological Overview**, In: ACM SIGCSE Bulletin, ACM Press, 1996.

WING, J. M. **Computational thinking**. Communications of the ACM, v. 49, n. 3, p. 33–35, 2006.

WIRTH, N. **Algoritms + Data Structures = Programs**. Prentice Hall, 1976.

WOLZ, Ursula; MALONEY, John; PULIMOOD, Sarah Monisha. **'scratch' your way to introductory cs.** In: ACM SIGCSE Bulletin. ACM, 2008.

YIN, R. K. **Case study research: design and methods**. London: Sage Publications, 1994.

YIN, R. K. **Estudo de caso. Planejamento e métodos.** Porto Alegre, Ed. Artmed, 2002.

YIN, Robert K. **Estudo de Caso: Planejamento e Métodos**. Ed. Bookman, 4ª Ed. 2010.

ZUFFO, M. K., **A convergência da Realidade Virtual e a Internet Avançada em novos paradigmas de TV Digital Interativa, Tese de (Livre Docência)** – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Sistemas, 2001. Disponível em: < http://www.lsi.usp.br/interativos/nrv/mkzuffo_livre-docencia.pdf>. Acesso em: 21 junho 2014.