UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ALINE JORDÃO
ANNE CARNEIRO ROCHA

Práticas para Fomentar o Ensino de Programação no Nível Médio

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2015

ALINE JORDÃO
ANNE CARNEIRO ROCHA

Práticas para Fomentar o Ensino de Programação no Nível Médio

Trabalho de Conclusão do Curso de graduação em Sistemas de Informação apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof^a. Dr^a. Marília Abrahão Amaral

CURITIBA

2015

Agradecimentos - Aline

Dedico este trabalho primeiramente, a minha mãe Izoraide que me apoia em todos os momentos da minha vida, a minha orientadora Marília a quem tenho grande adimiração e respeito.

Aos meus amigos, que me apoiaram e que sempre estiveram ao meu lado durante esta caminhada, em especial a minha amiga e dupla nesse trabalho Anne Carneiro Rocha, com quem muitas vezes compartilhei momentos de discussões, alegrias, angústias e ansiedade, mas que sempre esteve ao meu lado.

Ao meu amigo Henrique A. Rodriguez que esteve apoiando nesse trabalho de maneira incrível.

A estes dedico meu trabalho, sem a ajuda, confiança e compreensão, esta etapa não teria se realizada.

Muito Obrigada!

Agradecimentos - Anne

Agradeço a todos que estiveram ao meu redor, oferecendo inestimável suporte e paciência durante o completar de mais um ciclo.

Ao meu pai e mãe, pela confiança, apoio e dedicação que me motivam a seguir sempre em frente.

À minha orientadora Marília, uma pessoa com um coração enorme que é exemplo de força e determinação, não apenas para mim mas para muitas outras pessoas.

Aline, por ser não apenas minha companheira de trabalho, mas também uma grande amiga a qual eu tenho muito carinho.

Aos meus amigos e colegas, dentre estes Henrique A. Rodriguez, que se fez sempre presente nos períodos mais difíceis deste trabalho, e Gustavo Kira, por sua disposição em auxiliar sempre que se fez necessário.

O apoio e boa vontade de todos vocês me inspira a ser uma pessoa melhor. Muito obrigada.

”A Educação de que precisamos há de ser a que liberte pela conscientização. A que comunica e não a que faz comunicados”.

Paulo Freire

Resumo

A área da Computação possui a capacidade de permear diferentes atividades humanas, o que faz com que seja utilizada para a resolução dos mais diversos problemas na sociedade. Entretanto, esta área também enfrenta um cenário deficitário no Brasil, em que pesquisas da IDC Brasil (International Data Corporation) apontam no país um aumento de 117 mil vagas abertas até o ano de 2015, sem que os empregadores encontrem profissionais qualificados para atendê-las. Desta forma, é crescente a preocupação com o ensino de Computação e o incentivo da área em etapas anteriores ao nível de graduação, em especial ao ensino de programação, por ser considerado um dos pilares da área. Este trabalho investiga portanto as práticas educacionais julgadas adequadas para fomentar o ensino de programação no nível médio, obtidas pela intersecção das filosofias construcionista e socioculturalista de ensino, além da seleção de um modelo de avaliação de software educacional. A partir deste modelo e da avaliação de três ferramentas educacionais (Alice, Scratch e Pygame), as práticas selecionadas foram aplicadas por meio de uma aula de oficina de programação em Python com o público visado. Conforme os resultados apresentados e por meio de um questionário direcionado aos estudantes, este trabalho propõe que as práticas verificadas, além de cumprir com os objetivos de aprendizagem propostos, também permitem a criação de espaços para que o pensamento crítico e reflexivo se desenvolva, e servem como veículo para a promoção da criatividade e da motivação dos estudantes.

Palavras-Chave: Ensino de programação, práticas educacionais, ensino médio, Pygame, sociocultural, construcionismo.

Abstract

The Computing area has the ability to permeate different human activities, which makes it be used for solving various problems in society. However, this area also faces a deficit scenario in Brazil, and researches from IDC Brazil (International Data Corporation) points out an increase by 117,000 open positions in the country until 2015, without employers finding qualified professionals to fill them. Thus, there is a growing concern when it comes to the education in computing and the encouragement of the area before undergraduation level, especially the teaching of programming, which is considered one of the pillars of the area. Therefore, this dissertation investigates the appropriate educational practices to promote programming education at high school level, obtained by the intersection of constructionist and sociocultural teaching philosophies, in addition to selecting an evaluation model for educational software. From this model and the evaluation of three educational tools (Alice, Scratch and Pygame), the selected practices were carried out through a workshop class of Python programming, taught to the target audience. According to the results presented and through a questionnaire directed to the students, this dissertation concludes that the observed practices, besides complying with the proposed learning objectives, also allow the creation of spaces for the development of critical and reflexive thinking, and serve as a vehicle for promoting creativity and motivation for students.

Keywords: Programming teaching, educational practices, high school, Pygame, sociocultural, constructionism.

Lista de Figuras

Lista de Tabelas

Conteúdo

1 Introdução

Uma das características da área da Computação está ligada ao fato de permitir trabalhar em conjunto a diversas outras áreas do conhecimento – ou seja, tem a capacidade de permear as atividades humanas, fazendo com que seja amplamente utilizada para a resolução dos mais diversos problemas na sociedade.

Desta maneira, de acordo com Wing (2006), futuros profissionais destas diversas áreas deverão interagir com os de Computação por meio de um pensamento interdisciplinar, e consequentemente, é crescente a importância destes profissionais e o ensino e aprendizagem da área computacional.

Um fator significativo frente às áreas de Computação está ligado à escassez de profissionais. Em um panorama internacional, é possível citar um estudo realizado pela *Association for Computing Machinery* (ACM) no qual Report (2012) afirma que o Governo dos Estados Unidos estima que existam 3,7 milhões de postos de trabalhos abertos no país e que essa escassez de indivíduos com habilidades computacionais poderá piorar.

Esse estudo prevê que entre 2010 e 2020 a economia americana produzirá mais de 120 mil postos de trabalho na área de computação que exigem pelo menos um diploma de bacharel, mas no país, o número de profissionais que têm a formação prevista está muito abaixo dessa demanda chegando apenas a 44 mil profissionais.

Segundo ainda o estudo, empresas como Microsoft têm grande parte de sua receita investida em pesquisa e desenvolvimento (P&D) e dependem destes profissionais, podendo futuramente migrar suas demandas para países que tenham disponíveis indivíduos com essa formação, e fazendo então estimular a economia desses países como concorrentes ao país de origem da empresa.

Já de acordo com pesquisas da *International Data Corporation* (IDC), provedora global de inteligência de mercado, existe atualmente no Brasil uma carência de profissionais na área e até 2015 esse número deve crescer para 117 mil vagas abertas sem que os empregadores encontrem profissionais qualificados para atendê-las (Exame, 2014). Segundo a pesquisa Exame (2014), as principais razões para esse déficit de mão de obra qualificada são a rápida expansão das empresas de infraestrutura e tecnologia no país e a adoção acelerada de serviços de TI (Tecnologia da Informação) pelas iniciativas pública e privada.

Portanto, dentro deste cenário deficitário, é crescente a preocupação com o ensino de Computação e o incentivo à aprendizagem em etapas anteriores ao nível de graduação. Diversas entidades, como a Sociedade Brasileira de Computação (SBC), defendem o ensino de Computação nas escolas assim como o de outras ciências, visando uma educação voltada para as novas exigências sociais relacionadas aos desdobramentos das tecnologias (França et al., 2013).

Em um estudo sobre o ensino de programação no Ensino Médio, Leal (2014) afirma que

a tarefa de programação alinha-se ao desenvolvimento do raciocínio lógico exigindo do programador de computadores criatividade e atenção. Declara Leal (2014) ainda que há vários estudos a respeito do ensino de programação e em especial, relacionados à estudantes novatos à area de programação, o autor cita estudos de Brusilovsky et al. *(Teaching programming to novices: A review of approaches and tools)*, Soloway *(Proust: Knowledge-based program understanding)*, Delgado et al *(Identificando competências associadas ao aprendizado e de leitura e construção de algoritmos)* e Denhadi *(A Cognitive Study of Learning to Program in Introductory Programming Courses)*, então afirma que esses estudos têm esse foco devido ao fato de que muitos desses estudantes apresentam grandes dificuldades, seja por uma formação deficiária na área de Ciências Exatas ou a motivação que pode ser grande no início do curso e decai no decorrer do curso.

Leal (2014) complementa argumentando que estudantes do Ensino Médio iniciam o cursos na área de Informática sem ter uma noção clara do que irão estudar, por dificuldades relacionadas a conhecimentos prévios, ou por não encontrarem motivação suficiente para a atividade de programar.

Os estudos citados por Leal (2014), são classificados em quatro áreas. A primeira área aborda o currículo de cursos introdutórios de programação; os métodos pedagógicos com os quais o ensino e a aprendizagem de programação são conduzidos são discutidos na segunda área e a terceira trata sobre a escolha da linguagem de programação para o ensino de programação para estudantes novos à área. Por fim, a quarta área desenvolve-se em estudos sobre o desenvolvimento de ferramentas computacionais para ensino de programação. Será portanto de foco desse trabalho relacionar os estudos da segunda à terceira área de estudo de acordo com essas pesquisas.

Considerando este cenário atual, e tendo em vista que a aprendizagem da programação se faz um dos pilares da área de computação, além de ser considerada uma tarefa particularmente difícil que envolve diversos conhecimentos e habilidades Jenkins (2002), este Trabalho de Conclusão de Curso visa desmistificar o ensino de programação para o Ensino Médio no Brasil a fim de incentivar estudantes a prosseguirem seus estudos na área de Computação.

1.1 Objetivos

1.1.1 Objetivos Gerais

Desenvolver práticas para fomentar e desmistificar o ensino de programação no nível médio.

1.1.2 Objetivos Específicos

- Comparar métodos já aplicados no ensino de programação à estudantes do Ensino Médio;

- Definir práticas educacionais compatíveis com tais métodos. Aliar cada método estudado e suas respectivas práticas a uma ferramenta de ensino de programação;
- Caracterizar um perfil de classe/participante a ser envolvido nas práticas;
- Avaliar um ou mais métodos selecionados por meio de práticas definidas com o público alvo determinado; e
- Apresentar os resultados como forma de orientar eventual aplicação ou estudo relativo à área computacional.

1.2 Motivação

O que motiva a realização desse trabalho é a preocupação com a mudança necessária no ambiente escolar frente aos novos anseios e necessidades do mundo contemporâneo. A mudança no modo de aprender e sua essência é necessaria conforme o passar dos tempos, porém toda mudança deve ser pensada e estudada para que seus resultados sejam positivos.

Portanto, a oportunidade acerca de uma melhor compreensão dos caminhos para um aprendizado mais significativo para estudantes do ensino médio - e consequentemente visando oferecer novas oportunidades, gera um desafio motivacional para despender esforços na realização desse trabalho.

1.3 Justificativa

A necessidade de profissionais na área de Computação é cada vez maior e isso faz com que se pense cada vez mais cedo em estratégias de como atrair novos estudantes, ao mesmo tempo que também se buscam formas de diminuir a evasão destes em cursos nessa área, como destacado pela diretora de Educação da SBC, (Brasil, 2012, p. 16):

> "As estatísticas mostram que nos próximos cinco anos haverá um déficit de 300 mil profissionais das áreas de Tecnologia da Informação e Computação. E a SBC está estimulando a discussão sobre novas formas de ensino, já que sabemos que existe uma evasão muito grande de estudantes nos cerca de 1.700 cursos de graduação existentes no Brasil".

Uma das soluções que vem sendo adotada é a introdução de disciplinas que abordem conteúdos sobre programação antes dos estudantes entrarem no Ensino Superior, e, desta forma, a partir do Ensino Médio apresentar uma visão das áreas abrangidas pela Computação. Como afirmado por (França et al., 2013, p. 23), o ensino de programação no ensino médio permite:

> "[..] o desenvolvimento de diversas capacidades que contribuem para melhorar o raciocínio lógico dos estudantes. Programar envolve a habilidade de desenvolver uma solução para um problema, que se for grande requererá o exercício de outras habilidades (como dividir o problema em subproblemas e criar uma solução central)".

Este Ensino no Nível Médio também forneceria uma base melhor para estes estudantes antes dos mesmos ingressarem no Ensino Superior, podendo também ser um fator motivacional para que estes escolham a área de computação para prosseguir seus estudos.

Além disso, estudos apontam a importância de que o ensino de programação para novatos seja acompanhado de uma metodologia que mantenha os estudantes engajados e motivados para que as dificuldades possam ser superadas e para que os mesmos continuem interessados em aprender (Jenkins, 2002).

Com isso, firma-se a importância de se aliar a pesquisa às ferramentas [1] disponíveis para o ensino de programação pois ela (a ferramenta), quando escolhida devidamente, pode vir a possibilitar um bom ambiente para a aprendizagem destes estudantes. Logo, cada ferramenta possibilita uma maneira de interagir e interpretar um problema apresentado, podendo esta ser associada com um método específico - que quando avaliados irão testar o seu grau motivacional em estudantes.

Para Leal (2014), em seu estudo *Ensino de Programação no Ensino Médio Integrado*, ao se depararem com conteúdos relacionados à programação de computadores, muitos estudantes não se identificam com a área, seja pelas dificuldades relacionadas a conhecimentos prévios, ou pela falta de motivação para a atividade de programar. Leal (2014) afirma que além da motivação, é necessário desenvolver competências relacionadas a criatividade, pois programação está intimamente ligada a este fator. Portanto, a maneira com a qual o conteúdo é abordado influencia na motivação e no aprendizado destes estudantes, sendo importante desmistificar a maneira com que estes encaram o aprendizado e a área de Computação.

Para a maioria dos ingressantes em cursos voltados à área de Computação, as matérias em que mais se encontram dificuldades são as de programação e lógica, por justamente não terem tido uma visão antes de seu ingresso (Barreiros et al., 2014). Logo, se faz fundamental que os estudantes possam ter conhecimentos básicos de Computação desde o início da vida escolar, visto que o ponto crucial desta ciência é a compreensão e habilidade de desenvolver algoritmos - na qual também se encontram as principais dificuldades de aprendizagem.

Portanto, este trabalho é justificado pela necessidade de uma preparação maior de como inserir esses assuntos no ambiente acadêmico, e também na necessidade de encontrar modelos de referência a métodos que não encarem o ensino de programação apenas como o ensinar

[1] Ferramenta de programação ou software é um programa ou aplicativo utilizado para criar, depurar, manter, ou realizar alguma outro tipo de apoio para a criação de outros programas e aplicativos.

de uma nova linguagem, mas também como o despertar da motivação no estudante. Esta diferente abordagem ajudaria a promover o processo criativo, uma vez que existem muitas dificuldades encontradas pelos estudantes ao ingressar na área de computação.

1.4 Organização do Trabalho

A organização deste trabalho acontece de maneira a apresentar a Introdução e Metodologia descritas no início deste trabalho, por meio dos Capítulos 1 e 2, respectivamente.

O Referencial Teórico é levantado no Capítulo 3, em que são abordados os estudos acerca das teorias de aprendizagem e ferramentas para o ensino de programação, apresentados nas Seções 3.1 e 3.2, respectivamente. A Seção 3.3 trata de levantar as bases educacionais do curso de Sistemas de Informação, relacionando-as com as visões metodológicas de ensino estudadas, e na Seção 3.4 é realizada a fundamentação para a análise de softwares educacionais.

O Capítulo 4 trata do desenvolvimento do trabalho, no qual são abordadas as seleções das visões metodológicas, métodos de avaliação de software educacional e também as ferramentas a serem utilizadas na aula da oficina. Este capítulo também apresenta a contextualização da oficina, assim como o planejamento efetuado para a realização da mesma (Seções 4.4 e 4.5).

Os resultados obtidos por meio da realização da aula da oficina são apresentados pelo Capítulo 5, bem como a discussão destes na Seção 5.4. Por fim, as considerações finais deste trabalho se apresentam no Capítulo 6.

2 Metodologia

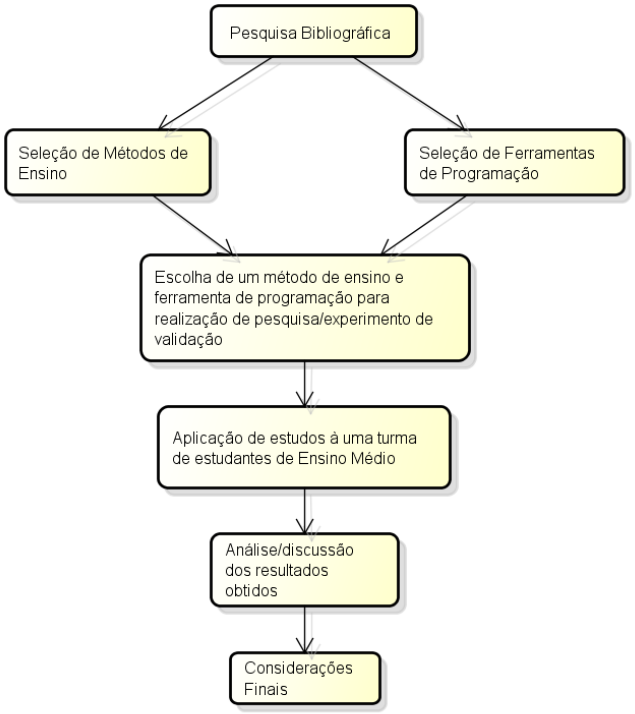


Figura 1: Desenvolvimento do Trabalho. Fonte: As autoras.

A metodologia de pesquisa científica desenvolveu-se a partir das seguintes classificações de tipo de pesquisa: natureza, forma de abordagem, objetivos e procedimentos Almeida(2012), nesse trabalho a classificação segue da seguinte maneira:

- Natureza, como uma pesquisa aplicada, a qual teve como objetivo desenvolver práticas dirigidas à dismistificação do ensino de programação, fomentando assim a área de Computação.
- Forma de abordagem, utilizando uma pesquisa de forma quantitativa e também qualitativa, foram traduzidas em números as opiniões e informações para serem classificadas e analisadas, e ao mesmo tempo em que se baseou em dados descritivos e subjetivos dos estudantes sobre percepções referentes à área de Computação.
- Objetivos, foram abordados na forma de uma pesquisa exploratória, a qual proporcionará maior familiaridade com o problema, explicitando-o. Foram estipulados critérios e métodos para a elaboração da pesquisa.
- Procedimentos, foram sustentados pela revisão da literatura, elaborada a partir de material já publicado, como livros, artigos, periódicos, estudo de casos, etc; e também

uma pesquisa experimental, tendo como objeto de estudo estudantes do ensino médio e definindo-se as formas de controle e de observação dos efeitos que os métodos estudados produzirão.

A metodologia empregada para realização deste trabalho foi definida nas seguintes etapas, com um conjunto de tarefas para que seja possível atingir o resultado esperado.

Primeiramente foi feita uma revisão da literatura acerca de teorias de aprendizagem utilizadas para o ensino e aplicadas à estudantes do Ensino Médio. Esse estudo permitiu criar uma base sólida para então se dar prosseguimento ao trabalho.

A partir disso, foi definido qual metodologia seria abrangida neste estudo e como esta poderia ser aplicada ao ensino de programação no ensino médio - e a seleção de uma ferramentas correspondente.

A intenção foi definir práticas educacionais compatíveis com métodos direcionados a um público específico, portanto, a área de pesquisa foi delimitada a estudos realizados com estudantes do ensino médio.

Então, foi selecionada uma ferramenta para ser avaliada por meio de de uma oficina. Esta oficina foi realizada com um grupo de estudantes do ensino médio em local previamente definido. Por meio de um questionário, foi avaliado interesse do estudante na área de computação depois da oficina, para ver qual o grau de interesse que os métodos empregados geraram no estudante. As pesquisas realizadas foram de cunho tanto quantitativo como qualitativo, pós a transcrição das informações, foram classificadas e analisadas ao mesmo tempo em que foram utilizadas opiniões, dados descritivos e subjetivos dos estudantes.

Foram também colhidas informações com os instrutores da oficina a respeito dos estudantes que participaram da mesma, através de dois questionários, um deles visou abordar o desempenho dos estudantes durante a oficina, e outro relativo a abordagem metodológica praticada na oficina, esse segundo, foi aplicado aos instrutores de edições anteriores dessa oficina.

Esses dados serviram para comparar as práticas desenvolvidas nas oficinas de programação em Python, e verificar a influência de uma metodologia de ensino no processo de aprendizagem.

O desenvolvimento do trabalho seguiu o esquema apresentado na Figura 1.

3 Referencial Teórico

Neste capitulo serão apresentadas algumas definições dos temas estudados, por autores das áreas específicas a fim de proporcionar um melhor entendimento no processo de desenvolvimento do trabalho.

3.1 Teorias de Aprendizagem e Ensino

O termo 'teoria de aprendizagem' é compreendido por Moreira (2011) como sendo "uma construção humana para intrepretar sistematicamente o tema aprendizagem, representando o ponto de vista de um autor/pesquisador sobre como intrepretar o tema", no qual esse autor/pesquisador estuda o que é, como funciona, e porque funciona o processo de aprendizagem.

Bower and Hilgard (1981) afirmam em sua obra, *Teorias de Aprendizagem*, que aprendizagem pode ser a referência a "mudança no comportamento ou no potencial do comportamento de um organismo em uma situação determinada, que se baseia em experiências repetidas do organismo nesta situação."

Para Prass(2012), teoria de aprendizagem é como denomina-se as diversas concepções que visam explicar o processo de ensino-aprendizagem. Diversos pesquisadores (Skinner, Piaget, Rogers, Freire...) construíram teorias no qual explicam como deveria ser abordado o processo de ensino para que se tivesse uma melhor e maior compreensão, de que se adquirisse maior conhecimento no processo de educação.

Essas teorias não somente contribuem para o aprendizado dos estudantes, mas segundo Moreira (2011), também contribuem para que os professores tenham melhores estratégias de ensino, enfoques didáticos e preparem melhor os materiais instrucionais.

Moreira (2011) classifica as teorias de aprendizagem de acordo com três filosofias subjacentes - a cognitivista *(construtivismo)*, a comportamentalista *(behaviorismo)* e a humanista, conforme podem ser observadas no esquema conceitual da Figura 2, que mostra os enfoques teóricos relacionados a aprendizagem e ao ensino, uma determinada teoria de aprendizagem não está ligada necessariamente a uma única filosofia como pode ser observado pela esquematização.

Aprendizagem pode ser entendida e classificada de várias maneiras e está diretamente relacionada com o conceito de ensino. Salles (2012) descreve que ensino pode ser definido como um meio de instrução, transferência, ou treinamento, envolvendo recursos didáticos para ajudar a adquirir conhecimento e saber usá-lo.

A educação é um processo de ensino-aprendizagem no qual o indivíduo aprende a aprender, Salles (2012) afirma que o indivíduo aprende a desenvolver de forma independente, um processo que ajuda a integrar todas as dimensões da vida, levando o indivíduo a participar

criar, inovar, e pensar.

O esquema conceitual apresentado na Figura 2 tem no topo o conceito de enfoque teórico, seguido pelos três principais enfoques definidos com palavras-chaves pelo autor e em sua base os autores que se relacionam em seus estudos com os enfoques apresentados.

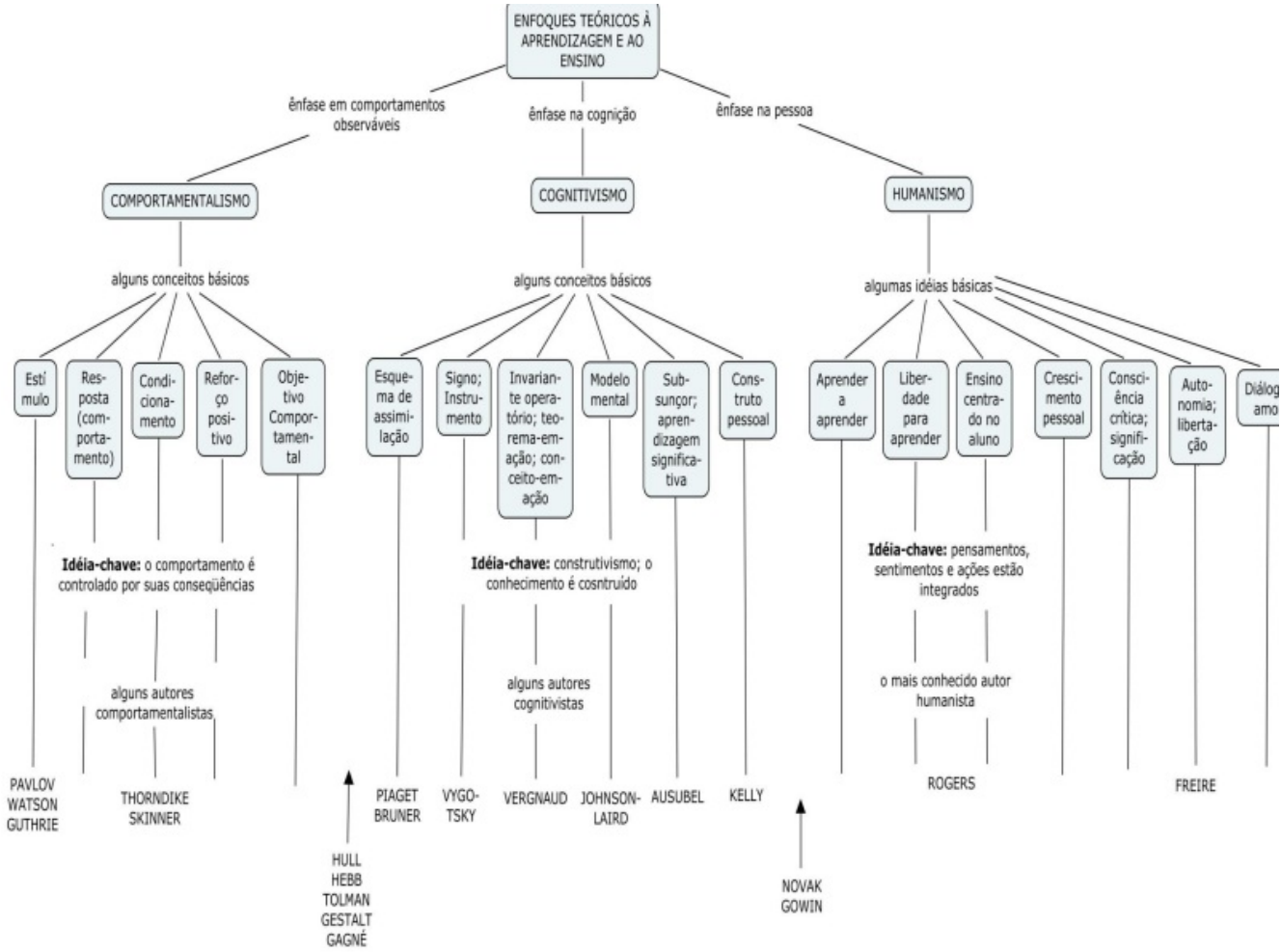


Figura 2: Esquema Conceitual dos Principais Enfoques Teóricos. Fonte: (Moreira, 2011)

Merriam e Caffarella (1991) apresentam em seu estudo, *Learning in Adulthood: A Comprehensive Guide*, um quadro comparativo entre as teorias de aprendizagem, que pode ser visto na Tabela 1

Tabela 1: Quadro comparativo dos diferentes aspectos entre as teorias de aprendizagem. Fonte: (Moreira, 2011)

Diferentes Aspectos	*Behaviorista*	Cognitivista	Humanista
Teóricos de aprendizagem	Thorndike, Skinner	Lewin, Piaget	Maslow, Rogers
Visão do processo de aprendizagem	Mudança no comportamento	Processo mental interno (incluindo *insight*, processamento de informações, memória e percepção)	Um ato pessoal para realizar potencial
Locus da aprendizagem	Estímulo no ambiente externo	Estrutura cognitiva interna	Necessidades cognitivas e afetivas
Propósito da educação	Produzir mudança de comportamento em direção desejada	Desenvolver capacidade e habilidades para aprender melhor	Tornar-se auto-atualizado e autônomo
Manifestação na aprendizagem de adultos	• Objetivos *behavioristas* Desenvolvimento e treinamento de habilidades	•Inteligência, aprendizagem e memória como função da idade •Aprendendo como aprender	•Andragogia •Aprendizagem autodirecionada

A partir dessas informações, define-se as teorias de aprendizagem estudadas nesse trabalho: Comportamentalismo/Behaviorista, Cognitivista, Humanista, e Sociocultural, e para cada teoria apresenta-se os autores que representam-a.

3.1.1 Comportamentalismo

Para Moreira (2011) a filosofia do comportamentalismo baseia-se em comportamentos observáveis e mensuráveis do sujeito e as respostas que ele dá aos estimulos externos, em que manipulando-se eventos posteriores à exibição de um comportamento pode-se em princípios controlá-los. Moreira (2011) afirma que o comportamentalismo aborda o que as pessoas fazem, contrapondo-se ao mentalismo, ao estudo do que as pessoas pensam e sentem. Já (Ostermann et al., 2010)afirma que é uma teoria baseada em estímulo-resposta (E-R), buscando relações funcionais entre estímulos e consequências, indicando que o comportamento humano é previsível.

Algumas abordagens do comportamentalismo se destacam pelo pensamento e influência dos autores descritos a seguir, John B. Watson, Edward L. Thorndike, e Burrhus Frederic Skinner:

Watson John B. Watson (1878-1958) é considerado o pai do comportamentalismo (ou behaviorismo), Moreira (2011) afirma que Watson criou o termo behaviorismo para elucidar sua preocupação com os aspectos observáveis do comportamento. Watson focalizava em seus estudos muito mais os estímulos do que as consequências, pretendia abordar o que as pessoas fazem, e omitia discussões sobre consciência.

Ostermann et al., (2010) afirma que Watson era influenciado pelo condicionamento clássico de Ivan Pavlov (1849-1936), isso era evidenciado nas suas explicações sobre aprendizagem (associação estímulo-resposta) abordados na forma do condicionamento clássico, e Watson não se interessava pelo reforço ou pela punição (consequências) como causas da aprendizagem.

De acordo com Ostermann et al., (2010), Watson usava dois princípios em suas explicações, o da frequência onde ele afirma que quanto mais frequentemente uma dada resposta é associada a um certo estímulo, maior a probabilidade de que essa associação ocorra outra vez e o da recentidade, que refere-se a quanto mais recentemente uma certa resposta é associada a um dado estímulo, mais provavelmente será associada outra vez.

Pimentel (2002) explica que para Watson os seres humanos já nascem com certas conexões estímulo-resposta herdadas, chamadas reflexos. Watson sugere que a aprendizagem ocorre a partir de um condicionamento destas conexões, bem como na construção de novas conexões estímulo-resposta através do condicionamento clássico. Para ele o meio ambiente exerce uma grande influência sobre o indivíduo (o homem estaria à mercê do meio).

Edward L. Thorndike Edward L. Thorndike (1874-1949) teve grande influência nas origens do behaviorismo. Moreira (2009) declara que sua grande contribuição está na chamada Lei do Efeito, no destaque que deu nas consequências do comportamento como determinantes das conexões E-R.

Moreira (2011) afirma que para esse autor, a aprendizagem consiste na formação de ligações estímulo-resposta que assumem a forma de conexões neurais, e que essas conexões são de natureza fisiológicas. Moreira (2009) explica que Thorndike acreditava que os seres humanos chegam a respostas apropriadas em grande parte por ensaio-e-erro. Para ele, a concepção de aprendizagem estava sujeita a três leis: a Lei do Efeito, a Lei do Exercício, e a Lei da Prontidão.

A Lei do Efeito se dá quando uma conexão é seguida de uma consequência satisfatória que é então fortalecida e, enfraquecida quando a consequência é insatisfatória, dependendo da situação a frequência de resposta aumenta ou diminui. A Lei do Exercício diz respeito ao fortalecimento das conexões com a prática (lei do uso) e o enfraquecimento ou esquecimento que acontece quando há a falta de prática (lei do desuso). A Lei da Prontidão é quando há uma preparação para ação, se a ação é concretizada, a ação é satisfatória, caso contrário, a ação é insatisfatória.

Skinner Burrhus Frederic Skinner (1904-1990), Figura 3, segundo afirma Zacharias (2008), baseou suas teorias na análise das condutas observáveis. Dividiu o processo de aprendizagem em respostas operantes e estímulos de reforço. Skinner sempre esteve preocupado com as aplicações práticas da psicologia, destacando-se nesse quesito a "educação programada".

Figura 3: Burrhus Frederic Skinner Fonte (Zacharias, 2008)

A aprendizagem era para Skinner basicamente uma mudança de comportamento que é ensinado por meio de reforços imediatos. Pimentel (2002) explica que Skinner propôs mudanças no processo de ensino de sua época criando técnicas para conduta em sala de aula, pois para ele o professor sozinho não tinha condições para dar reforço a todos os estudantes ao mesmo tempo. E isso gerou a grande necessidade de se utilizar instrumentos mecânicos para esta função reforçadora para auxiliar o professor.

Assim, Skinner propôs a utilização das "máquinas de ensinar", Figura 3, em que o estudante deveria responder a uma questão ou problema. Se a resposta fosse correta, um mecanismo seria liberado para a próxima pergunta, podendo estar associado, por exemplo, a um som, como reforço. Se a resposta fosse incorreta, o mecanismo não se acionaria e o estudante faria outra tentativa.

O uso dessas máquinas tinha como principal objetivo estimular uma atitude ativa do estudante, no que se refere às respostas, por ele mesmo elaboradas. Dessa forma se garantiria o sucesso em sua aprendizagem, fazendo com que o estudante se motivasse constantemente.

As primeiras "máquinas de ensinar", Figura 4, surgiram por volta de 1920, foram desenhadas várias máquinas para testar automaticamente a inteligência e a informação.

Figura 4: A máquina de ensino de Skinner Fonte (Pimentel, 2002)

Zacharias (2008) afirma que essas máquinas eram a organização de material didático de maneira que o estudante pudesse utilizar sozinho, recebendo estímulos à medida que avançava no conhecimento, esses estímulos baseavam-se na satisfação de dar respostas corretas aos exercícios propostos.

Esclarece ainda Zacharias (2008) que Skinner considerava o sistema escolar predominante um fracasso por se basear na presença obrigatória, sob pena de punição. Segundo Zacharias (2008), Skinner defendia que se dessem aos estudantes "razões positivas"para estudar, como prêmios aos que se destacassem.

De acordo com Pimentel (2002) "as influências de Skinner e sua Instrução Programada

Linear podem ser consideradas como as primeiras abordagens adotadas no uso do computador aplicado à Educação, e as suas idéias básicas prevalecem ainda hoje na construção de muitos softwares (por exemplo: tutoriais, programas usados em cursos de língua estrangeira, etc.)"

Para Zacharias (2008) muito da Tecnologia Educacional Moderna se baseia nos pressupostos do behaviorismo, assim como muitos softwares ditos "educativos", que nada mais são que atividades fechadas baseadas em reforços positivos.

3.1.2 Cognitivismo

A filosofia cognitivista[2] enfatiza o modo como o ser humano pensa e sente, contrapõe-se ao comportamentalismo. Essas duas filosofias surgiram na mesma época. Moreira (2011) declara que "a filosofia cognitivista aborda principalmente os processos mentais e se ocupa de atribuição de significados, da compreensão, transformação, armazenamento e uso da informação envolvida na cognição."

Marques et al., (2009) descreve o cognitivismo como a filosofia que estuda como o indivíduo conhece, como percebe, processa a informação, compreende, dá significados, e constrói estruturas cognitivas. Segundo (Marques et al., 2009, p. 12) ainda:

> "Cognição refere-se a um conjunto de habilidades cerebrais/mentais necessárias para a obtenção de conhecimento sobre o mundo. Tais habilidades envolvem pensamento, raciocínio, abstração, linguagem, memória, atenção, criatividade, capacidade de resolução de problemas, entre outras funções."

Para a filosofia cognitivista, estão representados nesse trabalho os autores David Ausubel e Jean Piaget, cuja participação e colaboração para a corrente filosófica é descrtita a seguir:

Ausubel Ostermann et al., (2010) afirma em sua obra *Teorias de Aprendizagem*, que David Ausubel (1918 - 2008) tinha como conceito principal em sua teoria o de aprendizagem significativa. De acordo com Moreira (2011), Ausubel define aprendizagem como organização e integração do material na estrutura cognitiva, e considera o que o estudante já sabe como fator que mais influencia na aprendizagem, então o professor deveria identificar e ensinar de acordo com esse fator. Novas ideias e conceitos devem ser instruídas aos poucos, depois que o estudante já tenha claro os conceitos relevantes e inclusivos. Acontece uma interação entre os conceitos já fixados e os novos conceitos, na qual os conceitos já fixados agem como âncoras

[2]Alguns autores estudados neste trabalho são classificados como cognitivistas por seus estudos e teorias originalmente se voltarem para a investigação dos processos da inteligência humana. Entretanto, mais tardiamente, conforme aponta (Revista, 1995), outros estudiosos se basearam em suas descobertas para formular novas propostas pedagógicas, surgindo desta forma a linha pedagógica do construtivismo, que visa o processo de construção do conhecimento e que se baseia principalmente nos estudos desenvolvidos por Jean Piaget. Desta forma, ainda de acordo com (Revista, 1995), esta nova linha pedagógica passou a reconhecer em certos autores, como Piaget e Vygotsky, compatibilidades com suas ideias construtivistas, ainda que estes não tenham sido os fundadores da mesma.

para o novo conteúdo, abrangendo-o, integrando-o, e ao mesmo tempo, modificando-se em relação a esse novo conteúdo.

O conceito de aprendizagem significativa de Ausubel é explicado por Moreira (2011), que relata que esse ocorre quando uma nova informação integra-se com conceitos e preposições relevantes preexistentes na estrutura cognitiva do aprendiz. O armazenamento de informações no cérebro humano para Ausubel é organizado como hierarquia conceitual, onde os conceitos específicos de um determinado assunto são assimilados a conceitos gerais já compreendidos pelo aprendiz. Portanto uma das condições para a aprendizagem significativa ocorrer é que o conteúdo seja relacionável.

Jean Piaget Jean Piaget (1896 - 1980) é o mais conhecido autor do cognitivismo do século XX por sua teoria de desenvolvimento cognitivo; de acordo com Moreira (2011), Piaget não desenvolveu uma teoria de aprendizagem, e sim uma teoria de desenvolvimento mental. Os conceitos chaves da teoria de Piaget são *assimilação, acomodação e equilibração.* Moreira (2011) afirma que Piaget considera as ações humanas como base do comportamento humano e não as sensações e emoções, ou seja, tudo no comportamento tem início na ação, o pensamento então é a interiorização da ação, acompanhada geralmente pela atividade motora, como por exemplo gestos e movimento dos olhos.

Piaget discorda de que a modificação de pensamento resultante da experiência gere conhecimento, pois essa ideia para ele traz uma dependência do ambiente, para ele, o aumento de conhecimento, ou seja, a aprendizagem, se dá quando o esquema de assimilação (períodos do desenvolvimento mental) sofre acomodação. Piaget destaca em seus estudos a importância de compatibilizar o ensino com o nível de desenvolvimento mental do estudante e não em um nível puramente formal.

3.1.3 Humanismo

O processo de aprendizado baseado na corrente humanista tem para Santos (2005), o enfoque no sujeito. Santos (2005) afirma que a corrente humanista pode apresentar enfoque na interação sujeito-objeto. Santos afirma ainda que o o processo de aprendizagem se dá com a participação do estudante, explorando a criatividade do aprendiz em uma escola que deve favorecer a autonomia e condições de desenvolvimento ao aprendiz, uma escola dita democrática.

Marques (2009) classifica a metodologia humanista como um método no qual a aprendizagem envolve o corpo, o intelecto, e sentimentos de maneira inseparável. Como apresentado na Figura 2 essa filosofia tem como algumas de suas ideias básicas o ensino centrado no estudante, o crescimento pessoal, e como aprender a aprender.

Seguindo semelhante descrição, Moreira (2011) afirma que a filosofia humanista “vê o

ser que aprende, primordialmente, como pessoa". Nesse processo primeiramente se dá importância para a auto-realização. A unidade do aprendiz está no todo, engloba sentimentos e ações, a aprendizagem vai além do aumento de conhecimento, tem papel influenciador nas atitudes e escolhas do indivíduo. Como referência para essa corrente filosófica de ensino, tem-se a representação de Carl Ranson Rogers, descrito a seguir:

Rogers Moreira (2011) afirma ao referir-se a Carl Ransom Rogers (1902-1987) que esse autor vê a educação como um modo de facilitar a aprendizagem, Rogers relaciona uma semelhança no "papel" do professor comparado ao de um terapeuta e ao aluno a semelhança de cliente[3], deste modo o professor tem em sua função facilitar o processo de aprendizagem para o estudante.

De acordo com Prass (2012), a teoria Rogeriana possui características que colocam-a entre o Behaviorismo e a psicanálise de Freud. Rogers apresentava um contraste à psicanálise de Freud, que tinha a prática limitada pela ortodoxia e o *behaviorismo* representado na época por B. F. Skinner (1904-1990) caracterizado pela submissão biológica. A teoria de Rogers ajusta-se na corrente filosófica humanista por fundamentar uma visão humanista do homem.

Rogers(1977) pressupõe alguns princípios que descrevem o processo de aprendizagem, Moreira (2011) descreve esses princípios em que Rogers (1977) alega que o ser humano tem "potencialidade natural"para aprender, pois possui curiosidade a respeito do mundo, porém, essa curiosidade pode ser sufocada pelo sistema educacional. Quando o aprendiz percebe a relevância do conteúdo que lhe é apresentado, para seus objetivos, há uma aprendizagem significativa e o estudante ao perceber a importância para seu desenvolvimento pessoal, torna-se motivado, tornando o processo de aprendizagem mais rápido.

Rogers (1977) exemplifica como as ameaças externas podem influenciar negativamente no processo de aprendizagem, em um de seus exemplos assume-se que um estudante "fraco" em leitura, naturalmente já se sentirá ameaçado em momentos de leitura, quando colocado em situações em que a leitura é obrigatória em voz alta em frente a um grupo, a pressão e o risco de ser ridicularizado, tornarão o processo de aprendizagem desmotivador, diferente de quando a ameaça é pequena ou nula. Nesse caso o processo de aprendizagem será efetivo.

Esses princípios Rogers (1977) afirma ainda que grande parte do aprendizado vem por meio de atos, e defende que o estudante deve participar ativamente do processo de aprendizagem, pois isso lhe beneficiará, fazendo com que se adapte melhor as mudanças do mundo, se aprender a aprender.

> "O único homem que se educa é aquele que aprendeu como aprender: que aprendeu como se adaptar e mudar; que se capacitou de que nenhum co-

[3]Rogers utiliza o termo "cliente"em vez de paciente, por que o primeiro enfatiza uma participação ativa, voluntária e responsável do indivíduo nas relações terapêuticas e também uma igualdade entre a pessoa que procura ajuda e o terapeuta (Moreira, 2011).

> nhecimento é seguro, que nenhum processo de buscar conhecimento oferece uma base de segurança"(Rogers, 1978, p. 116)

Moreira (2011) relata que Rogers tinha uma visão negativa sobre o ensino no modelo usualmente utilizado, pois dessa maneira projeta questões erradas fazendo entender que o que é ensinado é aprendido e o que é apresentado é assimilado. A crítica deriva ao fato de decisões que ponderam sobre o que ensinar, ou sobre o ponto de vista de um professor, o que é relevante para o conhecimento do estudante.

Interando o raciocínio, Rogers (1977) certifica que o professor no papel de facilitador da aprendizagem deve aprimorar características que o definam como pessoa, a fim de aproximar-se do estudante, como compreensão empática, e autenticidade.

3.1.4 Sociocultural

A educação fundamentada em um método de ensino associado a uma corrente sociocultural de acordo com Santana (2010), fez com que o estudante adquire conhecimentos que estão adequados à realidade vivida socialmente por este. Santana (2010) explica que o método de ensino aproxima o conteúdo das disciplinas lecionadas ao que o estudante vive na sociedade.

A corrente sociocultural pode ser denominada como uma corrente sócio-político-cultural ou ainda como uma corrente sócio-crítica na qual os estudantes serão transformados em agentes críticos, com um ensino pluralizado, segundo Santana (2010), devido a necessidade de contato desses estudantes com outras culturas. Essa corrente procura uma democratização do conhecimento e do aprendizado, dando oportunidades iguais de acesso aos saberes aos menos favorecidos socialmente já proporcionados às classes mais privilegiadas na sociedade.

Segundo Santana (2010) as instituições de ensino que optarem por esse método devem banir qualquer marginalização de seus estudantes a fim de evitar e corrigir os erros que porventura advenham de outras instituições. Santana (2010) afirma que o estudante inserido nesse processo de ensino terá a possibilidade de ter uma visão critica de todo conhecimento transmitido a ele, tornando-se politicamente consciente, já que o aprendiz interage com o aprendizado por meio de suas experiências no cotidiano.

Abreu et al., (1997) situa o autor Paulo Freire como referencia à abordagem sociocultural, e informa que havia grande preocupação com a cultura popular advinda desde a segunda Guerra Mundial. Para Abreu et al., (1997) no ensino sinalizado no contexto sociocultural, o homem é sujeito ativo no processo de educação, e toda ação educativa presente nas instituições de ensino, para que seja efetiva, deve-se passar por uma análise do homem e seu meio de vida.

Na concepção de Mizukami (1986) o processo de aprendizagem sociocultural é voltado para o movimento de cultura popular nos países industrializados e em países em desenvolvimento, assim como no Brasil é voltado para as camadas socio-econômicas inferiores tendo destaque

para a alfabetização de adultos. Mizukami (1986) posiciona a obra de Paulo Freire nesse processo de aprendizagem, qualificando-o de modo que sua obra passa a ser referência nesse campo de ensino no qual o homem é o sujeito da educação. Mizukami (1986) enfatiza que há, no processo sociocultural, uma abordagem interacionista, ressalva como imprescindível a interação homem-mundo para o desenvolvimento do sujeito e para que esse se torne sujeito de sua práxis. [4] Mizukami (1986) ainda ratifica que:

> "Sendo o homem o sujeito da sua própria educação, toda a ação educativa deverá promover o próprio individuo e não ser instrumento de ajuste deste à sociedade." (Mizukami, 1986, p. 86)

Com isso, Mizukami (1986) conclui que a práxis levará o humano a desenvolver um pensamento critico sobre o que o rodeia para só então chegar a libertação do seu pensamento e poder participar ativamente da história. Para isso, tem a definição de história como sendo as propostas dadas pelo homem à natureza, às estruturas sociais, e a outros homens. A participação só é possível segundo Mizukami (1986) quando o homem tem uma conscientização e seu pensamento é desmitificado.

A desmitificação se dá em um processo humanizante em que o humano adquire uma consciência crítica de uma realidade, destruindo os mitos que ajudam a prender o sujeito em uma estrutura dominante e que constrói uma sociedade objeto cujos processos culturais e o pensamento-linguagem são alienados, e, desta forma, a falta de elaboração de um pensamento autêntico gera uma distorção da compreensão da realidade objetiva.

No processo de educação sociocultural o conhecimento é concebido por meio do mútuo condicionamento, pensamento e, prática. Mizukami (1986) avalia que o conhecimento está ligado à conscientização, e essa se dá lentamente. Mizukami (1986) defende ainda que toda ação educativa tem que estar ligada à conscientização do homem e a uma análise do meio de vida desse para que seja uma ação educativa válida. Nesse aspecto Mizukami (1986) declara que:

> "A ausência de uma reflexão sobre o homem implica o risco de uma adoção de métodos educativos e diretrizes de trabalho que o reduzem à uma condição de objeto. A ausência de uma análise do meio cultural implica o risco de se realizar uma educação pré-fabricada, não adaptada ao homem concreto a quem se destina." (Mizukami, 1986, p. 94)

Conclui-se que processo de ensino-aprendizagem, uma educação problematizadora ou conscientizadora, objetiva o amadurecimento do pensamento crítico e a liberdade. De acordo com

[4] Práxis significa conduta ou ação. Corresponde a uma atividade prática em oposição à teoria. Aqui pode ser definida como Práxis, pois possui dois aspectos ao mesmo tempo distintos e inter-relacionados: o lado real, a realidade escolar; e o lado ideal, a teoria que orienta as práticas pedagógicas.

Moreira (2009), na concepção sociocultural do desenvolvimento, não deve-se mediar uma aprendiz como um ser isolado de seu meio sociocultural, uma análise do desenvolvimento do aprendiz, a avaliação de suas aptidões e sua educação será errônea se omitirmos seus vínculos.

Dois importantes autores dessa corrente filosófica, Lev Semenovich Vygotsky e Paulo Reglus Freire têm sua colaboração à área descritos a seguir:

Vygotsky Lev Semenovich Vygotsky (1896-1934) é apontado como um dos primeiros autores a sugerir que a cultura é parte formadora das funções psicológicas dos indivíduos, e de acordo com Moreira (2009), sua premissa principal é de que o desenvolvimento cognitivo não pode ser entendido sem referência ao contexto no qual este ocorre. Desta forma, Vygotsky defende que os processos mentais superiores do indivíduo têm origem nos processos sociais nos quais este está envolvido, e que o desenvolvimento depende do seu contexto social, histórico e cultural.

Portanto, este autor acredita que o desenvolvimento cognitivo é a conversão de relações sociais em funções mentais, não sendo, de acordo com Moreira (2009), "através do desenvolvimento cognitivo que o indivíduo torna-se capaz de socializar, mas através da socialização que se dá o desenvolvimento dos processos mentais superiores". É importante ressaltar que as teorias propostas por Vygotsky sofreram influência do período em que se deu sua vida na antiga União Soviética, o que contribuiu para suas raízes marxistas (Prass, 2012).

De acordo com Moreira (2009), outra ideia base da teoria vygotskyana é de que os processos mentais são mediados por instrumentos e signos, e que esta mediação é responsável pela conversão no indivíduo das relações sociais em funções psicológicas, por meio da da internalização. Moreira (2009) conclui que o que diferencia Vygotsky dos teóricos cognitivistas é, portanto, o foco na interação social que se dá entre o indivíduo e seu contexto. Esta interação possibilita o intercâmbio de significados no que se refere aos signos (como palavras e gestos), os quais são construções sociais. Ou seja, a interação social possibilita que por meio da captação dos significados já compartilhados socialmente, a internalização nos indivíduos dos existentes signos se faça possível.

Assim, a capacidade de operações psicológicas de um indivíduo aumenta e se modifica na medida em que este explora os símbolos existentes e seus sistemas. Estas novas funções psicológicas poderão ser aplicadas em cada vez mais atividades uma vez que este indivíduo aprenda um maior número de instrumentos. A linguagem é considerada um sistema articulado de signos e obtém, de acordo com Prass (2012), destaque importante nas ideias de Vygotsky, o qual a considera fundamental para o desenvolvimento cognitivo. Além de ser a principal via de transmissão da cultura e o veículo principal do pensamento, a linguagem permite a descontextualização, ou seja, liberação de vínculos contextuais imediatos. Desta forma, ao se distanciar de um contexto concreto e dominar a linguagem abstrata, a flexibi-

lização do pensamento conceitual e proposital é permitida (Moreira, 2009).

Freire Paulo Reglus Neves Freire (1921-1997) foi, de acordo com Prass (2012), um dos mais influentes educadores brasileiros. Seus projetos inicialmente tinham, em maioria, eixo na educação/alfabetização de adultos no final da década de 50. Ostermann et al. (2010) enfatiza a forte relação existente entre o método de Paulo Freire e a transformação social descrevendo que o método freireano está vinculado com a transformação total da sociedade. De acordo com Ostermann et al. (2010), Freire defendia a existência de uma cultura popular em que as vivências e conhecimentos cotidianos acompanham os estudantes, sendo assim, a conscientização desses estudantes busca uma transformação social através de uma postura crítica.

O raciocínio de Paulo Freire pautava-se, de acordo com Ostermann et al. (2010), no questionamento de que se era possível promover um pensamento crítico e uma transformação social independente da alfabetização, e se poderia engajar alfabetizados como sujeitos desse processo e não como objetos dele, propondo trabalhar com uma hierarquia horizontal entre educador e educando. Esse questionamento, de acordo com Ostermann et al. (2010) permeou toda a obra de Freire, onde de acordo com o autor:

> "Ao contrário da forma tradicional de ensino, muito centrada na autoridade de um professor, a forma horizontal em que alunos e professor aprendem juntos com intensa interação, se mostrou bastante mais eficiente. Convém salientar que, quando se fala hierarquia horizontal, não está se eliminando a hierarquia professor-aluno. Apenas ele se estabelece de forma totalmente distinta da tradicional. A hierarquia horizontal pressupõe uma participação igualitária do professor e do aluno no processo de aprendizagem." (Ostermann et al., 2010, p. 30)

Segundo Ostermann et al. (2010), Freire possui uma singularidade em seu trabalho, justificada pelo fato de considerar a educação como libertadora, e essa consideração é fundamentada na obra de Paulo Freire, *Pedagogia do Oprimido* (1968), em que fundamenta que o educador deve instigar o debate, desafiando os estudantes com questionamentos em torno das palavras geradoras, deve atentar-se às adversidades e dificuldades dos estudantes esclarecendo as dúvidas por meio de discussões com o apoio de recursos educacionais, como vários recursos didáticos tais quais projetor de transparências ou slides e postêres, onde sempre era destacada a palavra geradora.

3.2 Ferramentas de Programação

Os desdobramentos das tecnologias trouxeram reflexos para várias áreas, dentre elas a educação, repensando assim o modo de ensino. A viabilização destas mudanças se deu, principalmente, pela contribuição das ferramentas computacionais emergentes juntamente com os desenvolvimentos mais recentes das teorias de aprendizagem (Fiolhais and Trindade, 2003). Portanto, ainda de acordo com Fiolhais and Trindade (2003), desde muito cedo busca-se o uso pedagógico do computador para auxiliar no modo de como os estudantes aprendem.

Os grupos de ferramentas computacionais que vêm sendo discutidos no âmbito de teorias de aprendizagem são os de desenvolvimento e ensino de programação. No contexto de desenvolvimento, Pears et al., (2007) afirma que a maioria dos programadores já experientes prefere trabalhar com um Ambiente de Desenvolvimento Integrado (*Integrated Development Environment - IDE*) que aumente a sua produtividade ao proporcionar acesso a capacidades adicionais, como:

- Organização dos componentes do programa e recursos do projeto;
- Recursos linguísticos avançados de edição; e
- Integração de ferramentas com suporte adicional, como *debuggers*, ferramentas de teste, geradores de documentação, e analisadores de código.

No entanto, apesar de essenciais e amplamente utilizadas por programadores, estas ferramentas de programação apresentam grau elevado de complexidade e quantidade de recursos, e são geralmente desenvolvidas para ir ao encontro das necessidades dos programadores profissionais, apresentando grandes conjuntos de conceitos e características que vêm a ser problemáticos para os novatos (Pears et al., 2007). Isto pode acabar por oferecer um efeito contrário quando num contexto de cursos introdutórios de programação, fazendo com que estudantes tenham que dedicar um tempo excessivo ao aprendizado daquela ferramenta.

Portanto, para tentar solucionar esta situação, vêm sendo desenvolvidas ferramentas projetadas especificamente para as necessidades dos programadores iniciantes. Outra motivação para o desenvolvimento destas ferramentas tem sido o intuito de reduzir ou simplificar o sobre-trabalho do professor, por meio de recursos como a avaliação automática, detecção de plágio e gerenciamento de curso (Pears et al., 2007). Este desenvolvimento de novas ferramentas para novatos em programação fez com que estas passem a se classificar em duas categorias de softwares: conducionistas (também classificados como *programming support tools*) e construtivistas (ou *microworlds*) (Fontes, 2014).

Basicamente, as conducionistas são geralmente baseados em ferramentas de suporte à programação, as quais dão suporte aos aprendizes na criação de programas dentro de um ambiente de execução padrão, enquanto os construtivistas fornecem ambientes baseados em

metáforas físicas, no qual a execução do programa é refletida no estado visual de objetos concretos dentro do ambiente (Pears et al., 2007).

3.2.1 Ferramentas de Programação Conducionistas

As ferramentas de programação conducionistas apresentam, de maneira geral, sequências instrutivas fixas, nas quais os exercícios e práticas são realizados em ordem crescente de complexidade, com o objetivo de ascensão de níveis do saber (Fontes, 2014). Estas ferramentas fazem com que cada passo seja contruído por uma unidade limitada de saber, se mostrando eficiente no ensino e aprendizagem de operações geralmente menos complexas e sujeitas a mecanização.

Em geral, estas ferramentas oferecem um subconjunto limitado das capacidades de um IDE profissional, sendo que em alguns casos o IDE é modificado visando esconder as suas mais avançadas ferramentas, como é o caso do GILD, NetBeans BlueJ Edition (Pears et al., 2007). Em outros casos, certas ferramentas como o BlueJ, jGRASP, DrJava, JPie são projetadas para dar suporte ao aprendizado de conceitos de programação mais específicos.

Pears et al., (2007) salienta que a maioria destas ferramentas tem largamente utilizado a linguagem Java como preferencial, apontando assim uma tendência mundial e maior aceitação da mesma, ao mesmo tempo em que pode corresponder à necessidade destes aprendizes em obter maior apoio em seu aprendizado.

A tabela 2 apresenta a lista de ferramentas conducionistas levantadas neste trabalho.

Tabela 2: Ferramentas de Programação Conducionistas

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
Dr Java	Java	- Painel de Interação: pode-se inserir expressões e declarações Java e imediatamente ver seus resultados; - Painel de Definição: pode-se criar e editar definições de classe com suporte a diversos recursos.	- Windows, - MAC OS X (Apenas para Apple Java 6 JDK)

Continua na próxima página

Tabela 2 – *Continuada da página anterior*

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
BlueJ	Java	- Diagrama de Classes UML: para visualização e edição da estrutura de aplicação; - Editor de Texto: permite ao usuário ler e editar o código-fonte de uma classe.	- Windows, - MAC OS X, - Ubuntu/ Debian Linux, - Raspian Linux
jGRASP	Java, C, C++, Python, ADA, VHDL	- Diagrama de Estrutura de Controle (Controle Structure Diagram - CSD): para a visualização de código-fonte; - Diagramas de Classe UML para visualização estrutural do programa; - Visualizadores Dinâmicos e Visualizador Canvas: oferecem visualizações em tempo de execução para tipos primitivos e objetos, incluindo estruturas de dados tradicionais, tais como listas ligadas e árvores binárias.	- Windows, - MAC OS X, - Linux/ UNIX
JPie	Java	- Manipulação direta dos elementos do programa: não é necessário escrever códigos mas sim manipular diretamente seus componentes funcionais - cápsulas que representam variáveis, métodos e outras unidades de programa JPie.	- Windows, - MAC OS X

Continua na próxima página

Tabela 2 – *Continuada da página anterior*

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
VisuAlg	Portugol	- Se limita ao ensino de principios básicos da programação estruturada, além de também ser possível simular o que acontece na tela do computador com o uso dos comandos 'leia' e 'escreva'. Possibilidade de verificação dos valores das variáveis, o acompanhamento passo a passo da execução de um algoritmo, e o suporte ao modo simples de depuração.	- Windows
TutorICC	Pascal	- Conteúdo dividido em níveis de dificuldade, permitindo estabelecer o caminho mais conveniente para o aprendizado. Provas que são corrigidas automaticamente por um recurso do programa, que então recomenda ao estudante o melhor caminho a seguir dentro do conteúdo da disciplina; - Visualizador gráfico do programa: linhas do código são percorridas mostrando qual comando está sendo executado no momento.	- Windows

DrJava DrJava é um ambiente de desenvolvimento leve para escrever programas em Java. Ele é projetado principalmente para estudantes, fornecendo uma interface intuitiva e a capacidade de avaliar de forma interativa código Java. Ele também inclui recursos para usuários mais avançados, estando disponível gratuitamente sob a licença BSD, além de estar em constante desenvolvimento pelo grupo JavaPLT na Universidade Rice (Foundation et al., 2015).

A intenção da criação do DrJava foi a de introduzir aos estudantes os mecanismos de escrita de programas em Java e alavancar o entendimento da linguagem em si, para então

ser possível proporcionar um suporte adequado no desenvolvimento de programas (Allen et al., 2012). Ao desenvolver esta ferramenta, a intenção foi a de suportar uma interface de programação transparente projetada para minimizar o chamado "fator de intimidação" que estudantes iniciantes experienciam quando confrontados com o desafio de se escrever um código. Conforme Allen et al., (2012), a interface transparente consiste em uma janela com dois painéis:

- Um painel de interação, em que o estudante pode inserir expressões e declarações Java e imediatamente ver seus resultados; e
- Um painel de definição, em que o estudante pode criar e editar definições de classe com suporte a recursos como coincidência de chaves, destaque de sintaxe, dentre outros.

Com o uso do DrJava, programadores iniciantes podem escrever programas em Java sem se confrontar com complicações como *input/output* de texto, interface de linha de comando, variáveis de ambiente como classpath dentre outras complexidades das interfaces de projeto suportadas pelo ambiente de desenvolvimento comercial do Java (Allen et al., 2012).

Em tutorial apresentado em Foundation et al., (2015), descreve-se a mais distinta e principal característica do DrJava, chamada de painel de interação (*Interaction Pane*). Este painel oferece para ambos, iniciantes e mais experientes programadores, a habilidade de rapidamente testar o código sem ter que escrever complexos métodos principais, conforme a Figura 5.

Uma maneira de utilizar os painéis interativos é inserindo expressões simples como 1+1, de maneira que estas serão interpretadas e avaliadas e o resultado será mostrado, de acordo com a Figura 6.

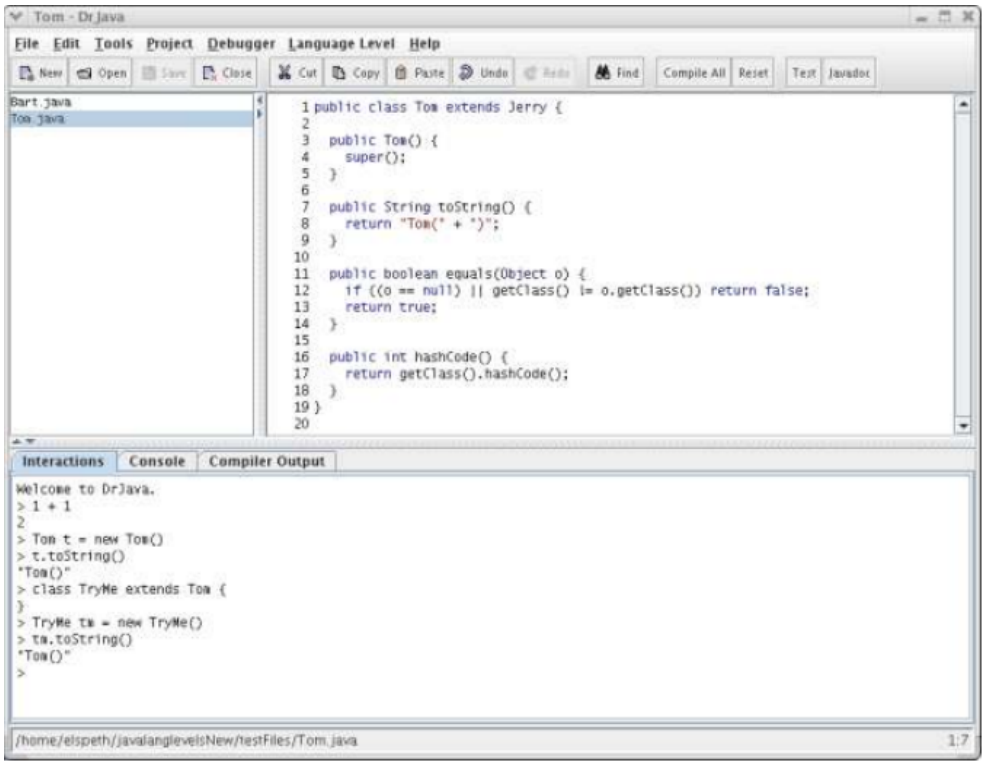


Figura 5: Painel Interativo e Painel de Definição do DrJava. Fonte: (Foundation et al., 2015)

Figura 6: Painel de Interação, resultado da operação 1+1. Fonte: (Foundation et al., 2015)

Também é possível instanciar classes e chamar seus métodos por meio do painel de interação, conforme a figura 7.

Figura 7: Painel de Interação, criação da instância de uma classe e a chamada de seu método. Fonte: (Foundation et al., 2015)

De acordo com Allen et al., (2012), esta capacidade de interação com os componentes do programa enquanto o mesmo está sendo desenvolvido permite que o programador acesse

seus componentes sem a necessidade de recompilá-lo. Desta maneira, é particularmente útil no contexto de cursos introdutórios de programação, pois permite que o estudante conduza experimentos simples para determinar como construções linguísticas se comportam, e também determinar os resultados de subcomputações dentro de um programa (Allen et al., 2012).

Outra função principal do DrJava é a de detectar com exatidão erros básicos de sintaxe assim que são cometidos, além de destaque de palavras-chave e verificação de chaves.

Os desenvolvedores desta ferramenta citam como trabalhos relacionados os programas DrScheme - o qual veio a ser utilizado pela Universidade de Rice e também serviu como modelo para o desenvolvimento do DrJava, e o programa BlueJ, o qual se diferencia por descrever programas utilizando ambos diagramas UML e telas de codificação, o que faz deste um ambiente mais complexo do que DrJava (Foundation et al., 2015).

BlueJ BlueJ é um Ambiente de Desenvolvimento Integrado para o ensino de programação orientada a objetos. Ele foi especialmente projetado para o ensino introdutório de programação. Seu surgimento se deu a partir de três percepções (of Kent, 2012):

- O meio ambiente não é orientado a objetos: o ambiente deveria refletir o paradigma da linguagem, sendo que estudantes normalmente interagem com linhas de código-fonte, e não objetos;
- O ambiente é muito complexo: certos ambientes são desenvolvidos para programadores profissionais, e apresentam uma sobrecarga de componentes de interface e funcionalidades; e
- O ambiente concentra-se na interfaces do usuário: muitos ambientes se concentram em construir a interface gráfica do usuário (GUI) em vez do pensamento sobre a construção da própria aplicação.

Quando um projeto Java é aberto no BlueJ, a janela principal mostra o diagrama de classes da Linguagem Unificada de Modelagem (*Unified Modeling Language - UML*), visualizando desta forma a estrutura de aplicação (Figura 8). Os usuários podem então interagir diretamente com classes e objetos. O ícone da classe pode ser usado para executar um construtor, o que resulta em um objeto a ser criado e colocado na bancada de objetos na parte inferior da janela principal. Uma vez que o objeto tenha sido criado, qualquer método público pode ser executado (Quig et al., 2003).

Um clique duplo no ícone de uma classe abre um texto editor que permite aos usuários ler ou editar o código-fonte de uma classe. Um clique no botão "Compilar"irá recompilar todas as classes alteradas e a execução pode começar novamente. Os erros de compilação são exibidos diretamente no editor sendo destacados na linha correspondente, mostrando também o texto da mensagem de erro (Quig et al., 2003).

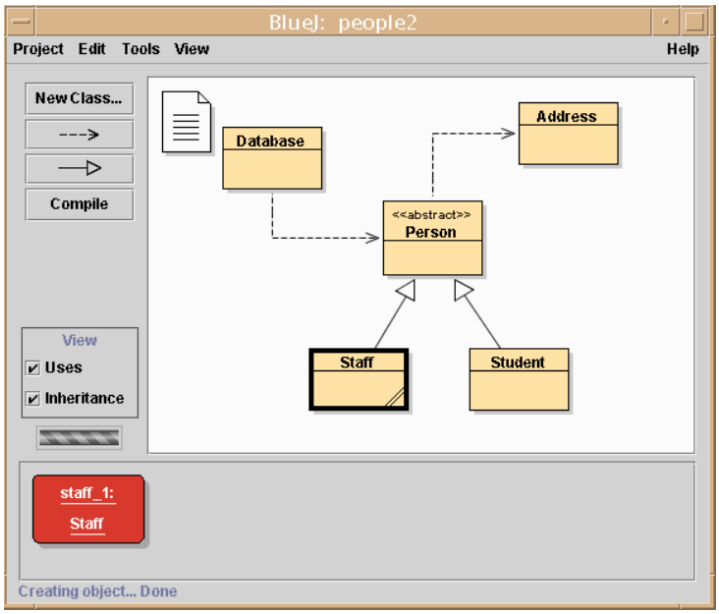


Figura 8: Janela principal do BlueJ. Fonte:(Quig et al., 2003)

Tanto a criação de objetos de uma classe quanto a invocação de seus métodos é feita por meio de cliques no ícone da classe ou dos objetos.

Uma das características centrais do ambiente BlueJ é a estrutura de classe mostrada na janela principal, o que força o estudante a reconhecer e pensar sobre a estrutura do programa desde a primeira vez que o mesmo interage com um programa Java. Isto faz com que desde o princípio o estudante perceba claramente que uma aplicação é um conjunto de classes cooperantes, ficando também claros os relacionamentos entre as classes e os objetos. Sem a necessidade de muita explicação, o estudante pode claramente perceber que uma classe é utilizada para criar objetos e que os mesmos contém dados, além de que os objetos são manipulados através da invocação de operações que alteram seu estado. Desta maneira, a visualização se faz a principal vantagem do BlueJ, ajudando o estudante a visualizar as importantes entidades abstratas da orientação a objetos e permitindo a interação entre elas.

jGRASP jGRASP é um ambiente leve de desenvolvimento integrado (IDE), criado pelo grupo de pesquisa da Universidade de Auburn especificamente para fornecer visualizações para melhorar a compreensão de software. jGRASP é implementado em Java, e, portanto, funciona em todas as plataformas com uma máquina virtual Java. Além de suportar Java, também suporta as linguagens de programação: C, C ++, Objective-C, Python, Ada, e VHDL (Auburn, 2015).

Este IDE vem se destacando por fornecer a geração automática de três visualizações de software importantes:

- Diagrama de Estrutura de Controle (*Controle Structure Diagram - CSD*) para a visualização de código-fonte;
- Diagramas de Classe UML para visualização estrutural do programa; e
- Visualizadores Dinâmicos e Visualizador Canvas que oferecem visualizações em tempo de execução para tipos primitivos e objetos, incluindo estruturas de dados tradicionais, tais como listas ligadas e árvores binárias.

De acordo com Auburn (2015), o ambiente jGRASP disponibiliza o Diagrama de Estrutura de Controle (CSD) nas linguagens C, C ++, Objective-C, Java, VHDL, e Python, e tem como principal objetivo melhorar a compreensão de código-fonte, descrevendo claramente as construções de controle, caminhos de controle, e também a estrutura geral de cada unidade de programa. A janela CSD em jGRASP (conforme Figura 9) fornece suporte completo para geração CSD, bem como a edição, compilação, execução, e depuração de programas. Além disso, depois de editar o código-fonte, a regeneração do diagrama é rápida, eficiente e sem interrupções.

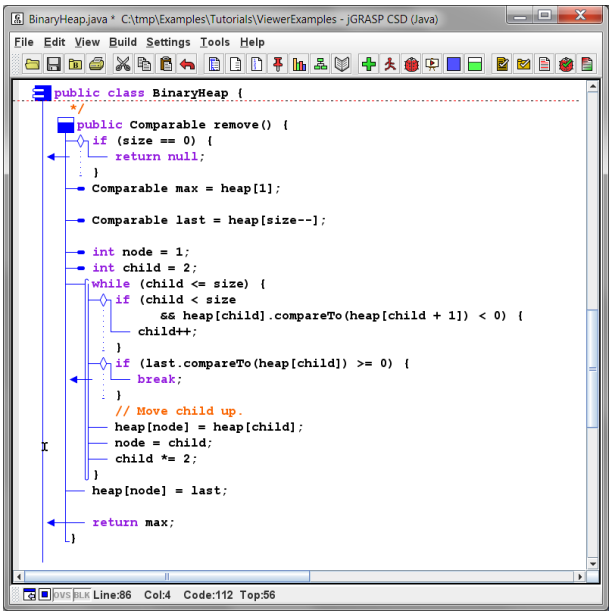


Figura 9: Exemplo de CSD no jGRASP. Fonte: Fonte (Auburn, 2015)

O jGRASP também disponibiliza a visualização de diagramas de classes UML para Java, de acordo com a Figura 10. Além de mostrar as relações de classe, o diagrama também tem recursos interativos úteis para rastrear e entender as dependências de classes.

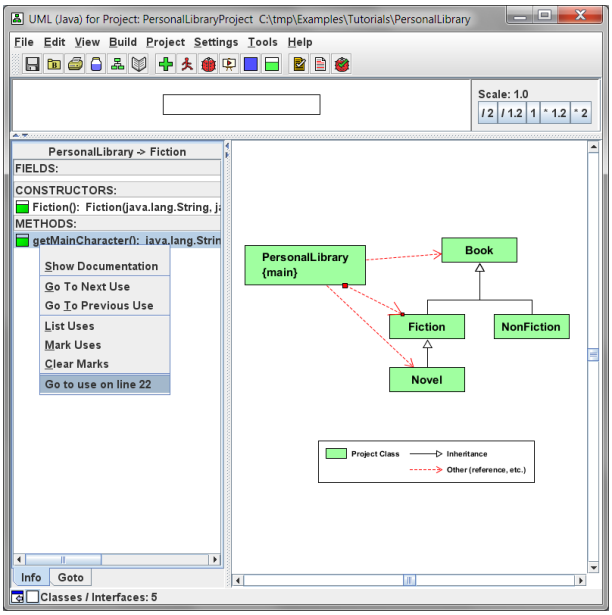


Figura 10: Exemplo de Diagrama de Classes UML no jGRASP. Fonte: (Auburn, 2015)

Como citado por Pears et al., (2007) o grande diferencial do jGRASP quanto a outras ferramentas de visualização é que este, além de oferecer visualizações estáticas, oferece também visualizações dinâmicas. Os visualizadores dinâmicos para objetos e tipos primitivos de dados fornecem visualizações enquanto o usuário percorre um programa no modo de depuração ou quando chama os métodos de um objeto no *workbench*. Quando um visualizador é aberto durante a depuração ou no *workbench*, um identificador de estrutura tenta reconhecer automaticamente as listas ligadas, árvores binárias, tabelas hash e *arrays* diversos como listas, pilhas, filas, etc. Quando uma identificação de alguma destas estruturas é feita, uma visualização do objeto é apresentada (Figuras 11 e 12).

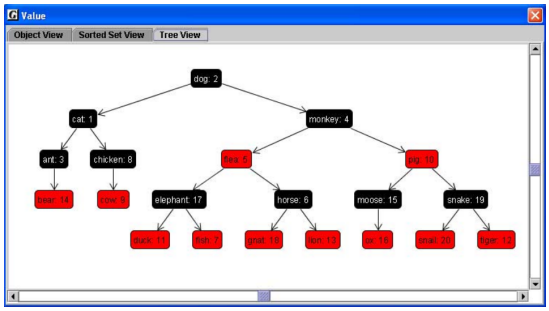


Figura 11: Exemplo de uma visualização de uma árvore binária no jGRASP. Fonte: (Hendrix et al., 2004).

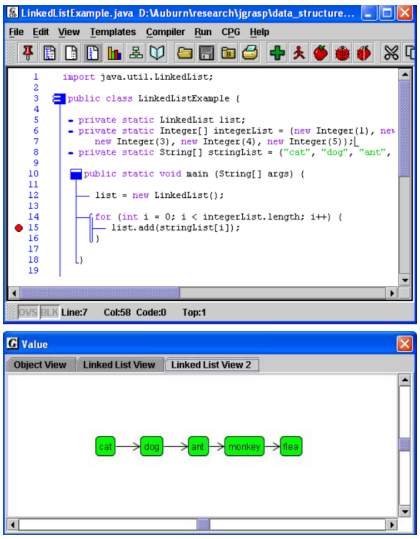


Figura 12: Exemplo de uma lista ligada no jGRASP. Fonte: (Hendrix et al., 2004).

Outro diferencial do jGRASP é a possibilidade de visualizações dinâmicas sincronizadas, como ilustra a Figura 13.

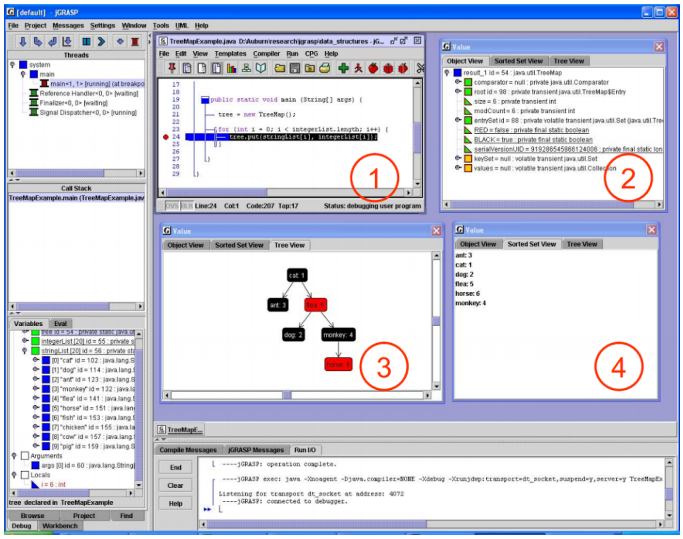


Figura 13: Exemplo de visualizações dinâmicas sincronizadas no jGRASP Fonte: (Hendrix et al., 2004).

JPie O ambiente de programação interativo JPie foi desenvolvido com a intenção de empoderar e tornar mais acessível o uso de softwares a um público mais vasto. Por meio deste, todos os aspectos do comportamento de um programa podem ser modificados enquanto o mesmo é executado, sem a necessidade do tradicional ciclo de edição-compilação-teste que a maioria dos ambientes de programação possuem (in St. Louis, 2015). As mudanças efetuadas afetam os outros componentes, portanto, se uma classe é modificada seus objetos também serão.

Portanto, o IDE JPie oferece a facilidade de manipulação direta dos elementos do programa, não sendo necessário escrever códigos mas sim manipular diretamente seus componentes funcionais. Isso gera um fator motivacional ao permitir que programadores inexperientes alcancem sucesso precoce sem a necessidade de passar pelo completo e difícil aprendizado textual que programar exige.

No JPie, o programa é construído por meio de cápsulas que representam variáveis, métodos e outras unidades de programa JPie, conforme Figura 14.

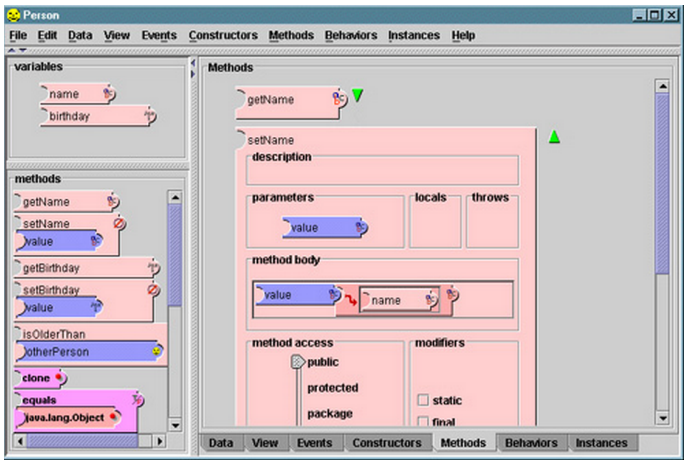


Figura 14: Exemplo de cápsulas no JPie. Fonte: (in St. Louis, 2015)

VisuaAlg O VisuAlg é uma ferramenta de apoio à programação desenvolvida pelo brasileiro Cláudio Morgado de Souza. Segundo nota do próprio criador (Tonet, 2007), o desenvolvimento do VisuAlg se deu a partir da necessidade dos estudantes iniciantes exercitarem seus conhecimentos em um ambiente que se apresentasse mais próximo à realidade dos mesmos, visto que as linguagens de programação mais utilizadas são escritas na língua inglesa.

Desta maneira, o VisuAlg surgiu ao se defrontar com o cenário da época, que exigia a escolha entre a utilização do papel no aprendizado das técnicas de elaboração de algoritmos, ou a submissão dos iniciantes às linguagens robustas de programação como o Pascal ou C. Souza então desenvolveu um ambiente baseado em uma linguagem que se assemelhava com o "Portugol"[5], que apresentava na época grande popularidade nos meios acadêmicos e livros que tratavam do assunto (Tonet, 2007).

Por meio do VisuAlg, Figura 15, é possível ensinar principios básicos da programação estruturada de maneira simples, além de também ser possível simular o que acontece na tela do computador com o uso dos comandos 'leia' e 'escreva' (Tonet, 2007). Outra característica é a verificação dos valores das variáveis, o acompanhamento passo a passo da execução de um algoritmo, e o suporte ao modo simples de depuração.

O VisuAlg não depende de DLLs (*Dynamic Link Libraries*) e outros componentes, sendo desta forma considerado um programa simples, o qual apresenta uma instalação rápida que exige cerca de 1 MB de espaço em disco. Pode ser executado sob Windows 95 ou posterior, e tem melhor aparência com resolução de vídeo de 800x600 ou maior (Tonet, 2007).

TutorICC O TutorICC é um ambiente interativo e adaptável, desenvolvido e utilizado para o ensino de programação em Pascal na Universidade de Brasília. Segundo Piccolo et al., (2010), este ambiente foi utilizado em um contexto de ensino semipresencial, no qual o conteúdo

[5]O Portugol é uma pseudo-linguagem algorítmica utilizada na descrição de algoritmos, destacando-se pelo uso de comandos em português. Isto facilita o aprendizado da lógica de programação, habituando o iniciante com o formalismo de programação.

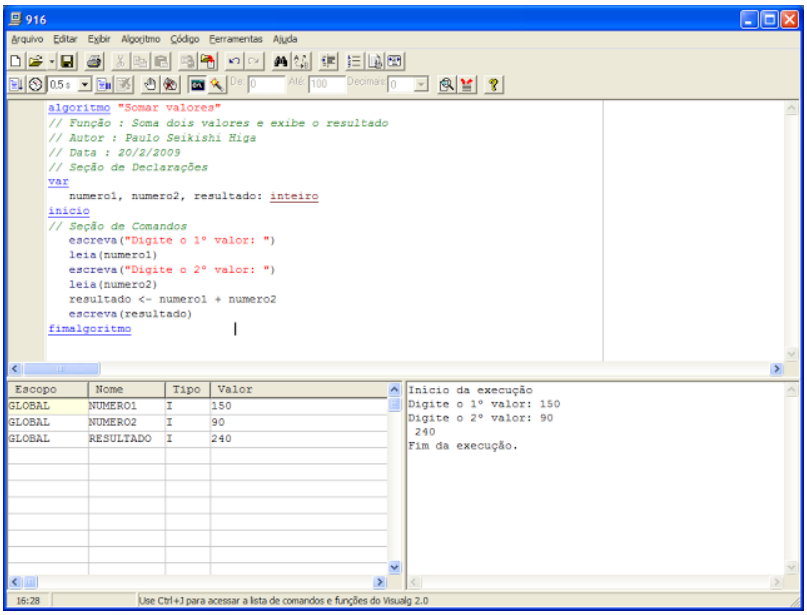


Figura 15: Tela Inicial do VisuAlg. Fonte: (Tonet, 2007)

está dividido em níveis de dificuldade, permitindo assim ao próprio estudante estabelecer o caminho mais conveniente para o aprendizado, dentro do conteúdo da disciplina. Durante o curso, os estudantes fazem provas que são corrigidas automaticamente por um recurso do programa, que então recomenda ao estudante o melhor caminho a seguir dentro do conteúdo da disciplina.

O principal recurso do TutorICC é o visualizador gráfico do programa, no qual as linhas do código são percorridas mostrando qual comando está sendo executado no momento, conforme a Figura 16.

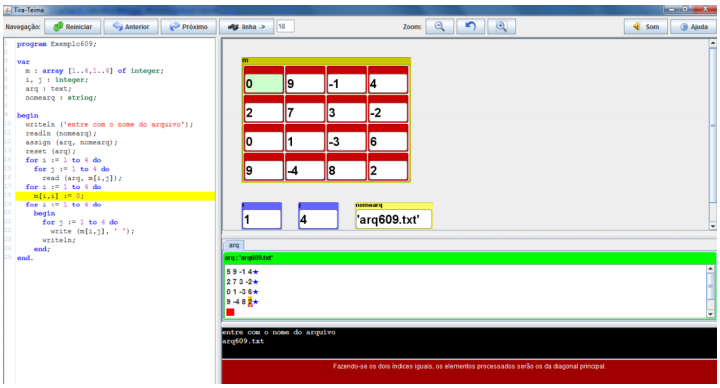


Figura 16: Visualizador Gráfico do TutorICC.Fonte: (Píccolo et al., 2010)

3.2.2 Ferramentas de Programação Construtivistas

Estas ferramentas são também os chamados "micromundos informáticos", e possibilitam a expressão e a exploração individualizada por parte do aprendiz. Isso faz com que estes possam desenvolver aspectos específicos na aprendizagem, exercitando suas capacidades cognitivas e levando em conta os conhecimentos prévios do estudante, bem como seus interesses, expectativas, e ritmos de aprendizagem (Fontes, 2014).

A metáfora física de um ambiente micromundo destina-se a diminuir a distância entre os modelos mentais dos estudantes e a linguagem de programação (Pears et al., 2007). O "mundo"vem a ser uma visualização em uma tela de computador ou um ambiente físico real. Micromundos podem variar entre si, diferindo principalmente nas linguagens de programação que cada um usa, a natureza e extensão de suas metáforas, e o nível de suporte à programação que eles oferecem (Pears et al., 2007).

A tabela 3 apresenta a lista de ferramentas construtivistas levantadas neste trabalho.

Tabela 3: Ferramentas de Programação Construtivistas

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
Karel the Robot	Karel (Baseado em Pascal)	- Cursor programável que pode se mover através de um mundo representado por uma grade. É possível programar o Karel para realizar tarefas, de maneira a localizar, transportar ou alocar sinalizadores em algum lugar específico da grade, sendo também possível criar barreiras nas ruas e avenidas de maneira que estas sejam intransponíveis.	-Todos exceto Macintosh OS 9
Alice	Java, Alice (Baseado em Python)	- Programação de animações em 3D, suportando todas as estruturas de programação e permitindo que os estudantes visualizem imediatamente como o seu código de programa é executado.	- Windows - MAC OS X - Linux

Continua na próxima página

Tabela 3 – *Continuada da página anterior*

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
GreenFoot	Java	- Painel de Visualização: visualiza objetos, seus estados e comportamento; - Painel de Interação: tela principal, em que é possível interagir com os objetos. Baseado em: Karel the Robot, BlueJ.	- Windows - MAC OS X - Ubuntu/ Debian
Scratch	Logo	- Manipulação de blocos por meio de quatro painéis: o da esquerda mostra os comandos disponíveis, o do meio mostra o script com os comandos selecionados para o objeto em questão, o da direita mostra o palco em que as ações acontecem e abaixo os objetos disponíveis. Baseado em: Logo Turtle	- Windows - MAC OS X - Ubuntu/ Debian
Jeroo	Java, C, C++	- Painel de Edição: possibilita editar o código-fonte, em que o animal pode se locomover e realizar tarefas; - Painel de Visualização: permite a visualização gráfica imediata do animal na ilha. Baseado em: Karel the Robot.	- Windows - Mac OS X - Linux

Continua na próxima página

Tabela 3 – *Continuada da página anterior*

Nome	**Linguagem**	**Funcionamento**	**Sistema Operacional**
Robocode	Java, C, Scala	- A codificação da batalha de tanques robô para competir contra outros robôs em uma arena de batalha se dá pelas áreas: 1, em que são definidos e setados os valores das variáveis; 2, o código que irá rodar apenas uma vez por instância de robô; 3, em que é programado uma ação repetitiva do robô através de um laço de repetição e 4, em que são adicionados os métodos os quais o robô pode vir a utilizar, ou onde se pode tratar eventos.	- Windows - Linux
Pygame	Python	- Painel de Interação: feedback imediato, pode-se inserir expressões e declarações e imediatamente ver seus resultados; - Editor de Código: sintaxe simples e flexível, módulos fáceis de usar e o requerimento de uma indentação apropriada - tela de codificação pode instanciar uma nova janela, em que as animações acontecem.	- Windows - Mac OS X - Linux

Karel the Robot Karel the Robot ensina os conceitos e habilidades de programação fundamentais, introduzindo as idéias centrais de maneira descontraída (Pattis, 2015). Karel é basicamente um cursor programável que pode se mover através de um mundo representado por uma grade (*gridwork*) em uma tela de linhas verticais e horizontais (avenidas e ruas) que formam intersecções ou esquinas da grade. O mesmo é restrito a se mover de uma esquina à outra, executando um movimento por vez. Além disso, ele pode girar 90 graus para a esquerda quando requisitado, podendo também se encontrar nas faces Norte, Sul, Leste e

Oeste.

Para variar o ambiente do Karel, foram adicionados sinalizadores (*beepers*), os quais piscam e que Karel pode detectar/ouvir, além de poder apanhar, carregar, e largar o mesmo (Untch, 1999).

É possível programar o Karel para localizar, transportar ou alocar esses sinalizadores em algum lugar específico da grade, sendo também possível criar barreiras nas ruas e avenidas de maneira que estas sejam intransponíveis.

As instruções primivitas de Karel são: mover (*move*), virar à esquerda (*turnleft*), apanhar sinalizador (*pickbeeper*), largar sinalizador (*putbeeper*) e desligar (*turnoff*) (Untch, 1999). Todas as instruções são mostradas graficamente quando executadas. Além disso, novas instruções podem ser definidas para estender o vocabulário de Karel. Como exemplo, pode-se definir a instrução de virar à direita como na Figura 17:

```
1   DEFINE-NEW-INSTRUCTION virarDireita AS
2
3    BEGIN
4   begin{quote}
5           turnleft;
6           turnleft;
7           turnleft;
8   end{quote}
9   END
10
```

Figura 17: Instruções no vocabulário de Karel

Também existem um conjunto de predicados que podem ser testados por Karel como bloqueios no caminho, proximidade do beeper, direção da face e sinalizadores na mala através dos comandos:

(front/left/right$_i s_b locked$), ($next/not_n ext\_to\_a\_beeper$),
(facing/not_facing_ north/south/east/west),
(any/no_beepers_in_beeper_bag). (Untch, 1999)

Na construção das instruções é possível utilizar as declarações condicionais *if/else*, além das de iteração *iterate/while/do_while*.

Cada tarefa de Karel pode ser diferente dentro de uma situação que descreve o estado atual do mundo de Karel (onde são paredes e apitos, onde está Karel, quantos *bips* estão em sua mala, etc.). Um exemplo é apresentado por (Pattis, 2015) na Figura 18:

"Todas as manhãs, Karel é despertado na cama quando o jornal - representado por um sinal sonoro - é jogado na varanda da frente da casa. Programe Karel para recuperar o papel e trazê-lo de volta para a cama".

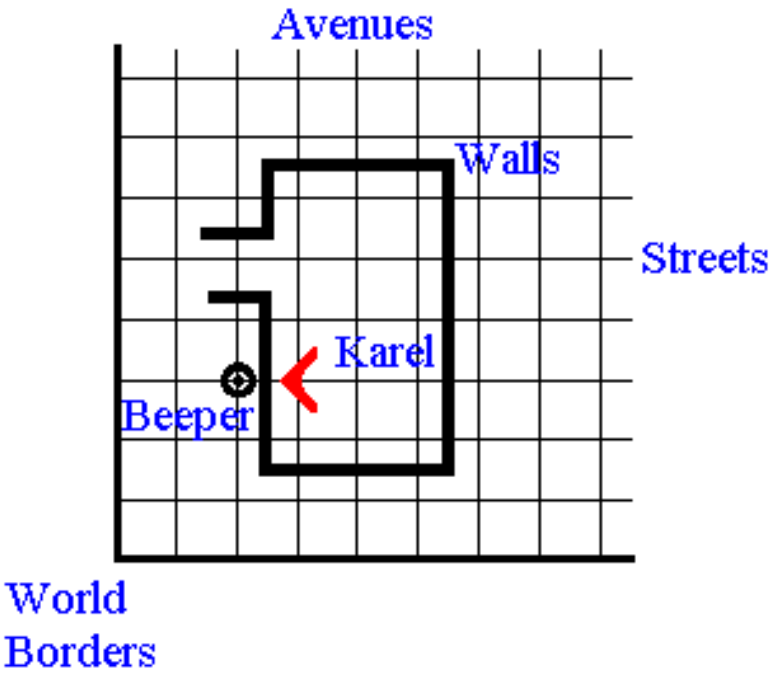


Figura 18: Representação da tarefa a ser cumprida por Karel the Robot. Fonte: (Pattis, 2015)

Uma nova versão do Karel foi desenvolvida para linguagem orientada a objetos, definida como Karel++ (Bergin et al., 2002).

Alice Alice é um ambiente de programação em 3D que visa a criação de animações para contar histórias por meio de um jogo interativo. Esta ferramenta é disponibilizada gratuitamente, e foi projetada com a intenção de ser o primeiro contato de um estudante com a programação orientada a objetos (Mellon, 2015). Ela permite que os estudantess aprendam conceitos fundamentais de programação no contexto da criação de filmes animados e videogames simples. Em Alice, objetos 3D como pessoas, animais, e veículos preenchem um mundo virtual no qual os estudantes criam programas para animar estes objetos.

De acordo com Cooper et al., (2003), a utilização de gráficos para o aprendizado de programação se faz interessante pois muitas vezes os estudantes são atraídos para a área da computação por conta de experiências anteriores com vídeos, jogos, filmes de animação, e multimídia. Entretanto, os mesmos se decepcionam ao perceber que seriam necessários alguns anos para conseguir desenvolver algo no mesmo nível. Desta maneira, Alice tem a intenção de satisfazer as expectativas iniciais destes estudantes cobrindo tópicos abstratos da programação de maneira mais atrativa, contribuindo assim para a motivação e persistência dos mesmos (Cooper et al., 2003).

Cooper et al., (2003) também afirma que a programação de animações em 3D é vantajosa ao permitir que os estudantes visualizem imediatamente como o seu código de programa é executado, obtendo um feedback altamente visual que faz com que o estudante consiga conectar as linhas individuais de código à ação executada na animação - o que leva a uma compreensão do funcionamento real das diferentes construções de linguagem de programação.

Alice foi originalmente construído em cima da linguagem de programação Python, sendo posteriormente implementada em Java. Entretanto não se faz necessário o conhecimento destas linguagens de programação, ou seja, programadores iniciantes podem facilmente programar no ambiente Alice sem conhecimento prévio destas linguagens (Cooper et al., 2003).

No Alice as decisões são expressas por meio da estrutura tradicional de *if/then/else*, obtendo um feedback visual imediato dos resultados desta instrução de decisão. Como exemplo apontado em Cooper et al., (2003), em uma cena em que o gato e os peixes estão perseguindo um ao outro na praia, se a distância entre o gato e o peixe é inferior a um metro, o estudante vê o gato saltar para cima e para baixo. Caso contrário, nada acontece, conforme Figura 38.

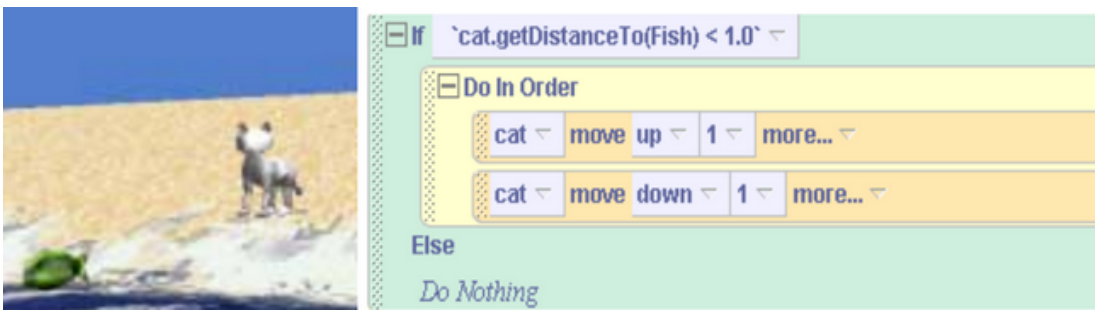


Figura 19: Exemplo de instrução if/else no Alice. Fonte: (Cooper et al., 2003)

Todas as estruturas tradicionais são possíveis, como *loops*, *while* e recursão. Cooper et al., (2003) apresenta um exemplo de recursão na Figura 20, no qual o método de caça (*chase*) é chamado recursivamente até que a distância do gato seja menor que 2.

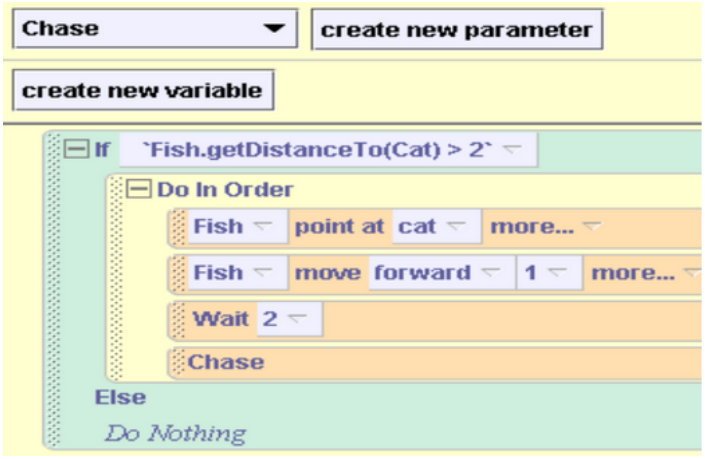


Figura 20: Exemplo de método recursivo no Alice. Fonte: (Cooper et al., 2003)

Alice também suporta a ocorrência e tratamento de eventos, assim como a execução de instruções concorrentes e o uso de coleções (como *arrays* e listas).

GreenFoot Greenfoot ensina a orientação a objetos para estudantes iniciantes através da linguagem Java, se assemelhando ao Alice ao criar atores/personagens que vivem em mundos para construir jogos, simulações e outros programas gráficos, sendo assim um programa visual e interativo. Ele se diferencia por ser em ambiente 2D e principalmente pelo fato dos atores serem programados no código Java textual padrão, o que, segundo Of Kent in Canterbury (2015) , proporciona uma experiência de programação que combina linguagem baseada em texto tradicional com uma execução visual (Figura 21).

O projeto Greenfoot foi originalmente inspirado na combinação de dois outros ambientes de ensino populares também apresentados neste trabalho: Karel the Robot e BlueJ. De acordo com Henriksen and Kolling (2003), a inspiração em Karel the Robot se deve ao fato deste possuir uma excelente visualização de objetos, seus estados e comportamento, faltando-lhe no entanto um meio para interação direta com os mesmos, enquanto BlueJ oferece esta interação de maneira direta porém carecendo de uma melhor visualização e detalhes dos objetos.

Apesar de baseado nestas duas ferramentas, Henriksen and Kolling (2003) afirma que o Greenfoot resultou em qualidades que vão além da combinação destas duas, pois permite trabalhar com uma classes mais gerais, incluindo efetivamente um grande número de estilos de problemas a serem simulados.

Como mostra a Figura 22, o mundo de objetos também se torna interativo com objetos programáveis:

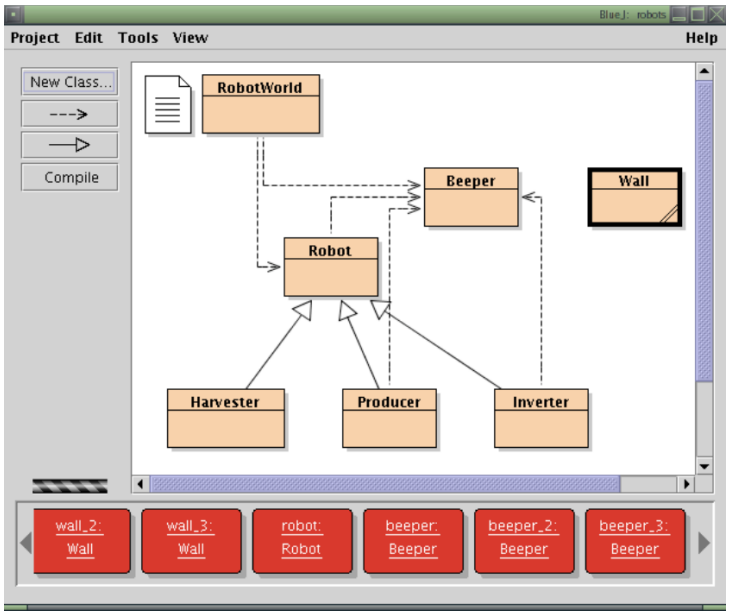


Figura 21: Visualização de objetos no Greenfoot. Fonte: (Henriksen and Kolling, 2003)

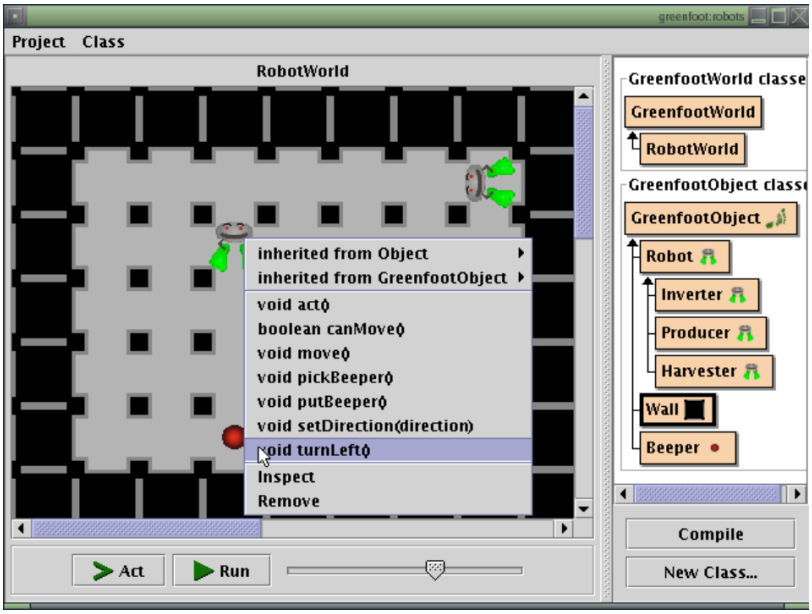


Figura 22: Tela principal do Greenfoot. Fonte: (Henriksen and Kolling, 2003)

Logo Em meados da década de 1960, Seymour Papert, um matemático que havia trabalhado com Jean Piaget em Genebra, mudou-se para os Estados Unidos, onde ele co-fundou o Laboratório de Inteligência Artificial do *Massachusetts Institute of Technology* (MIT) em pareceria com Marvin Minsky (Foundation, 2015b). A primeira versão do Logo foi criada então por Papert em trabalho em conjunto com sua equipe no MIT, em 1967.

Portanto, Logo é uma linguagem de programação, sendo considerada um dialeto da linguagem LISP, e seu ambiente de programação tem se desenvolvido nos últimos 48 anos

baseando-se na filosofia educacional construtivista. De acordo com Foundation (2015a), Logo foi desenvolvido para ser um ambiente no qual a aprendizagem piagetiana pudesse ocorrer e ser suportada, pois entende-se que os estudantes constroem o conhecimento ao interagir com as coisas e as pessoas existentes ao seu redor. Dentro desse suporte ao aprendizado construtivista, visa-se facilitar não apenas o aprendizado de programação de computadores, mas também de outros domínios, como matemática, linguagem, música, robótica, telecomunicações, e ciência.

A linguagem Logo é utilizada para desenvolver simulações e para criar apresentações de multimídia e jogos, sendo, de acordo com Foundation (2015b), projetado para ter um alto grau de flexibilidade. Isso significa que Logo é acessível aos novatos, incluindo crianças pequenas, e também suporta explorações complexas e projetos sofisticados por usuários experientes e de outras idades.

As características principais desta linguagem de programação são definidas em: Interatividade, modularidade, extensibilidade, e flexibilidade. A interatividade fornece ao usuário um *feedback* imediato sobre instruções individuais, auxiliando assim na depuração e no processo de aprendizagem. A modularidade e extensibilidade são definidas pelo fato de que os programas em Logo são uma coleção de pequenos procedimentos, os quais permitem construir projetos complexos através de passos pequenos. Desta forma, a programação em Logo é feita ao se adicionar novos procedimentos ao vocabulário existente, ensinando assim novas palavras ao se relacionar com as palavras que este já conhece, e sendo desta maneira um processo semelhante à forma como as pessoas aprendem a linguagem falada (Foundation, 2015d).

Quanto à flexibilidade da linguagem, esta se dá pela não necessidade de especificação exata do tipo de dados que o programador está usando, pois, de acordo com Foundation (2015d), esta definição rigorosa torna as coisas mas fáceis para o computador e não para as pessoas que desenvolvem, uma vez que as mesmas não têm o costume de pensar sobre certas definições exatas no seu dia-a-dia. Portanto, Logo foi pensado para facilitar a exploração da linguagem natural, pois suas estruturas de dados - palavras e listas - fazem um paralelo muito próximo das palavras, frases e sentenças que compõem a linguagem falada e escrita.

Inicialmente, a maioria dos ambientes Logo se baseavam na ideia central da tartaruga, sendo o programa Turtle o principal destes. A ideia era de que este animal pudesse ser direcionado para se movimentar, através de comandos digitados no computador, e de que além de se movimentar, também pudesse deixar "traços" por onde passasse, conforme Figura 23.

O processo de aprendizagem de Logo de uma nova palavra (ou procedimento), poderia se dar de forma simples conforme o exemplo do comando da Figura 24, no qual a palavra "quadrado" (*square*) é definida e aprendida.

Ao longo de mais de quatro décadas de evolução, Logo passou por diversas mudanças que

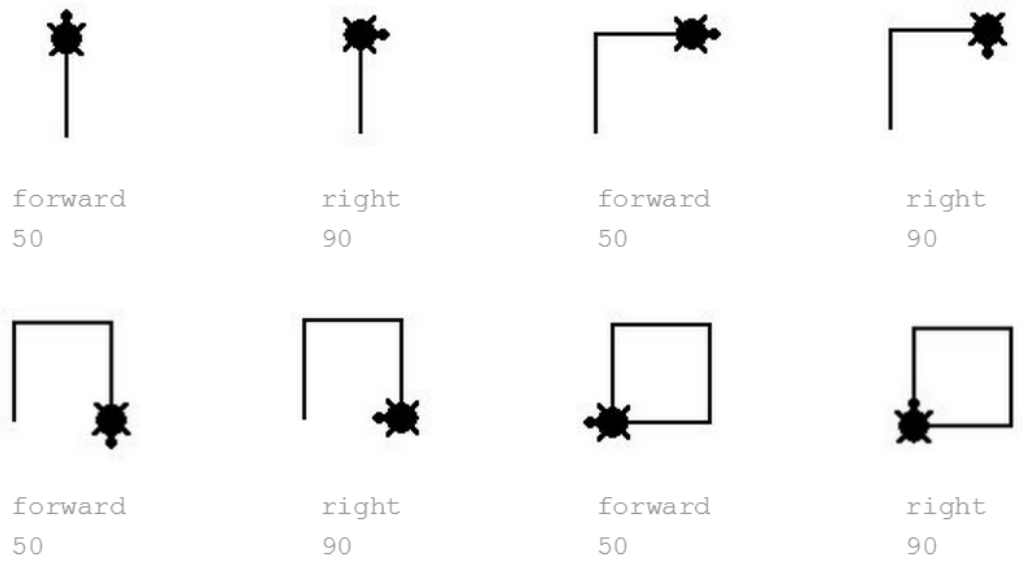


Figura 23: Desenho da forma de um quadrado através da tartaruga em Logo. Fonte: (Foundation, 2015c)

```
to square
repeat 4 [forward 50 right 90]
end
```

Figura 24: Definição do procedimento "quadrado". Fonte: (Foundation, 2015c)

visavam acompanhar o crescimento das tecnologias de computador, possuindo atualmente um grande conjunto e variedade de programas para o seu desenvolvimento. Turtle, que foi uma das primeiras figuras utilizadas nos ambientes para desenvolvimento da linguagem Logo, evoluiu juntamente com a computação gráfica para outros personagens em diferentes programas.

Um dos principais ambientes de programação desenvolvidos a partir de então foi o Scratch, que foi concebido em 2004 pelo laboratório de mídia do MIT, sendo adequado para projetar e construir histórias interativas, animações, jogos, música, e arte. Ele utiliza o paradigma de blocos de programação, que foi originalmente implementado como *Logo Blocks*.

Depois da popularidade obtida através do Scratch, os blocos de programação tornaram-se amplamente difundidos e passaram a ser utilizados em uma série de outras aplicações Logo, como Turtle Art, Scratch for Arduino, Snap!, e StarLogo TNG (Foundation, 2015b).

Scratch Scratch é um ambiente de programação visual inicialmente projetado para usuários de idades entre 8 e 16 anos, para o aprendizado de programação através da criação de

histórias animadas e jogos (Lab, 2015). Um dos seus principais objetivos é a introdução da programação para usuários que nunca tiveram contato com a mesma. A programação se dá pelo encaixe de blocos de comando para controlar objetos gráficos 2D que se encontram em um fundo chamado palco (*stage*). Os projetos que são criados no Scratch podem ser salvos localmente ou compartilhados no website do Scratch. Como a manipulação de mídias é uma atividade popular entre crianças e adolescentes, o Scratch se baseou nestas atividades para incentivar os estudantes a aprender através da exploração e compartilhamento entre colegas. Desta maneira, segundo John Maloney et al., (2010), dá-se um foco menor em instruções diretas, o que é comum ocorrer nas demais linguagens de programação.

O projeto Scratch começou em 2003, tendo o seu website publicamente lançado em 2007 e sua distribuição gratuita sendo disponível em mais de 50 línguas, incluindo a versão em Português, o que pode ser um diferencial (Lab, 2015). Além de já ter tido mais de dois milhões de *downloads*, o Scratch é também frequentemente distribuído por escolas e organizações educacionais. Em 2010, a média era de 1500 novos projetos postados no website todos os dias, o que resultava em uma média de mais de um novo projeto por minuto (John Maloney et al., 2010).

De acordo com John Maloney et al., (2010), o projeto se baseou em ideias construcionistas, de maneira a ajudar seus usuários a tornar seus projetos engajadores, motivadores e significativos, Scratch facilita a importação ou criação de diversos tipos de mídias, como imagens, sons, e músicas. Além disso, proporciona um contexto social para todos os usuários, os quais podem receber *feedbacks* dos projetos compartilhados, apoio de outros usuários e aprendizados advindos de projetos alheios.

Em geral a aprendizagem do Scratch se dá conforme os usuários testam o mesmo, ao tentar comandos da palheta ou explorando o código de projetos existentes, e para facilitar a escrita de projetos o ambiente de programação do Scratch foi desenhado para proporcionar feedback imediato na execução de um script, tornando a execução e os dados visíveis ao usuário (John Maloney et al., 2010).

A tela de interação do Scratch (Figura 25) é formada por quatro painéis: o da esquerda mostra os comandos disponíveis, o do meio mostra o script com os comandos selecionados para o objeto em questão, e o da direita mostra o palco onde as ações acontecem e os objetos disponíveis.

No website do Scratch (Lab, 2015) é também possível testar online o ambiente de programação e também aprender com projetos postados pelo próprio Scratch, conforme Figura 26.

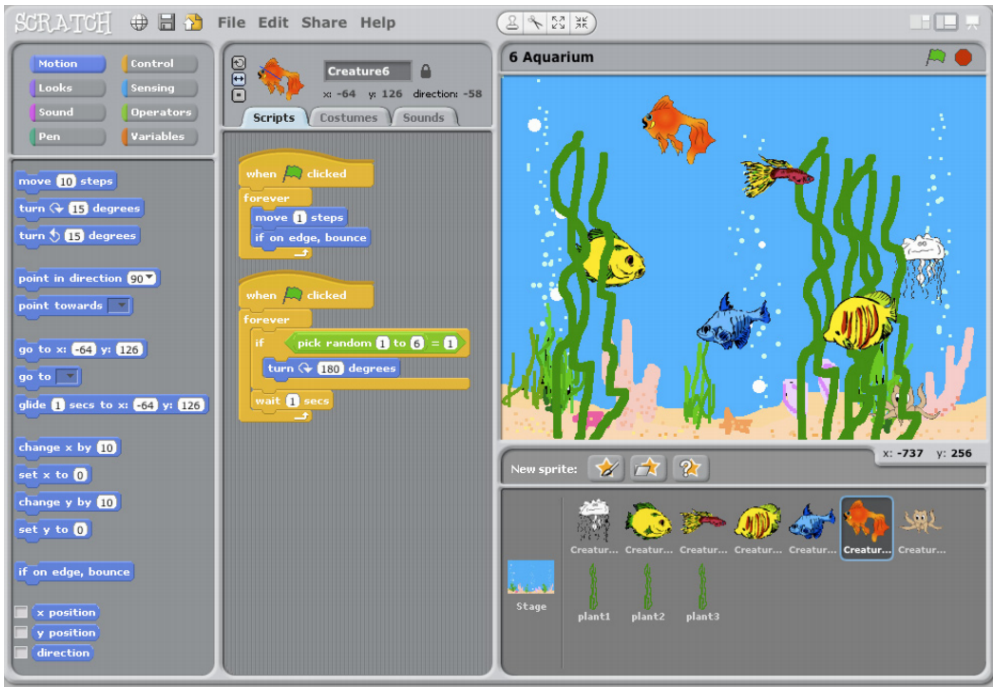


Figura 25: Tela principal do Scratch. Fonte: (John Maloney et al., 2010)

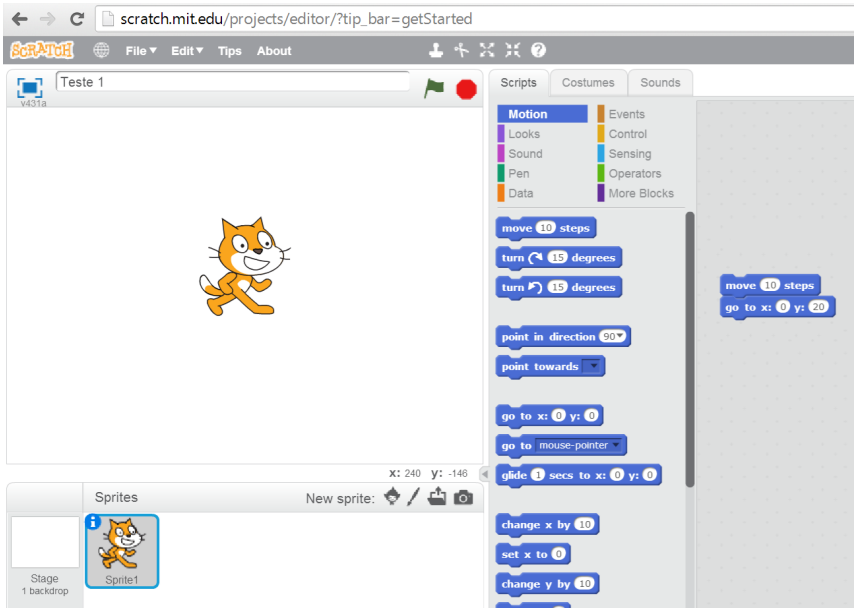


Figura 26: Testando o Scratch online.

Jeroo Jeroo é uma ferramenta de ensino de programação que foi desenvolvida com o objetivo de ajudar os novatos na área a aprender conceitos fundamentais de programação orientada a objetos, focando principalmente em: Instância e utilização de objetos, escrita de métodos para compreensão de comportamento, e seleção e utilização de estruturas de controle fundamentais (Nebraska, 2015). Para isso o objetivo de Jeroo é envolver os estudantes por meio do conto

de histórias e execuções animadas de código.

De acordo com Dorn and Sanders (2003), os estudantes novatos tendem a perder o interesse no assunto devido ao fato de que em geral estes cresceram com videogames e interfaces gráficas de usuário, enquanto as típicas atividades de programação envolvem entradas/saídas baseadas em texto, as quais são ligeiramente disfarçadas com mensagens e caixas de diálogo simples. O aprendizado de programação através das ferramentas tradicionais faz com que os discentes tendam a perder a auto-confiança, pois muitos dos conceitos são abstratos e difíceis de visualizar ou entender inicialmente.

Jeroo é um ambiente de desenvolvimento integrado e micromundo que foi inspirado por Karel the Robot e seus descendentes (Nebraska, 2015). Segundo Dorn and Sanders (2003), os estudantes que trabalharam com Jeroo reportaram um aumento da confiança em sua capacidade para aprender programação, além de que os professores que utilizaram Jeroo em seus cursos apontaram um envolvimento imediato dos estudantes com o tema e que os mesmos pareciam dominar conceitos importantes mais rapidamente do que fizeram quando Jeroo não foi utilizado.

O ambiente Jeroo se baseia em uma metáfora, na qual este é um animal semelhante à um kangaroo que vive em uma ilha isolada e se locomove com saltos. Ele apenas pode se locomover nas 4 direções do compasso, e também encontra na ilha um tipo de flor, que serve como alimento, podendo pegar, carregar e plantar as mesmas. Os Jeroos também devem estar atentos para armadilhas de caçadores que querem capturá-lo para formar um novo zoológico.

A linguagem Jeroo foi projetada para que sua sintaxe se espelhasse na sintaxe comum de Java, C ++ e C . A interface de usuário consiste em uma única janela, de modo que o estudante possa visualizar as consequências de suas ações. De maneira geral, no painel da esquerda pode-se editar o código-fonte, enquanto no da direita é possível visualizar a ilha do kangoroo, como exemplifica a Figura 27.

O programa se baseia nas seguintes condições (Dorn and Sanders, 2003):

- Direções relativas são dadas pelas constantes: *LEFT, RIGHT, AHEAD, HERE*;
- As direções do compasso são especificadas por: *NORTH, EAST, SOUTH, WEST*;
- Cada objeto Jeroo tem três atributos que são sempre visíveis e que podem ser afetados por construtores ou outros métodos pré-definidos: localização, direção, e número de flores;
- Métodos sensores de Jeroo são aqueles que dão respostas a perguntas como "existe uma armadilha (rede) ao lado de Jeroo?"; Eles são: *hasFlower(), isFlower(), isNet(), isWater(), isFacing()*;
- A linguagem de Jeroo suporta três estruturas de controle fundamentais: *while, if* e *if-else*; e

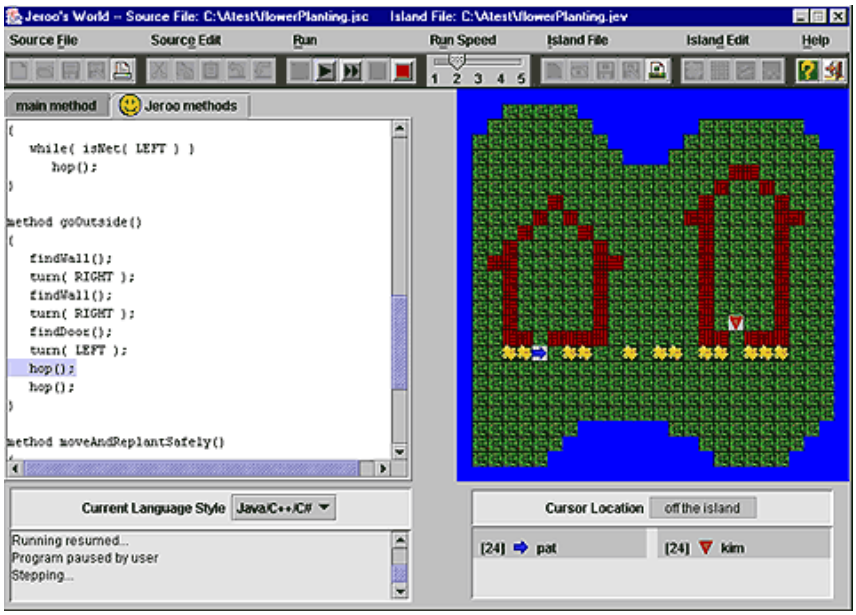


Figura 27: A interface do usuário em Jeroo. Fonte: (Dorn and Sanders, 2003)

- Os métodos de ação permitem Jeroo circular pela ilha e interagir com redes e flores, sendo estes: *hop(), turn(), pick(), plant(), toss()*.

Robocode Robocode é um jogo de programação, onde o objetivo é codificar uma batalha de tanques robô para competir contra outros robôs em uma arena de batalha. O jogador é o programador do robô, que não terá influência direta sobre o jogo, mas sim deverá desenvolver a inteligência do robô para que este saiba se comportar e reagir em campo de batalha, conforme Figura 28 (Nelson, 2015). As batalhas rodam na tela em tempo real.

Além de ser um jogo de programação, Robocode é usado para aprender a programar, principalmente na linguagem Java, mas outras linguagens como C e Scala estão se tornando populares também. O Robocode não tem sido apenas utilizado para o ensino de programação, mas também para o estudo de inteligência artificial (AI) (IBM, 2003).

Quando o Robocode é ativado, duas janelas GUI inter-relacionadas são exibidas, que formam o IDE do Robocode, sendo estas: o campo de batalha e o editor do robô. Um robô em Robocode consiste em uma ou mais classes Java, e cada robô é um veículo que possui uma arma e um radar ao topo, sendo os três alinhados por padrão.

Os comandos do robô são os seguintes:

- *turnRight/Left(double degree)*: vira o robô em um determinado grau;
- *ahead/back (double distance)*: move o robô na distância e direção especificadas;
- *turnGunRight/Left (double degree)*: move a arma, independente da direção do veiculo; e

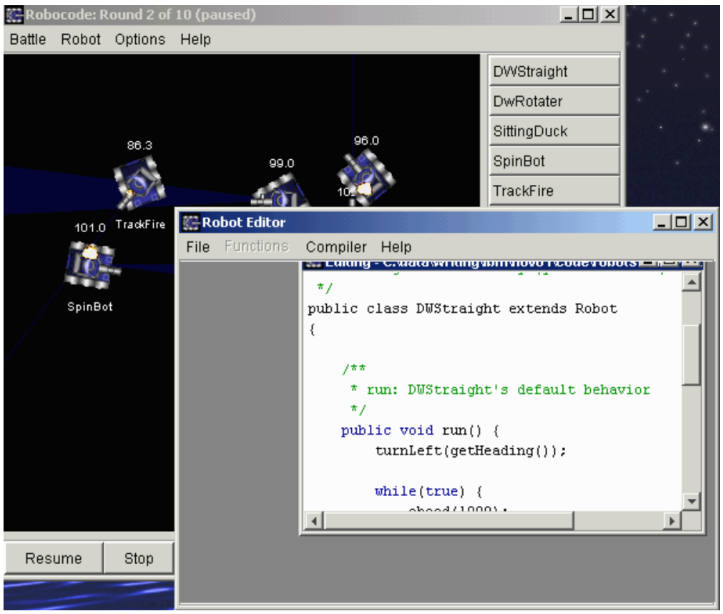


Figura 28: O campo de batalha e o editor do robô em Robocode. Fonte: (IBM, 2003)

- *turnRadarRight/Left(double degree)*: move o radar, independente da direção da arma e do veículo.

Para se obter informações do robô, utilizam-se os seguintes comandos:

- *getX(), getY()*: retorna a coordenada atual do robô;
- *getHeading(), getGunHeading(), getRadarHeading()*: retorna o apontamento em graus do veículo, arma ou radar; e
- *getBattleFieldWidth() e getBattleFieldHeight()*: retorna a dimensão do campo de batalha.

Depois de ter aprendido como mover o robô e seu armamento disponível, os estudantes aprendem as tarefas relacionadas a disparar e a controlar danos. Cada robô inicia com uma energia padrão, e é considerado destruído quando o seu nível de energia cai para zero. Quando em modo de disparo, o robô pode utilizar até três unidades de energia. Quanto mais energia fornecida à bala, mais dano vai provocar ao robô alvo. Apesar dos robôs possuírem armamento para ser utilizado, não necessariamente os jogos precisam envolver o uso destes.

Na criação de um robô, um código é autogerado e dividido em 4 áreas (conforme Figura 29), que devem ser codificadas mais especificamente:

- Área 1: em que são definidos e setados os valores das variáveis;
- Área 2: é o código que irá rodar apenas uma vez por instância de robô; faz com que o robô adentre um estado antes de tomar uma ação repetitiva;

- Área 3: em que é programado uma ação repetitiva do robô através de um *loop while()*; e
- Área 4: em que são adicionados os métodos os quais o robô pode vir a utilizar, ou onde se pode tratar eventos.

```
package dw;
import robocode.*;

/**
 * DWStraight - a robot by (developerWorks)
 */
public class DWStraight extends Robot
{
 ...  // <<Area 1>>
 /**
  * run: DWStraight's default behavior
  */
 public void run() {
  ... // <<Area 2>>
  while(true) {
  ... // <<Area 3>>
  }
 }
      ... // <<Area 4>>
 public void onScannedRobot(ScannedRobotEvent e) {
  fire(1);
 }
```

Figura 29: As áreas de 1 a 4 no editor de robô em Robocode. Fonte: (IBM, 2003)

Pygame Pygame é uma biblioteca multi-plataforma projetada para tornar mais fácil a escrita de softwares multimídia, como jogos, em Python. Pygame requer a linguagem Python e biblioteca multimídia SDL. (Shinners, 2015).[6]

Ele também pode fazer uso de bibliotecas populares, como a SDL (*Simple DirectMedia Layer*). Segundo o seu criador Shinners (2015), o objetivo inicial era de utilizar o Python para tornar fácil a realização de tarefas simples e também descomplicar as tarefas difíceis no âmbito da programação.

Phyton é considerada por alguns autores ideal para introdução da programação (Rebouças et al., 2010)(Marques et al., 2010) por ser fácil para o desenvolvimento rápido de jogos, além de facilitar o primeiro contato com a programação por conta de possuir uma sintaxe simples e flexível, feedback imediato, módulos fáceis de usar e o requerimento de uma indentação apropriada.

No Pygame, a tela de codificação pode istanciar uma nova janela, onde as animações acontecem. O autor dá um exemplo de como faria para codificar a imagem de uma bola quicando, conforme Figura 30.

```
1   import sys, pygame
2   pygame.init()
3
4   size = width, height = 320, 240
5   speed = [2, 2]
6   black = 0, 0, 0
7
8   screen = pygame.display.set_mode(size)
9
10  ball = pygame.image.load("ball.bmp")
11  ballrect = ball.get_rect()
12
13  while 1:
14      for event in pygame.event.get():
15          if event.type == pygame.QUIT: sys.exit()
16
17      ballrect = ballrect.move(speed)
18      if ballrect.left < 0 or ballrect.right > width:
19          speed[0] = -speed[0]
20      if ballrect.top < 0 or ballrect.bottom > height:
21          speed[1] = -speed[1]
22
23      screen.fill(black)
24      screen.blit(ball, ballrect)
25      pygame.display.flip()
```

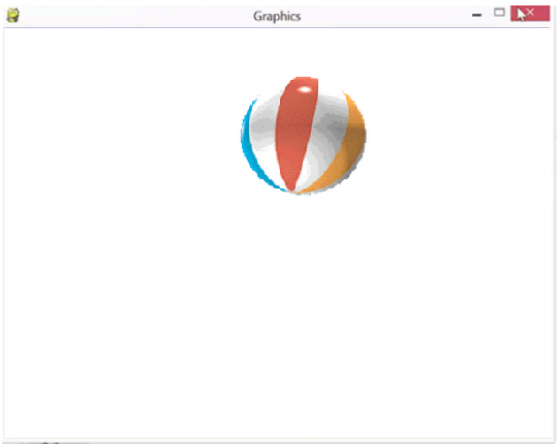


Figura 30: Código para criação de uma bola quicante no Pygame. Fonte: (Shinners, 2015)

[6]Python é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Em outras palavras, Python suporta múltiplos paradigmas de programação, é uma linguagem que foi criada para produzir código limpo e fácil de manter de maneira rápida. (PythonBrasil, 2015)

Na linha 10, a imagem da bola é carregada, e na linha seguinte uma área retangular para a bola é criada. A partir da linha 13, dentro de um loop infinito, verifica-se se o usuário desejou sair, se não, move a bola e a redesenha novamente. As linhas 17 a 21 são para corrigir caso a bola tenha saído dos limites da tela. Na linha 23 limpa-se a tela, na 24 redesenha-se a bola e na 25 os elementos da tela se tornam visíveis.

O Pygame ainda pode possibilitar o desenvolvimento de uma grande diversidade de formas não gráficas e gráficas, podendo estas serem animadas ou não, como calculadoras, formas geométricas (círculos, quadrados, etc) e até jogos completos, conforme os tutorias aprensentados em (Craven, 2015).

3.3 Bases Educacionais do Curso de Sistemas de Informação

No processo de escolha da metodologia de ensino empregada, a qual está aliada aos softwares educacionais elencados, é necessário considerar não somente os aspectos relativos ao processo de ensino-aprendizagem do corpo discente que participará das dinâmicas propostas via softwares educacionais, mas também, há que se considerar os aspectos relativos à formação discente em Bacharelado em Sistemas de Informação (BSI), já que esse Trabalho de Conclusão de Curso tem aderência a esse curso e área.

Por esse motivo, decidiu-se utilizar o projeto abertura do curso de bacharelado em Sistemas de Informação, apresentado ao Conselho de Ensino pela Comissão designada pela Portaria n 285 de 11 de dezembro de 2007 da Diretoria do Campus Curitiba da UTFPR, como elemento de análise para a definição da metodologia e dos softwares voltados ao ensino de programação selecionados.

Esse projeto de abertura do curso discorre sobre a organização didático-pedagógica do mesmo e os objetivos que pretendem ser alcançados durante a formação do estudante. Esse projeto possui as seguintes bases: Colegiado, Integração, Multidisciplinaridade/ Interdisciplinariedade, Flexibilidade e Visão Humanista, das quais as últimas quatro servirão de orientação para a escolha da metodologia de ensino empregada.

A Integração Interdisciplinar é apontada como uma das prioridades do curso de BSI, com pretensão de ser desenvolvida tanto em períodos específicos quanto ao longo de todo o curso. Graeml et al., (2008) também afirma que: “este modelo preconiza a substituição de disciplinas isoladas, por disciplinas integradas, nas quais os conteúdos comuns serão investigados/descobertos pelos estudantes e evidenciados/valorizados pelos professores”.

Apesar da não existência de uma definição precisa para a Interdisciplinaridade, Fortes (2012) sugere um sentido comum, o qual se caracteriza por muito mais do que uma simples integração dos conteúdos. A Interdisciplinaridade é, para Fortes (2012) tornar as disciplinas comunicáveis entre si e compreender as ligações entre diferentes áreas de conhecimento, visando expandir sabedorias, mostrar possibilidades e “ultrapassar o pensar fragmentado”,

por meio de investigação constante na tentativa de superação do saber.

É importante enfatizar que a interdisciplinaridade compõe um eixo integrador com as disciplinas de um currículo, para que os estudantes aprendam a olhar o mesmo objeto sob perspectivas diferentes. Oliveira (2010) informa que a Interdisciplinaridade começou a ser abordada no Brasil a partir da Lei de Diretrizes e Bases Nº 5.692/71[4]. Além da sua grande influência na legislação e nas propostas curriculares, a interdisciplinaridade tornou-se cada vez mais presente no discurso e na prática de professores.

Portanto, relaciona-se nesse trabalho as bases educacionais do curso de Bacharelado em Sistemas de Informação: Integração, Interdisciplinaride, Flexibilidade, e Visão Humanista com autores e visão metodológicas de ensino correspondente.

3.3.1 Integração

A integração no curso de Bacharelado em Sistemas de Informação é descrita por Graeml et al., (2008) como sendo uma das prioridades do curso. Fazenda (2011) define que integração é uma condição para o cumprimento da interdisciplinaridade, pois estimula novas buscas e a transformação da realidade. Fazenda (2011) ainda considera que a integração é como um "momento de organização e estudo dos conteúdos das disciplinas " e isso faz com que a integração seja como uma etapa necessária à interdisciplinaridade, o autor argumenta ainda que a integração é um passo necessário para a interdisciplinaridade pois essa é o aspecto prático da integração. A integração, de acordo com Fazenda (2011), gera preocupações que se desenvolvem no sentido de indagar a realidade e visualizar as possibilidades de transformação, isso classifica a integração como uma fase anterior à interdisciplinaridade.

Graeml et al., (2008) alegam que o modelo integrado indica a substituição de disciplinas isoladas por disciplinas integradas e nessas os conteúdos comuns podem ser descobertos pelos estudantes e incrementados pelos educadores. A integração de duas ou mais disciplinas resultam na interdisciplinaridade conforme afirma (Fazenda, 2011), pois para isso "compõe-se de pessoas que receberam sua formação em diferentes domínios do conhecimento (disciplinas) e com seus métodos, conceitos, dados e termos próprios". Logo, necessidade de uma integração aponta uma interdisciplinaridade com novas indagações para alcançar uma mudança na maneira de compreender e aprender.

De acordo com Abramof and Mota (2007) há diferentes estratégias de integração de disciplinas, no curso de Bacharelado em Sistemas de Informação a estratégia de integração utilizada é a interdisciplinaridade. A integração interdisciplinar de acordo com Abramof and Mota (2007) busca um nível de contribuição entre as disciplinas organizadas de maneira que alcancem um nível superior de entendimento de conceitos, métodos, leis, e fenômenos envolvidos.

[4] Legislação que regulamenta o sistema educacional (público ou privado) do Brasil (da educação básica ao ensino superior). E recentemente mudada para a nova LDB n 9.394/96.

3.3.2 Interdisciplinaridade

De acordo com Krausz (2011), o primeiro pensador a definir o ensino transdisciplinar como um grau mais elevado de interdisciplinaridade foi Jean Piaget. Krausz (2011), cita que Piaget afirmava que haveria um momento na história do pensamento humano em que a interdisciplinaridade alcançaria um grau de conexão tão intenso que as disciplinas, para além do diálogo, chegariam a um nível mais elevado de interação.

Outro autor que possui uma relação com a interdisciplinaridade é Paulo Freire, essa relação é descrita no estudo de Rocha and Rocha (2013) em que o autor afirma que a interdisciplinaridade para Paulo Freire está fundamentada em dois pressupostos:

1. Afirma que a interdisciplinaridade ainda não foi compreendida, e isso faz com que o tratamento prático para interdisciplinaridade seja equivocado, onde na prática tem-se assumido uma reunião de disciplinas, a negação do conhecimento disciplinar ou tomado fim em si mesma; e

2. Há um rompimento da visão fragmentada de mundo e de educação, empenha-se em abordar uma visão integradora, totalizando a prática pedagógica à edificação do conhecimento, demarcadas pelas visões pedagógicas de Paulo Freire em que este define que "novos conhecimentos, valores e atitudes vão sendo construídos em práticas sociais diferenciadas".

A proposta de abertura de curso de BSI da UTFPR indica a importância da interdisciplinaridade justificando-a pela necessidade de atualização constante da formação em computação e a revisão continuada de conteúdos resultante da atuação de docentes de múltiplos departamentos acadêmicos.

O ensino de programação também pressupõe essa constante formação e revisão continuada de conteúdos e conceitos.

3.3.3 Flexibilidade

Conforme apresentado em Graeml et al., (2008), a flexibilidade é vista como uma das bases do projeto do curso de BSI, em que esta é definida como a compatibilidade com outros campos do saber, tanto com relação ao que é oferecido nos cursos da UTFPR (em especial com o curso de Engenharia de Computação), quanto também com cursos oferecidos em outras instituições nacionais e estrangeiras. Esta compatibilidade se dá por aquilo que o curso oferece em termos de conteúdo básico e também em termos de disciplinas optativas profissionalizantes, garantidos pelo embasamento em documentos de relevância nacional e internacional.

Desta forma, a flexibilidade apresentada pelo projeto do curso se pauta no entendimento de que o estudante formado (ou que migrem para o mesmo) poderão estar assegurados quanto ao seu conhecimento e processo de aprendizagem, fornecendo desta forma autonomia e liberdade para se mover para outros espaços que se encontrem fora do curso de BSI.

Portanto, a flexibilidade compreendida pelo curso pode ser traduzida em flexibilidade curricular, em que um currículo mais flexível possibilita a articulação entre diversos campos do saber, e consequentemente proporciona maior autonomia e liberdade aos seus estudantes. Segundo projeto do curso, esta flexibilização é facilitada também pela expressiva porcentagem de disciplinas optativas, o qual visa abrir um arco de formação no qual, de acordo com Graeml et al., (2008), "facilita a recepção de estudantes oriundos de outras instituições vistas as condições estruturais propostas pelo REUNI".

Conforme (Fior and Mercuri, 2009), o termo flexibilidade sofre variações quanto à sua compreensão de acordo com o contexto no qual é utilizado. Portanto, enfatiza-se que a flexibilidade educacional se distancia bastante da flexibilidade almejada no contexto empresarial. Desta forma, (Fior and Mercuri, 2009) chama atenção para o perigo da submissão de uma à outra, pois a flexibilidade curricular não é a sobreposição das questões empresariais às sociais, não sendo uma adaptação dos currículos às necessidades profissionais, ou seja, às demandas das empresas. Essa pesquisa trata sobre flexibilidade no contexto educacional.

Flexibilidade curricular se caracteriza então pela possibilidade de experiências que ampliem a graduação do estudante, acrescentando o contato com diversas áreas do conhecimento, pressupondo assim uma "opção filosófica que valoriza os atores educativos, o desenvolvimento contextualizado de práticas educativas, a autonomia da instituição, do professor e do aluno" (Fior and Mercuri, 2009). De acordo com a Orientação para as Diretrizes Curriculares dos cursos de graduação, como parte da flexibilização sugere-se a realização de atividades complementares, com o intuito de ir além dos limites das grades de disciplinas dos cursos, assegurando desta forma "a prática de estudos e atividades independentes com características interdisciplinaridades e opcionais, a fim de enriquecer e implementar o perfil profissional do formando".

Portanto, de acordo com (Neto et al., 2004), com da flexibilização curricular e da consequente expansão dos horizontes do seu conhecimento, o corpo discente pode obter uma visão crítica que o permite ir além da aptidão específica do seu campo de atuação profissional, contribuindo em grande parte para sua formação enquanto cidadão.

Quando correlacionando o conceito às visões metodológicas de ensino, o cognetivista Jean Piaget já defendia um ambiente de aprendizagem que estimulasse o estudante a passar pelo máximo de experimentação em programas mistos, para desta forma se dar a construção da educação e não do adestramento, viabilizando assim a reconstrução da verdade pelo próprio estudante (Treviso and de Almeida, 2014).

Em termos de oposição a um currículo imposto de forma rígida e unilateral, a linha Construtivista prioriza a flexibilização do mesmo através de conjuntos de problemas que envolvam diversos conceitos de maneira simultânea. Nesse contexto surge a Teoria da Flexibilidade Cognitiva (TFC) (Pessoa and Nogueira, 2009), desenvolvida por Rand Spiro e colaboradores

na década de 80, sendo baseada em pressupostos construtivistas. A principal preocupação da mesma é a busca de respostas aos desafios presentes na construção de conhecimentos em domínios complexos e pouco estruturados, os quais são comumente abordados no ensino superior. Conforme discorre Pessoa and Nogueira (2009), em vez da priorização da familiaridade de conceitos e da imitação de regras ou reprodução de conhecimento, os objetivos da aprendizagem mais avançada em TFC passam a ser o domínio da complexidade de forma que os conhecimentos aprendidos possam ser utilizados de várias maneiras, em novos contextos, e diferentes situações.

Portanto, a TFC se relaciona com a flexibilização curricular na medida em que se faz necessário o revisitar do mesmo conteúdo em uma variedade de contextos diferentes, trazendo desta forma aspectos adicionais à complexidade do conteúdo, assim o aprendizado se dá por explorações conduzidas de forma flexível e multidimensional (Pessoa and Nogueira, 2009). O aprendizado através de múltiplas representações em TFC não é compatível portanto com o saber linear, hierárquico, compartimentado, e dependente de uma única perspectiva.

Numa perspectiva Humanista, conforme aponta Aguiar (2011), Paulo Freire visa o curriculo m ciclos, o qual considera os estágios de desenvolvimento ritimos de aprendizagem dos estudantes. Portanto, critica a lógica da escola seriada e sua concepção curricular técnico-linear, o que ele chama de "educação bancária", ou seja, que se dá em uma relação verticalizada de dominação do educador-educando, e onde o educando se torna um "mero espectador da realidade".

Desta forma, Freire se mostra contrário ao ensino descontextualizado, em que os conteúdos são apresentados sem significado e alheios à experiência existencial dos educandos, ou seja, num contexto fora da comunidade onde se está inserido. Portanto, seus pensamentos se relacionam com o conceito de flexibilização curricular uma vez que "se valoriza o conjunto de práticas culturais e sociais com intencionalidades éticas e políticas, que se inter-relacionam com o tempo histórico e o espaço escolar" (Aguiar, 2011).

À vista disso, a flexibilização curricular ideal prevista por Freire faz parte, juntamente com uma educação dialógica e crítica numa práxis libertadora, das práticas educacionais necessárias para a superação da educação bancária. É possível perceber, apesar de não exatamente da maneira idealizada por Freire, semelhanças entre estes ideais humanistas e o conceito de flexibilidade apontado pelo projeto do curso de BSI. Isto se dá principalmente pela valorização da autonomia do estudantes fora do próprio curso, além do incentivo quanto às disciplinas optativas e a troca com a comunidade por meio da já existente prática de atividades complementares.

3.3.4 Visão Humanista

Em Graeml et al., (2008), a fundamentação do projeto do curso é baseada no princípio da Visão Humanista, em que este é definido pela concepção de uma graduação que vai além da formação de bons profissionais na área, sendo também necessário que estes discentes se formem cidadãos críticos, reflexivos e cientes de suas obrigações. Para o alcance disso, o projeto do curso propõe que sejam consideradas fundamentais as disciplinas das áreas de Ciências Humanas, Sociais Aplicadas, e Ciências Ambientais (além das atividades complementares), devendo ser oferecidas de maneira transversal do início ao fim do curso.

Dada a importância da construção da cidadania de modo a formar indivíduos mais conscientes pelo projeto de curso de BSI, alguns autores chamam a atenção para o uso deste conceito. Ferreira (2008) se volta para a necessidade da reflexão do mesmo quanto à definição de "cidadão crítico, criativo e participativo" no âmbito educacional, uma vez que se percebe o fortalecimento deste conceito nos textos oficiais, juntamente com o aumento do risco do mesmo se tornar "cristalizado" e meramente "um lugar comum". Isso se dá ao ter seu emprego generalizado ou banalizado sem a devida reflexão do seu verdadeiro significado. A autora Ferreira (2008) parafraseia Paulo Freire ao afirmar que "é ingenuidade imaginar que se constrói a cidadania, a criticidade, a criatividade apenas copiando e repetindo os discursos oficiais onde tais jargões se fazem presentes".

Desta forma, os documentos oficiais que orientam as práticas pedagógicas, sendo estes a Lei de Diretrizes e Bases da Educação Nacional (LDB), os Parâmetros Curriculares Nacionais (PCN) e o Projeto Pedagógico Institucional (PPI), preveem a necessidade da educação em formar cidadãos, de forma a obter uma "formação integral do sujeito que seja capaz de lidar com a diversidade de maneira crítica, com autonomia e iniciativa para propor soluções para os problemas que se apresentem" (Ferreira, 2008). Portanto, de maneira geral, esta cidadania abre portas para um profissional que tem a capacidade de agir com criatividade, criticidade e autonomia no mundo.

Em termos de políticas educacionais, o conceito de cidadania é ampliado, englobando os significados éticos necessários para a construção das ações que a concretizem neste contexto, sendo estas definidas pela PCN como: i) aprender a conhecer, ii) aprender a fazer, iii) aprender a viver com os outros, e iv) aprender a ser. De acordo com Ferreira (2008), esta construção da cidadania a coloca como requisito básico para a existência dos indivíduos no mundo, sem a qual os mesmos seriam "relegados à condição de coisa, objeto manipulável", impedindo a construção de "sujeitos". A escola é portanto, o espaço eleito para a legitimação desta transformação, o que se constitui em um direito de todos previsto em lei.

Um dos autores que reconhece a importância de uma educação para a formação crítica é Paulo Freire. Conforme apontado em Cortez and Moraes (1979), a educação é descrita por Freire como "prática da liberdade, é um ato de conhecimento, uma aproximação crítica

da realidade". Portanto, Freire considera que o primeiro objetivo de toda educação é a provocação de uma atitude crítica, de reflexão, que comprometa a ação. Desta forma, suas ideias estão intimamente ligadas ao conceito de cidadania, uma vez que, de acordo com o autor, o se fazer "sujeito" se dá por esta capacidade de auto-reflexão sobre sua situação: "quanto mais refletir sobre a realidade, sobre sua situação concreta, mais emerge, plenamente consciente, comprometido, pronto a intervir na realidade para mudá-la" (Ferreira, 2008).

Este estímulo da consciência e de uma atitude crítica proporciona ao indivíduo a libertação através da autenticidade de suas escolhas e decisões próprias, o que o torna "ser humano" e o livra da condição de submissão e possível "domesticação". Portanto, Freire foca na educação social e política para o desenvolvimento da cidadania, e, de acordo com (Bonin, 2008), esta pedagogia da libertação supõe o surgimento do "sujeito epistêmico, conhecedor consciente dos processos sociais de sua cultura a fim de superar uma consciência ingênua".

Ainda correlacionando o conceito às visões metodológicas de ensino, em uma perspectiva diferente da apresentada por Freire, que se interessa em pesquisar a ação para a formação do cidadão, Vygotsky, inspirado nos trabalhos iniciais de Piaget, tem como foco o desenvolvimento da mente como formação social. Conforme descrito em (Bonin, 2008), para ambos os autores a consciência não é um mecanismo desconectado do contexto de vida e da ação transformadora do homem. Portanto, a construção social do sujeito, para Vygotsky, está atrelada ao contexto histórico-cultural do mesmo, se contrapondo, conforme aponta (Rossetto and Brabo, 2007), ao conceito de "subjetividade privatizada" de sua época. Este conceito se baseava em ideais liberais e românticos e se fundamentava na ideia de um "sujeito abstrato, à parte de sua cultura, desconectado de sua história e relações sociais nas quais estava inserido", considerando sua autonomia, decisões e emoções um fenômeno de ordem privada e não coletiva (Rossetto and Brabo, 2007).

À vista disso, é possível afirmar que as ideias de Vygotsky se correlacionam com o desenvolvimento da cidadania nos indivíduos na medida em que estas apontam para a construção de um sujeito social. Com base de sua pesquisa acerca dos sistemas psicológicos que ocorrem no processo de individuação do homem, Vygotsky explora as origens do pensamento, da consciência, da comunicação em geral e, essencialmente, da linguagem (Bonin, 2008), apontando um processo de construção do sujeito que é intrinsecamente atrelado à inserção histórica e social do mesmo em uma cultura.

3.4 Análise de Softwares Educacionais

A análise de softwares, segundo Campos and Campos (2011) deve primar por normas de qualidade de software, afirma ainda que um software deve seguir a norma ISO (ISO/CD8402, 1990), na qual é definido que "qualidade é a totalidade das características de um produto ou serviço que lhe confere a capacidade de satisfazer as necessidades implícitas de seus usuários",

isso deixa evidente que a qualidade está ligada à satisfação do cliente.(Campos and Campos, 2001) argumenta que além dessa norma citada, o processo de análise de um software deve passar por outras normas, exemplifica a norma ISO/IEC 9126:1991 que define qualidade de software relacionando em externa, que é a visível aos usuários do sistema, e interna, que é relacionado a desenvolvedores, (Campos and Campos, 2001)relata a importância da qualidade do produto, no caso o software, citando a série de Normas ISO/IEC 9126, na série ISO/IEC 14598 e na Norma ISO/IEC 12119, esta última focalizando os requisitos de qualidade de pacotes de software.

Porém, (Campos and Campos, 2001) afirma que para análise de um software educacional, além dessas normas deve-se levar em considerações a tecnologia específica. Para (Campos and Campos, 2001) as teorias de aprendizagem espelham visões diferentes de como ocorre a aprendizagem e estas visões têm desdobramentos nos software educacionais. O autor (Campos and Campos, 2001) defende ainda que a avaliação de um software educacional deve se iniciar pela identificação do ambiente educacional ou do potencial de uso do software para um determinado ambiente educacional. (Campos and Campos, 2001) cita como características importantes de um software educacional as características pedagógicas, facilidade de uso, características da interface, adaptabilidade, documentação, portabilidade, e retorno do investimento.

(Vieira, 2010) interpreta que antes de pensar a respeito dos softwares educativos, deve-se refletir sobre o papel do computador na educação, o autor (Vieira, 2010) defende que o computador não tem o papel somente de facilitar o processo de aprendizagem, tem como objetivo estimular o desenvolvimento de habilidades auxiliar a aprendizagem dos estudantes para que se tornem sujeitos participativos na sociedade.(Vieira, 2010) explana ainda que avaliar um software educativo significa analisar como um software pode ter uso educacional, como um software pode auxiliar o estudante a participar da realidade em que está vivendo a partir de uma mudança de compreensão de mundo.

A avaliação do software para Vieira (2010) colabora na identificação de qual proposta pedagógica o software deve ser melhor adaptado. Para (de Souza Rezende, 2013) a evolução das Tecnologias de Informação e Comunicação (TIC) dão um propósito real para avaliação de software, pois justificam a verificação dos produtos de software, para que haja um uso correto desses recursos. (de Souza Rezende, 2013) afirma que para compreender métodos de avaliação de software têm que utilizar análise de modelos, produtos e ferramentas de uma maneira que ocasionem em contribuições para a qualidade de software.

(Rocha and de Campos, 1993) categorizam os usuários de softwares educacionais em professores, estudantes, mantenedores, e desenvolvedores, para cada uma dessas categorias (Rocha and de Campos, 1993) afirma que há um significado diferente no que esses esperam da qualidade de software, pois a qualidade irá refletir o ponto de vista desses usuários. O autor

(Rocha and de Campos, 1993) argumenta então que no desenvolvimento do software educacional tem que estar inserido o modelo ensino-aprendizagem proposto, ou seja, a metodologia de ensino e a filosofia de aprendizagem devem ser subjacente ao software e os atributos que integram a qualidade devem ser definidos na fase de análise de requisitos.

Para a seleção dos softwares educacionais presentes nessa pesquisa, serão considerados softwares que contribuam para o aprendizado do estudante, dadas as definições de (Giraffa, 2009), que define que Software Educacional (SE) pode ser qualquer software, desde que se adeque aos padrões de ensino e aprendizagem conforme a metodologia de ensino utilizada pelo educador. O uso de softwares educativos, de acordo com (Giraffa, 2009), possibilita para o estudante formas de conectar os estudos de sala de aula a aplicações do seu cotidiano, assim permite que os educadores trabalhem aspectos lúdicos que estimulam o interesse dos estudantes.

A análise dos softwares educacionais depende de alguns critérios (Giraffa, 2009) afirma que para isso, tradicionalmente são utilizadas classificações e que nem sempre estas acompanham a evolução rápida da tecnologia e dos projetos de sistemas.

(Giraffa, 2009) ainda explica que na taxonomia de Taylor, softwares educacionais são classificados de acordo com três grupos: tutor, tutelado, e ferramentas.(Giraffa, 2009) explica que essa taxonomia classifica o software pela visão do computador e não pela visão do usuário. A divergência entre esses tipos de softwares está na especificação deles, programas classificados como tutelados permitem ao estudante "programar" o computador para atuar de acordo com as perspectivas e necessidades desse. Como ferramenta na taxonomia de Taylor[7] enquadram-se softwares de manipulação de informações, específicos como por exemplo os editores de textos, gerenciadores de banco de dados, planilhas e pacotes gráficos. A classificação de softwares tutores são para aqueles que tutelam o aprendizado do discente, podendo citar o micromundo LOGO.

(Campos and Campos, 2001) defende que os softwares educacionais devem ser delimitados por teorias de aprendizagem que distinguem ambientes educacionais pelo grau de interatividade, participação e controle os estudantes na concepção do conhecimento. (Campos and Campos, 2001) argumenta que o desenvolvimento do software educacional procura compreender características da educação de uma formação global necessária ao estudante, trabalha em funções da cognição influenciando o estudante a aprender a aprender, a pensar, a inovar e questionar.

[7]Classificação de Robert P. Taylor, é atribuido três papeis para o computador na educação, no primeiro, o computador funciona como um guardião,no segundo, o computador funciona como uma ferramenta. No terceiro, o computador funciona como um tutor para o estudante.Para funcionar como uma ferramenta na sala de aula,o computador só precisa ter alguma capacidade útil programada nele, como análise estatística,super cálculo, ou processamento de texto. Os estudantes podem, em seguida, usá-lo para ajudá-los em uma variedade de assuntos. Por exemplo, eles podem usá-lo como uma calculadora em matemática e várias tarefas de ciência, como uma ferramenta de interação em mapas na geografia,ver instrumentos inacessíveis da música, ou como um editor de texto.((Taylor, R. P. 1980)The computer in school: Tutor, tool, tutee.New York:Teachers College Press)

O autor (Campos and Campos, 2001) ainda afirma que apesar de um avanço nos estudos teóricos a cerca de modelos de aprendizagem, há uma grade tendência ao desenvolvimento de programas educativos voltados para o sistema educacional tradicional, e esses se baseiam em exercício e prática, jogos e tutoriais. (Campos and Campos, 2001) por fim descreve as características de softwares que interpretam os modelos tradicionais como sendo programas que definem estratégias de ensino, oferecem reforço para as respostas corretas informam aos estudantes os escores alcançados, determinam objetivos educacionais mensuráveis, e promovem uma avaliação objetiva.

(Alves, 2006) define como critérios para análise de um software ser educacional três aspectos: A documentação, currículo, e aspectos didático. O primeiro aspecto se refere ao material que acompanha o software, com informações sobre faixa etária, manuais de uso, e conteúdos. O currículo é referente à programação prevista para o desenvolvimento de atividades do educador com o software no ambiente escolar. O terceiro aspecto envolvem aspectos didáticos e pode ser dividido em clareza de conteúdos, assimilação e acomodação, recursos motivacionais, avaliação do aprendizado, carga educacional e no tratamento das dificuldades, tratamento de um erro, ou seja, a possibilidade de refazer um exercício e registro de dificuldades encontradas.

3.4.1 Diretrizes para Informática na Educação

O ProInfo, inicialmente denominado Programa Nacional da Informática na Educação, foi criado pelo Ministério da Educação (MEC) com a finalidade de promoção do uso da tecnologia como ferramenta de enriquecimento pedagógico no ensino (FNDE, 2012). A partir de 2007, o ProInfo passou a ser o Programa Nacional de Tecnologia Educacional, tendo como objetivo promover o uso pedagógico das tecnologias de informação e comunicação nas redes de educação brasileiras. O seu funcionamento se dá de forma descentralizada em cada unidade da Federação com uma coordenação Estadual, as quais possuem, de acordo com (FNDE, 2012), infraestrutura de informática e comunicação que reúnem educadores e especialistas em tecnologia de hardware e software.

Desta forma, anteriormente à escolha dos modelos de avaliação de softwares educacionais, é fundamental a compreensão das orientações que regem o uso da informática na educação no Brasil. Conforme apresentado pelas Diretrizes para o Uso de Tecnologias Educacionais do Governo do Estado do Paraná (Melo et al., 2010), para esta compreensão se faz relevante a discussão de alguns conceitos no contexto educacional, a começar pelo de "mediação", uma vez que este subsidiará o trabalho do professor quanto ao uso de Tecnologias da Informação e Comunicação (TIC).

(Melo et al., 2010), através da orientação do MEC de que "o uso das tecnologias na educação deve estar apoiado numa filosofia de aprendizagem que proporcione aos estudantes

oportunidades de interação e, principalmente, a construção do conhecimento", reitera a necessidade de clareza quanto ao trabalho que o professor desempenha enquanto mediador deste conhecimento. Portanto, (Melo et al., 2010), através da fundamentação em um conjunto de autores incluindo Vygotsky, compreende que a aprendizagem é um produto das relações estabelecidas entre os sujeitos, ou seja, um produto da sua interação com o meio social, em que o professor tem a função de mediador didático-pedagógico. O processo de mediação é deste modo tido como:

> "A mediação do professor consiste em problematizar, perguntar, dialogar, ouvir os estudantes, ensiná-los a argumentar, abrir-lhes espaço para expressar seus pensamentos, sentimentos, desejos, de modo que tragam para a aula sua realidade vivida." (Melo et al., 2010, p. 12)

À vista disso, entende-se que é de responsabilidade do professor prover aos estudantes os conhecimentos histórica e culturalmente construídos, além de mediar o processo de aprendizagem com vistas à reconstrução do conhecimento, através de uma "metodologia específica, estratégias de ensino, e os mais diversos recursos didáticos disponíveis, dentre os quais as tecnologias educacionais, pois nisso consiste o processo de ensino" (Melo et al., 2010).

De acordo com estas diretrizes, o papel de mediação do professor envolve então a determinação de novas formas de processar e utilizar a informação com envolvimento ativo e emocional dos educandos no desenvolvimento de ações.

Neste processo de mediação, o uso das tecnologias educacionais proporciona oportunidades de "desafios, de criação e reconstrução de novos conhecimentos". Ou seja, conforme (Melo et al., 2010), as tecnologias educacionais tem o objetivo de não apenas o repasse de conceitos e conhecimentos científicos, mas sim a expansão dos espaços de aprendizagem, o que vem a ampliar as possibilidades de leitura e expressão da realidade dos educandos, potencializando a aprendizagem destes ao mesmo tempo em que enriquece a prática do professor.

Para a transformação da tecnologia em recursos educativos efetivos, (Melo et al., 2010) enfatiza que se faz necessário o planejamento das situações de ensino-aprendizagem, o que requer "intenção, sistematização e objetivos definidos do que se quer ensinar, para quem, com que recursos e como ensinar". A compreensão do contexto educacional se faz essencial, uma vez que ultrapassa o pensamento linear no qual se tenta fazer com que todos aprendam necessariamente "nos mesmos tempos e espaços" (Melo et al., 2010). A contextualização também quanto ao uso das tecnologias auxiliaria, desta forma, a produção de conhecimentos que levem à transformação, promovendo uma sociedade mais "participativa, crítica e igualitária", ou seja, a tecnologia a serviço da cidadania.

No que tange a cidadania, (Melo et al., 2010) determina que as TIC, pelo fato de poder potencializar a articulação do conhecimento das diversas áreas, podem contribuir para a

integração das disciplinas e também permitir que discentes e professores se envolvam em atividades socialmente relavantes e significativas. Ao mesmo tempo, (Melo et al., 2010) reforça a importância de estas tecnologias educacionais serem acompanhadas da devida consciência não apenas social, mas também política e pedagógica das escolas, o que evita a preponderação da racionalidade técnica e portanto, a desumanização da mesma. Esta desumanização aconteceria na medida em que a escola, sem as devidas precauções inclusive quanto à utilização da tecnologia, correria o risco de se transformar num espaço de meras "decisões tecnicistas". Assim, o papel desempenhado pela mesma é definido por:

> "A escola, concebida como espaço de síntese, estaria contribuindo efetivamente para uma educação básica de qualidade: formação geral e preparação para o uso da tecnologia, desenvolvimento de capacidades cognitivas e operativas, formação para o exercício da cidadania crítica, formação ética." (Melo et al., 2010, p. 13)

Uma vez que são feitas estas considerações acerca da aprendizagem e da utilização da tecnologia no contexto educacional, a recomendação geral prevista por (Melo et al., 2010) é a de que, apesar da existência de diferentes modelos pedagógicos que se apoiam na utilização das TIC para o processo de aprendizagem, caberá aos professores, coerentemente com as finalidades expressas nos planos de ensino e no Projeto Público Pedagógico das instituições, a compreensão desses modelos como "recursos configurados histórica e culturalmente como parte do processo educativo, incorporando-os à sua realidade e contexto de forma crítica e criativa", para que essa apropriação tecnológica possa de fato contribuir para o desenvolvimento efetivo/significativo da aprendizagem.

3.4.2 Modelos de Avaliação de Softwares Educacionais

Os métodos ou modelos de avaliação de softwares surgem da necessidade de determinação da qualidade dos mesmos, e a sua consolidação se dá através do estudo, o qual envolve um "modo sistemático de análise de métodos, modelos, produtos e ferramentas" (de Souza Rezende, 2013), tendo como objetivo atingir resultados efetivos quando da avaliação destes softwares.

De acordo com (de Souza Rezende, 2013), através desta análise e entendimento quantos aos requisitos e atributos envolvidos, o uso de modelos para avaliação de software se faz fundamental uma vez que possibilita uma "mensuração qualitativa ou quantitativa do real valor do software para atender a determinado objetivo previamente estabelecido".

Estes modelos se dividem quanto à orientação, podendo ser voltados para a avaliação da usabilidade de software, no que diz respeito à interação humano-computador, ou para uma avaliação de cunho pedagógico, que, de acordo com (de Souza Rezende, 2013), além de levar em conta os requisitos exigidos por softwares em geral, deve também atender a exigências

específicas do ensino e aprendizagem.

Esta seção irá abordar os modelos e ferramentas de avaliação de software voltados para a área pedagógica e que servirão de base para a avaliação dos softwares educacionais: Modelo TUP, Modelo PECTUS e o Modelo de Avaliação de Software Educacional (baseado na proposta do ProInfo) (Ribeiro, 2013a).

Modelo TUP O modelo TUP (*Technology, Usability and Pedagogy*) tem como objetivo prestar a avaliação, para educadores, criadores, e usuários de ambientes educacionais, no que se refere a aspectos tecnológicos, de usabilidade e pedagógicos destas ferramentas (Bednarik, 2013). De acordo com (de Souza Rezende, 2013), o modelo surgiu da necessidade de criação de um método de avaliação de software educacional que permitisse aos educadores a seleção de softwares de acordo com um conjunto estruturado de critérios definidos. A Figura 31 apresenta os atributos, os quais pertencem ao conjunto dos três requisitos previstos no modelo (tecnologia, usabilidade e pedagogia).

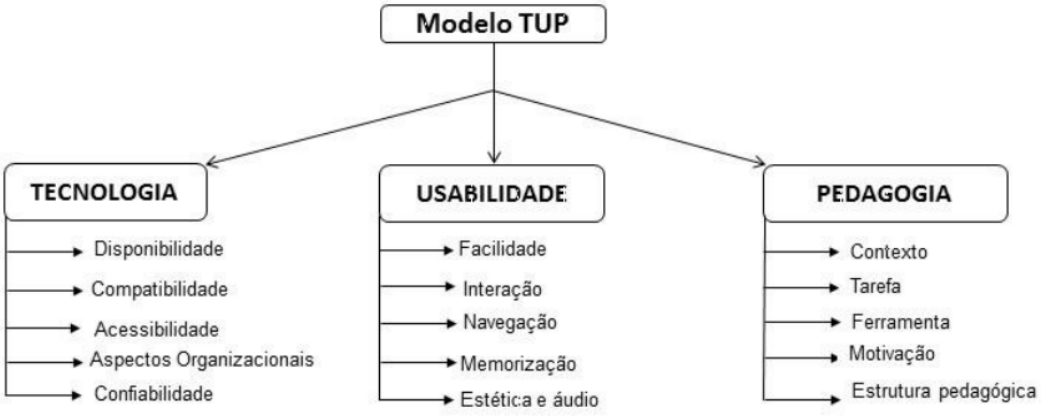


Figura 31: Modelo TUP. Fonte: (de Souza Rezende, 2013)

A partir do modelo TUP, foi desenvolvida uma ferramenta online (TUP Online) (Bednarik, 2013) para que estas avaliações possam ser realizadas, ao mesmo tempo que permite a consulta quanto a avaliações feitas por outros usuários, como demonstra a Figura 32.

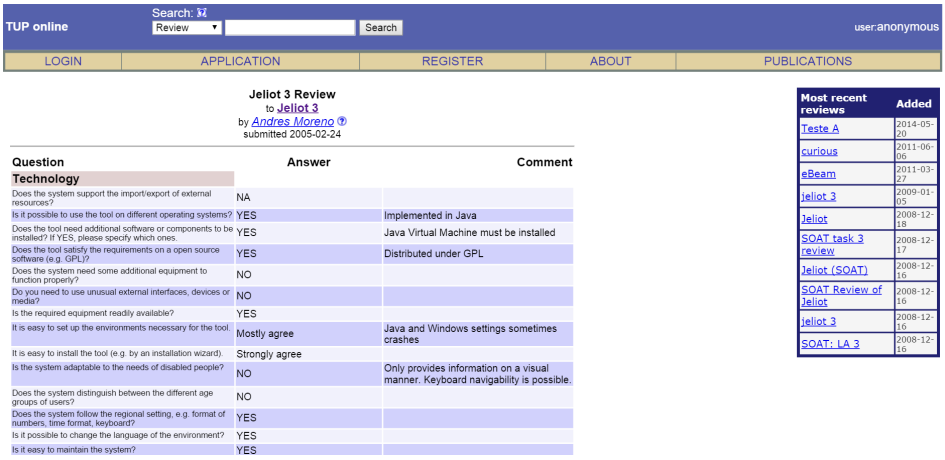


Figura 32: Exemplo de avaliação TUP de software educacional. Fonte: (Bednarik, 2013)

O questionário utilizado no modelo TUP é do tipo checklist, e o usuário é livre para comentar cada um dos atributos, em que se dividem em 26 perguntas quanto a tecnologia, 26 perguntas quanto a usabilidade, 40 perguntas referentes a pedagogia e duas perguntas finais gerais, sobre a recomendação ou não do software e uma visão geral quanto as capacidades educacionais do mesmo.

O autor desse modelo reconhece que a aprendizagem pode ocorrer em todos os três diferentes grupos de teorias de aprendizagem: comportamentalista, através da construção de reforços, tais como regras, estruturas, recompensas, punições, e consequências (as quais, segundo o autor, são facilmente implementáveis em computadores); cognitivista/construtivista, a qual é baseada na interação, no diálogo, na percepção, e processamento do conhecimento; e social, a qual sublinha a importância de ambientes sociais no processo de aprendizagem (Bednarik, 2002).

Entretanto, no que tange ao requisito pedagógico, o autor em (Bednarik, 2002) salienta que as teorias construtivistas, as quais se opõem às behavioristas, trouxeram mudanças na educação, de maneira que o foco principal deixou de ser centrado no professor e passou para o estudante, além de passar de uma aprendizagem mais focalizada para uma mais holística. Portanto, (Bednarik, 2002) conclui que as ferramentas educacionais da modernidade devem seguir essas mudanças, sem que, no entanto, o behaviorismo e outras teorias da aprendizagem sejam totalmente desconsideradas.

Por exemplo, quando se considera o sequenciamento das tarefas e o nível de abstração, o autor em (Bednarik, 2002) expõe que pela linha construtivista todas as atividades de aprendizagem devem ser auto-contidas e independentes de sequência, entretanto em algumas

situações pode existir a necessidade de se seguir alguma ordem definida para a construção de um conhecimento, o que tenderia para uma linha mais comportamentalista de aprendizagem.

Para a construção do modelo TUP, o autor se baseia no fato de que todo sistema educacional tem que corresponder a três principais necessidades dos estudantes: o crescimento, a diversidade, e a motivação. De acordo com (Bednarik, 2002), o crescimento é considerado como o principal objetivo da educação, no qual o conhecimento do educando evolui pela promoção de novas ideias; a diversidade é definida pelas diferenças entre os estudantes, como por exemplo gênero, ambientes, determinações culturais, etc., e essas devem ser levadas em consideração de modo que seja possível acomodá-las; e por fim, a necessidade de motivação que os discentes possuem.

Além disso, o autor também se baseia na consideração de que os ambientes educacionais têm que abranger aspectos referentes ao contexto, tarefas, ferramentas e interfaces, e estes conceitos se apoiam nas linhas construtivistas e socioculturalistas de aprendizagem (Bednarik, 2002). De acordo com o autor, o conhecimento precisa ser apresentado no contexto mais autêntico possível, além de estar alinhado com os objetivos da aprendizagem, o qual respeita os papéis existentes no processo de mesma, permitindo a personalização e customização quando necessário e considerando as diferenças culturais. As tarefas suportadas pelo software têm que ser capazes de levar à completude dos objetivos da aprendizagem, os quais precisam ser definidos de forma clara e ter proximidade com o mundo real, além de preferencialmente manter e aumentar a motivação do estudante durante o processo (Bednarik, 2002).

As ferramentas devem ser capazes de abranger todo o processo de aprendizagem, além de atender diferentes estilos de aprendizagem de estudantes. Se for o caso, o autor argumenta que a ferramenta também deve ajudar os professores a preparar, editar, e compartilhar os materiais de aprendizagem, assim como fornecer diferentes visões sobre o desempenho dos estudantes para facilitar a avaliação. Quanto às interfaces, (Bednarik, 2002) afirma que estas devem ser adaptadas às tarefas executadas e às necessidades dos discentes, além de claramente distinguir entre os diferentes papéis existentes no ambiente de aprendizagem.

De maneira resumida, os atributos que compõem o requisito pedagógico no modelo TUP estão representados na Tabela 4.

Modelo PECTUS O modelo PECTUS foi desenvolvido por (de Souza Rezende, 2013) com o objetivo de apoiar avaliações de softwares educacionais aplicados ao ensino de ciências. De acordo com o autor, este modelo foi construído através dos requisitos do modelo TUP, e seu desenvolvimento foi pautado principalmente nos conceitos definidos pela abordagem GQM (*Goal, Question, Metrics*), o qual representa um processo para a geração de planos de qualidade de software, e também dos requisitos de qualidade relacionados aos Aspectos Pedagógicos (P), de Ensino de Ciências (EC), de Usabilidade (US) e Tecnológicos (T).

Tabela 4: Atributos referentes ao requisito pedagógico do Modelo TUP. Fonte: (de Souza Rezende, 2013)

Atributos	Definições
Contexto	Refere-se aspectos multiculturais e multilíngues, que devem ser capazes de propiciar uma aquisição de conhecimento amplo e com diversidade etno-cultural.
Tarefa	Refere-se às atividades que compõem o ambiente de aprendizado, incluindo os meios, ajustes e condições que conduzem ao alcance do objetivo da aprendizagem.
Ferramentas	Refere-se a instrumentos que possibilitam a compreensão dos processos pedagógicos específicos de aprendizagem
Estrutura pedagógica	Refere-se ao suporte oferecido pelo software para o gerenciamento de materiais instrucionais, que ofereçam perspectivas dos estilos e processos de aprendizagem
Motivação	Refere-se à capacidade do software de despertar o interesse intuitivo, de incentivar a conduzir e a cumprir os objetivos da aprendizagem.

Portanto, os pilares para a estruturação deste modelo se baseiam no modelo TUP, além de utilizar a abordagem GQM para o desenvolvimento do mesmo. O GQM é uma abordagem para avaliar softwares definidos por metas. Ele leva em consideração os objetivos da avaliação (Goal), as questões (Question) associadas a cada objetivo e que constituem o modelo, e as métricas de mensuração (Metrics), que avaliarão cada questão e as quais fornecem as medidas do que está sendo analisado com valores qualitativos e quantitativos (de Souza Rezende, 2013).

Além destes dois focos principais, o modelo também utiliza considerações de qualidade do modelo ISO, características da teoria da engenharia da usabilidade e também de teorias educacionais, psicológicas e científicas Figura 33. Estas teorias, segundo o autor em (de Souza Rezende, 2013), tem a intenção de reforçar o corpo teórico e "justificar os fatores ligados ao indivíduo, às características de seu meio social, e suas atividades".

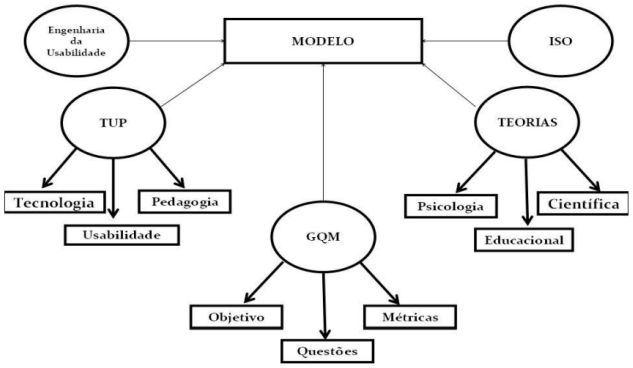


Figura 33: Conceitos do Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

De acordo com o modelo, a primeira definição, a qual segue a abordagem GQM, se refere aos objetivos pretendidos pela avaliação de qualidade do software educacional, e qual o nível de qualidade de software que se deseja atingir em cada aspecto (Pedagógico, Ensino de Ciências, Usabilidade e Tecnológico). Seguindo a abordagem GQM, o autor define as questões

associadas a cada um dos objetivos propostos, sendo, no geral, 40 questões (dez para a avaliação de cada requisito considerado): Aspectos de Pedagogia, Aspectos de Ensino de Ciências, Aspectos de Usabilidade e Aspectos de Tecnologia (de Souza Rezende, 2013). A partir da avaliação dos requisitos, guiada pela análise dos atributos relativos a cada requisito, pode-se chegar à determinação do Nível Geral de Qualidade do Software (NGQS). Para cada uma destas 40 questões, adotou-se uma escala Likert de cinco pontos, para que o participante expresse seu grau de concordância ou discordância quanto à determinado questionamento ou afirmação.

Tabela 5: Requisitos e atributos de qualidade no Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

Requisitos de Qualidade de Software Educacional para o Ensino de Ciências e seus Atributos			
Aspectos de Pedagogia	**Aspectos de Ensino de Ciências**	**Aspectos de Usabilidade**	**Aspectos de Tecdnologia**
Atributos: - Afetividade - Flexibilidade - Carga Cognitiva - Confiabilidade Conceitual - Suporte à Colaboração - Objetividade - Apoio ao professor - Controle por parte do Estudante - Motivação - Acomodação das Diferenças Individuais	**Atributos:** - Apoio à construção de Conceitos - Suporte para a Aplicação de Conceitos - Apoio a Aprendizagem Evolutiva - Suporte Empírico - Associação entre Teoria e o Mundo Real - Apoio à Representação de Teoria e Conceitos - Precisão dos Cálculos e Resultados - Rigor Científico - Clareza dos Procedimentos - Suporte para a Resolução de Problemas	**Atributos:** - Adequação de Software - Facilidade de Aprendizagem - Operacionalidade - Suporte à Memorização - Proteção aos Erros - Clareza das Informações - Acessibilidade - Qualidade do Design - Satisfação do Usuário - Funcionalidade Geral	**Atributos:** - Exigências de Equipamentos - Exigência de Software - Disponibilidade - Segurança - Confidencialidade - Comportamento em Relação ao Tempo - Portabilidade - Adaptabilidade - Mapeamento das Ações do Usuário - Facilidade de Instalação

Referente aos aspectos pedagógicos, o autor define que o modelo PECTUS se baseou, a níveis psicológicos em trabalhos definidos por Jean Piaget, e a níveis educacionais em trabalhos referentes a Paulo Freire (de Souza Rezende, 2013). A tabela 5 demonstra as definições dos atributos quanto ao requisito pedagógico. Após obter os valores anotados nas escalas do(s) participante(s), referentes a cada uma das 40 questões que compõem o modelo de avaliação, como demonstrado pelo exemplo da Figura 35, se calcula o valor atribuído pelo(s) mesmo(s) a cada um dos quatro requisitos incorporados ao modelo. Para este cálculo é utilizada a média aritmética e outros cálculos estatísticos, definindo-se um valor referente ao nível de qualidade para cada um dos requisitos do modelo, e um valor para o resultado geral da avaliação do software educacional.

Requisito "Aspectos de Pedagogia" e seus Atributos	
Atributos	**Definição**
Afetividade (Baixa - Alta)	Refere-se à explicitação de aspectos e comportamentos físicos e psicológicos, capazes de indicar o envolvimento do usuário, quando ele utiliza o software, tais como: emoção, estados de humor, motivação, ansiedade, sentimentos de raiva, desinteresse, prazer, alegria, etc.
Flexibilidade (Baixa - Alta)	Refere-se à capacidade intrínseca que o software tem de acomodar mudanças suportadas pela funcionalidade e de que maneira influência o tipo de ensino e aprendizagem, tais como: autoaprendizagem, objetivismo, construtivismo, etc.
Carga Cognitiva (Baixa – Alta)	Refere-se ao esforço mental requerido durante a execução das tarefas no software, como exploração dos conteúdos, uso da estrutura, respostas demandadas, etc.
Confiabilidade Conceitual (Baixa – Alta)	Refere-se à capacidade do software em despertar reações e comportamentos que expressam confiança nos seus conteúdos e resultados por ele propiciados.
Suporte à Colaboração (Baixo – Alto)	Refere-se ao apoio fornecido pelo software à realização de atividades de forma colaborativa, apoiando o compartilhamento de conhecimento e o desenvolvimento de habilidades sociais.
Objetividade (Baixa - Alta)	Refere-se à forma de funcionamento do software e dos procedimentos nele incorporados, ou seja, os quão bem definidos e padronizados eles são.
Apoio ao professor (Baixo – Alto)	Refere-se ao nível de apoio que o software oferece ao professor, que lhe permitirá atuar como provedor de informações e/ou de facilitador da aprendizagem.
Controle por parte do Estudante (Baixo – Alto)	Refere-se à possibilidade oferecida pelo software aos usuários, para este definir como explorar os módulos e conteúdos, ou seja, decidir que seções estudar, que caminhos seguir, que material utilizar e a ordem envolvida nessas decisões.
Motivação (Baixa – Alta)	Refere-se à capacidade de software em, por si só, motivar os usuários a explorar temas e conceitos, por meio de elementos como recursos. Multimídia, interação de boa qualidade, etc.
Acomodação das Diferenças Individuais (Baixa – Alta)	Refere-se à capacidade do software em considerar e facilitar a acomodação de diferenças individuais dos estudantes, ou seja, reforça a heterogeneidade em termos de atitudes, conhecimento e experiência anteriores, estilos de aprendizagem, etc.

Figura 34: Atributos referentes aos Aspectos Pedagógicos do Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

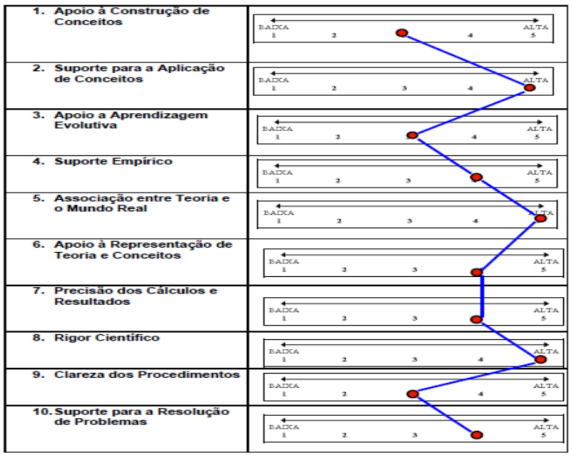


Figura 35: Exemplo de qualidade percebida para avaliação dos atributos de ensino de ciências. Fonte: (de Souza Rezende, 2013)

Modelo ProInfo O modelo de Avaliação da Qualidade de Software Educacional foi desenvolvido por de Barros Ribeiro (2013c), e está disponível na forma de ferramenta online em Barros Ribeiro (2013b), cuja proposta é avaliar os softwares educacionais com base nas diretrizes estipuladas pelo ProInfo quanto ao uso pedagógico das tecnologias de informação e comunicação (TIC).

De acordo o autor em de Barros Ribeiro (2013c), a Informática Educacional tem por objetivo auxiliar na construção do conhecimento, utilizando o computador como recurso para as práticas pedagógicas nos diversos componentes curriculares. (Ribeiro, 2013c) enfatiza que o planejamento de questões quanto a "quando", "porque", e "como" usar a informática se faz essencial para uma contribuição efetiva destes recursos no meio educacional. O "quando" se dá uma vez que determinado claramente os objetivos a serem atingidos, além de estes estarem claros também para os estudantes; o "por que" trata de o quão siginificativa de fato será a aprendizagem proporcionada pela situação a ser criada pelo recurso, e o "como" considera os caminhos para se atingir os resultados desejados, ao mesmo tempo em que determina se os recursos a serem utilizados são os mais adequados ou não.

Outro aspecto importante, que fundamenta o modelo desenvolvido e é enfatizado pelas diretrizes do ProInfo, é o foco no papel do educador como mediador, cujo objetivo é orientar o educando, intervindo para a contribuição do "desenvolvimento de valores pessoais e sociais, levando a construção de uma consciência critica" (Ribeiro, 2013c). Ou seja, a finalidade principal é a formação para a cidadania, a qual permite a avaliação de atitudes e escolhas com base no desenvolvimento de uma consciência crítica sobre o mundo e o contexto do educando.

Definidas estas bases, o autor em de Barros Ribeiro (2013d) determina que os softwares educacionais têm portanto o objetivo de propiciar o desenvolvimento de situações de aprendizagem as quais permitam a construção do conhecimento, enfatizando a habilidade de "aprender a aprender". Além de viabilizar a construção de conceitos de maneira a desenvolver as capacidades cognitivas, Barros Ribeiro (2013c) estabelece outros aspectos importantes na avaliação destes softwares, como a visão interdisciplinar, o respeito ao tempo de desenvolvimento de cada educando, a satisfação das necessidades de convivência em grupo e o tratamento de erros de forma construtiva.

O modelo enfatiza portanto a (re)construção do conhecimento, de modo que, contrariamente à mera transmissão de conhecimentos prontos, o educando se torna figura central, sendo ele o principal interessado no aprimoramento das estratégias de construção do seu saber através de um ensino interativo (Santos and Marques, 2009).

A avaliação do software educacional quanto a este quesito ocorre através da questão de número seis referente ao questionário de aspectos pedagógicos: "Através do fornecimento de feedback permite que o aprendiz construa seu conhecimento a partir da ação-reflexão-ação?". O conceito de ação-reflexão-ação é entendido pela reflexão sobre e na ação, o que produz, de acordo com Micheletto (2007), "conhecimentos competentes, autênticos, o saber fazer oriundo de realidades flexíveis e incertas". Em citação de Freire sobre ação e reflexão, este a define como fundamental para mudança consciente no mundo: "Os homens são seres da práxis. São seres do que fazer... Se os homens são seres do quefazer é exatamente porque

seu fazer é ação e reflexão. É práxis. É transformação do mundo" (Santos and Marques, 2009). Ainda por Freire: "É pensando criticamente a prática de hoje ou de ontem que se pode melhorar a próxima prática" (Micheletto, 2007).

A ação-reflexão-ação se relaciona também com a teoria de prática reflexiva de Schön para a formação de um profissional reflexivo (Micheletto, 2007), composto por: a reflexão-na-ação, que pode ser traduzido no saber-fazer, o conhecimento técnico ou solução do problema, e se dá através de um processo de "diálogo com a situação problemática e sobre uma interação particular que exige uma intervenção concreta" (Santos and Marques, 2009), ou seja, se traduz por um saber que está presente nas ações dos profissionais; a reflexão-sobre-a-ação, que, de acordo com (Micheletto, 2007), é a reconstrução mental que acontece durante a retrospectiva da ação realizada, na tentativa de analisá-la; por fim a reflexão-sobre-a-reflexão-na-ação, que seria a "a análise que o indivíduo realiza a posteriori sobre as características e processos da sua própria ação" (Micheletto, 2007), ou seja, através de processos como descrição, análise e avaliação sobre os produtos resultantes (armazenados na memória) de intervenções do passado, se faz possível a atualização do conhecimento do indivíduo.

O modelo PROINFO apresenta um total de 20 perguntas na forma de *checklist*, o qual considera também os aspectos técnicos do software educacional. Os aspectos pedagógicos são abordados por meio de 10 perguntas, conforme apresentado na Figura 36.

Aspectos Pedagógicos	Ótimo	MBom	Bom	Regular	Ruim
1-Proporciona um ambiente interativo entre aluno e o *software*?	○	○	○	○	○
2-Permite uma fácil exploração?(seqüencial, não linear)	○	○	○	○	○
3-Apresenta conceitos de forma clara e correta?	○	○	○	○	○
4-Desperta o interesse do aluno, sem perder de vista os objetivos do *software* e do usuário?	○	○	○	○	○
5-Possui vocabulário adequado apresentando texto de qualidade?	○	○	○	○	○
6-Através do fornecimento de feedback permite que o aluno construa seu conhecimento a partir da ação-reflexão-ação?	○	○	○	○	○
7-Apresenta uma visão interdisciplinar?	○	○	○	○	○
8-Atende a diferentes níveis de conhecimento?	○	○	○	○	○
9-Promove o desenvolvimento do raciocínio e descoberta do aluno?	○	○	○	○	○
10-Permite atividades em grupo?	○	○	○	○	○

Figura 36: Avaliação de aspectos pedagógicos de software educacional. Fonte: (Ribeiro, 2013b)

4 Desenvolvimento

4.1 Seleção da Visão Metodológica de Ensino

Ao se analisar as bases do projeto do curso de BSI e as diretrizes nacionais para o uso da informática na educação, é possível identificar, como esperado, similaridades que as fazem convergir quanto ao entendimento sobre a tecnologia na educação e consequentemente, seu papel na sociedade.

Como anteriormente demonstrado nesse trabalho na seção 5.1 pelos autores em (Melo et al., 2010), o aspecto de maior importância quanto às diretrizes é a necessidade da tecnologia ser utilizada para a construção de conhecimentos que levem à transformação, de maneira a promover uma sociedade mais participativa, crítica e igualitária. Ou seja, a principal preocupação é a utilização da tecnologia à serviço da cidadania. Esta preocupação é também enfatizada pela visão humanista do projeto do curso de BSI, na qual visa uma formação integral e mais consciente dos discentes, para que sejam capazes de lidar com a diversidade de maneira crítica. Desta forma, podem se tornar profissionais reflexivos, possuindo autonomia e iniciativa para propor soluções para os problemas que se apresentem no mundo.

Este objetivo se sustenta também nas outras bases estipuladas pelo projeto do curso (Graeml et al., 2008), na qual a flexibilidade possibilita mais liberdade e consequentemente autonomia através da articulação entre diversos campos do saber; a interdisciplinaridade contribui para a percepção de novos relacionamentos entre estes diversos campos; e a integração rompe com a visão fragmentada e isolada de mundo. Estas bases são também contempladas pelas diretrizes nacionais, uma vez que entende que as TIC, pelo fato de poderem potencializar a articulação do conhecimento das diversas áreas, podem contribuir para a integração das disciplinas e também permitir que estudantes e professores se envolvam em atividades socialmente relavantes e significativas.

Portanto, para a seleção da visão metodológica de ensino a ser utilizada, além de levar em consideração os aspectos já citados, também há de se considerar o fato das diretrizes estipularem o baseamento em uma filosofia de aprendizagem que proporcione aos estudantes oportunidades de interação, visando a construção do conhecimento. Além disso, também é recomendável que os estudantes tragam para a aula a sua realidade vivida, ou seja, se faz necessário a contextualização da aprendizagem, a qual é entendida como resultante da interação com o meio social, e em que o papel do professor é o de mediador didático-pedagógico.

À vista disso, é possível a união de mais de uma visão metodológica de ensino para abranger as expectativas e orientações apresentadas quanto ao uso da tecnologia na educação, pelo fato de estas serem complementares. Como ponto de partida, o construtivismo de Jean Piaget é destacado, uma vez que visa a construção do conhecimento, o "aprender a aprender", no qual, para se fornecer um ambiente de aprendizagem ideal, os educandos devem ter

a possibilidade de construir o conhecimento conforme julguem significativo. Ou seja, para Piaget, a aprendizagem só se realiza quando o estudante elabora o seu conhecimento, e em sua perspectiva, a inteligência é o instrumento de aprendizagem mais necessário para isso (Mizukami, 1986). Portanto, a visão de construção do conhecimento de Piaget dá ênfase ao aspecto cognitivo, em que "conhecer é fabricar mentalmente o objeto a ser conhecido, dependendo da familiaridade do novo conhecimento com os anteriores, da fase de desenvolvimento em que se encontra o sujeito e da própria experiência de aprendizagem" (Bahia, 2007).

Entretanto, de acordo com (Mizukami, 1986), apesar do respeito ao estudante quanto à sua própria atividade, ou seja, como ele irá trabalhar os conceitos e oportunidades de investigação, cabe enfatizar que esta abordagem é, em prima instância, individualista: "a forma de solução de cada problema é pertinente apenas a cada estudante e a ele caberá encontrá-la". Ou seja, na construção do conhecimento por Piaget a verdade se encontra no interior do indivíduo e é impossível de ser transmitida pois, conforme aponta (Treviso and de Almeida, 2014), esta nada mais é do que a organização do real pelo pensamento humano, ou seja, um processo interno ao sujeito.

Uma vez que a construção do conhecimento piagetiano é um processo interno e isolado, ou seja, pautado por processos intrassubjetivos, esta visão passa a ser conflituosa ao se pensar as relações sociais, pois se baseia em um "modelo biológico de desenvolvimento dos seres vivos e não no desenvolvimento humano histórico e social" (Treviso and de Almeida, 2014). Desta forma, Seymour Papert, que trabalhou com Piaget por vários anos, introduz o conceito de construcionismo (Papert and Harel, 2002). Baseado no construtivismo piagetiano, a principal diferença do construcionismo é o fato que, para a construção do conhecimento acontecer de forma mais positiva, os estudantes devem construir e compartilhar objetos publicamente, ou seja, os estudantes precisam estar conscientemente engajados na construção de uma entidade pública, o que possibilita uma interação social crítica (Papert and Harel, 2002):

> "A construção que acontece 'na cabeça' muitas vezes acontece especialmente de maneira feliz quando é apoiada pela construção de um tipo mais público 'no mundo' - um castelo de areia ou um bolo, uma casa de Lego ou uma corporação, um programa de computador, um poema ou uma teoria do universo. Parte do que eu quero dizer com 'no mundo' é que o produto pode ser mostrado, discutido, examinado, sondado, e admirado. [...] Atribui-se especial importância ao papel das construções no mundo como um apoio para aquelas na cabeça, tornando-se assim mais do que apenas uma doutrina puramente mental". Traduzido de (Blinkstein, 2004, p. 4)

Desta forma, além da abordagem construcionista quanto a construção do conhecimento, a visão sociocultural (Mizukami, 1986) do ensino também se faz adequada uma vez que visualiza a educação como meio para a formação da cidadania. Como já abordado, nesta

linha interacionista (Santana, 2010) o educando, ao interagir com o aprendizado por meio de suas experiências no cotidiano, consegue obter uma visão crítica do conhecimento transmitido a ele, o que forma indivíduos mais politicamente conscientes. Portanto, esta linha considera imprescindível a interação homem-mundo para o desenvolvimento do indivíduo e para que esse se torne sujeito de sua práxis, o que leva o homem a desenvolver um pensamento crítico sobre o que o rodeia para enfim atingir a libertação do seu pensamento e poder participar ativamente da história. Esta participação, portanto, se dá pela desmitificação do pensamento uma vez que o homem se torna reflexivo e consciente.

Freire é o principal representante da corrente sociocultural, e define que os estudantes podem ir da "consciência do real"para a "consciência do possível"na medida em que percebem as "novas alternativas viáveis"existentes para além das situações limitantes. Ou seja, para Freire (Blinkstein, 2004), o caminho para emancipação e humanização é perceber-se como agente ativo de mudança, além de perceber o mundo como uma entidade mutável.

Portanto, para participar ativamente na leitura e mudança do mundo, são precisas ferramentas, e apesar da linguagem ser a principal utilizada neste processo por Freire, ela não é necessariamente o único veículo de mudança (Blinkstein, 2004). É neste ponto em que o construcionismo de Papert age em favor da emancipação prevista por Freire, o qual se utiliza, de acordo com (Blinkstein, 2004), da noção vygotskiana de aprendizagem por meio da comunicação aplicada ao projeto e desenvolvimento de dispositivos (ou tecnologias) que tenham um alto grau de significância pessoal. Desta forma, a escolha da visão metodológica de ensino neste trabalho surge a partir desta intersecção, em que a tecnologia, quando baseada na visão sociocultural de ensino e suportada de maneira construcionista, pode criar um ambiente fértil para uma aprendizagem mais significativa.

4.2 Seleção do Método de Avaliação de Softwares Educacionais

Ao se selecionar o método de avaliação de software educacional, optou-se por avaliar os atributos dos três modelos apresentados em TUP (Bednarik, 2002), PECTUS (de Souza Rezende, 2013) e Avaliação de Software Educacional (Ribeiro, 2013b) quanto ao que melhor se adapta conforme às necessidades pedagógicas estipuladas.

Foi possível encontrar conjuntos de atributos semelhantes em cada modelo, visto que, conforme já apresentado, os três se apoiam em bases pedagógicas semelhantes: o TUP nas linhas construtivistas e socioculturalistas de aprendizagem (Bednarik, 2002), o PECTUS a níveis psicológicos em trabalhos definidos por Piaget e a níveis educacionais em trabalhos referentes à Freire (de Souza Rezende, 2013) e o de Avaliação de Software Educacional nas diretrizes estipuladas pelo ProInfo (Ribeiro, 2013b) (que também enfatizam a construção do conhecimento com tendências construtivistas e socioculturalistas).

Portanto, para seleção do método de avaliação de software mais adequado se faz preciso

uma análise dos formulários que definem o requisito pedagógico, para desta forma ser possível obter mais clareza quanto aos objetivos de cada modelo. Um ponto importante é o fato de que, apesar dos diferentes modelos tratarem muitas vezes do mesmo aspecto, estes podem se dar de maneiras distintas.

A primeira diferença notada é que o modelo TUP coloca as questões de forma mais direta e objetiva, enquanto PECTUS e Avaliação de Software Educacional fazem o mesmo de maneira mais generalizada. Este fato se dá também pela quantidade de questões pedagógicas que cada modelo propõe: 40 no TUP, 20 no PECTUS (se considerado também as 10 questões referentes ao ensino de ciências) e 10 no de Avaliação de Software Educacional.

Para comparação dos modelos, separou-se os atributos em 11 grupos (definidos por cores), e relacionou-se os aspectos pertencentes quanto aos mesmos em comum nos três modelos, conforme apontam as Figuras . Os grupos definidos foram: contexto educacional, diferenças individuais, clareza/credibilidade, motivação/envolvimento, associação ao mundo real, representação do conhecimento, suporte ao professor, suporte à colaboração, controle do estudante, adaptação/flexibilidade e objetivos de aprendizagem.

A diferença dos três modelos é perceptível ao analisar o aspecto referente a adaptação da ferramenta educacional às diferenças individuais do estudante. No modelo TUP, as seguintes questões são propostas (variando de acordo com a escala Likert): "O ambiente é auto-adaptável de maneira a refletir o desenvolvimento do estudante?", "O sistema permite a customização dos objetos de aprendizagem de acordo com as necessidades do estudante?", "Questões culturais são abordadas e propriamente tratadas pelo ambiente?", "A sequência de tarefas é adaptável de acordo com o desenvolvimento do estudante?", "O nível de abstração corresponde com o desenvolvimento do estudante?", "A ferramenta permite diferentes estilos de aprendizagem (por exemplo aprendizagem por exploração, aprendizagem por realização de tarefas, etc.)?".

No modelo PECTUS estas questões são analisadas pelo aspecto pedagógico, que se generaliza em: "Acomodação das Diferenças Individuais (Baixa - Alta): Refere-se à capacidade do software em considerar e facilitar a acomodação de diferenças individuais dos estudantes, ou seja, reforça a heterogeneidade em termos de atitudes, conhecimento e experiência anteriores, estilos de aprendizagem, etc". E no modelo de Avaliação de Software esta questão é unicamente abordada a partir da questão de número 8 (Ótimo - Ruim): "Atende à diferentes níveis de conhecimento?".

Outro aspecto importante é o destaque, dentro da abordagem construcionista, da criação, debate e compartilhamento público dos artefatos. O modelo TUP aborda este fator através dos seguintes questionamentos: "A ferramenta facilita o trabalho em grupo?", "A ferramenta suporta a criação e compartilhamento de artefatos de aprendizagem?", "A ferramenta suporta atividades de aprendizagem independentes do computador (por exemplo, motiva dis-

cussões)?", "É possível que os usuários preparem apresentações utilizando este ambiente?".

No modelo PECTUS estas questões são abordadas de maneira geral por meio do aspecto: "Suporte à Colaboração (Baixo - Alto): Refere-se ao apoio fornecido pelo software à realização de atividades de forma colaborativa, apoiando o compartilhamento de conhecimento e o desenvolvimento de habilidades sociais.". No modelo de Avaliação de Software Educacional essa questão é endereçada através da questão de número 10 (Ótimo - Ruim): "Permite atividades em grupo?".

Outra diferença notada é que o modelo TUP realiza a avaliação considerando a adequação daquele software quanto aos objetivos de aprendizagem que o educando busca. Isto se dá com perguntas que pretendem verificar se a ferramenta e as informações fornecidas por esta são suficientes para atingir os mesmos: "A seleção dos objetivos de aprendizagem são apropriados?", "O layout da interface consegue atingir os objetivos de aprendizagem?", "O layout da interface contém todas as informações necessárias para se atingir os objetivos da aprendizagem?", "Você atinge os objetivos de aprendizagem com a ferramenta?".

Já no modelo PECTUS, por se dividir em questões não apenas pedagógicas mas também voltadas ao ensino de ciência, os objetivos da ferramenta estão parcialmente pré-definidos uma vez que se verificam aspectos como: apoio à construção de conceitos, suporte para aplicação de conceitos, suporte empírico, precisão dos cálculos e resultados, clareza de procedimentos, suporte para resolução de problemas e rigor científico. Por exemplo, em PECTUS a descrição do aspecto de "Suporte para resolução de problemas" tem como explicação o objetivo de "promover o raciocínio crítico e analítico". Ou seja, este em si já é auto-definido como um objetivo da aprendizagem. O mesmo se dá para o modelo de Avaliação de Software Educacional, posto que os objetivos da aprendizagem já estariam implícitos no próprio questionário proposto para a avaliação (como por exemplo, a "construção do conhecimento através da ação-reflexão-ação" ou "desenvolvimento do raciocínio e descoberta").

Portanto, o modelo TUP de avaliação de software educacional é o selecionado para avaliar as ferramentas abordadas neste trabalho, pois além de contemplar os aspectos educacionais desejados e também abordados nos outros dois modelos, este se mostra mais objetivo e específico, além de também flexível quanto aos objetivos, o que clarifica a compreensão e diferenciação quanto aos aspectos pedagógicos das ferramentas analisadas.

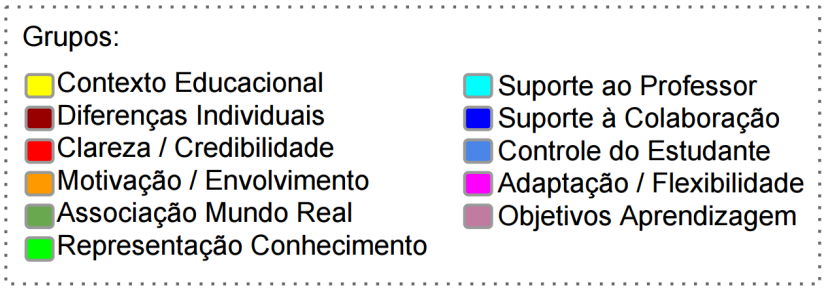


Figura 37: Grupos referentes às Práticas Educacionais para classificação dos modelos.

TUP

- Is the subject of learning presented in an authentic context?
- Usage of the tool fits into the learning setting (e.g. class versus individual learning)
- The tool conveniently complements other class activities.
- The content fits into the institutional curriculum.
- Does the system distinguish between the roles of the users in the learning process?
- Is the environment self-adaptable in order to reflect learners' growth?
- Does the system enable the customization of the perspectives of learning objects according to the learners' needs?
- Cultural issues are addressed and properly handled by the environment.
- Are the resources and references used in the tool credible?
- Is the information presented by the tool up to date?
- Do you feel the tool maintains trust in it?
- Learners' ownership of learning is maintained by the environment.
- Are the tasks designed to maintain (or increase) learners' motivation?
- Is the selection of the learning goals appropriate?
- Is it possible to modify the learning goals?
- Does tool enable multiple paths to achieve the learning goals?
- Are the task sequences adaptable according to the learners' growth?
- The level of abstraction corresponds with the learners' growth.
- Learning tasks correspond with the real world matching.
- Does the system provide enough means for the proper knowledge representation?
- Does the environment provide multiple representation of knowledge?
- Can the teacher prepare and modify learning materials using the tool?
- Can the teachers share learning materials using the tool?

PECTUS

Requisito "Aspectos de Pedagogia" e seus Atributos	
Atributos	**Definição**
Afetividade (Baixa - Alta)	Refere-se à explicitação de aspectos e comportamentos físicos e psicológicos, capazes de indicar o envolvimento do usuário, quando ele utiliza o software, tais como: emoção, estados de humor, motivação, ansiedade, sentimentos de raiva, desinteresse, prazer, alegria, etc.
Flexibilidade (Baixa - Alta)	Refere-se à capacidade intrínseca que o software tem de acomodar mudanças suportadas pela funcionalidade e de que maneira influência o tipo de ensino e aprendizagem, tais como: autoaprendizagem, objetivismo, construtivismo, etc.
Carga Cognitiva (Baixa – Alta)	Refere-se ao esforço mental requerido durante a execução das tarefas no software, como exploração dos conteúdos, uso da estrutura, respostas demandadas, etc.
Confiabilidade Conceitual (Baixa – Alta)	Refere-se à capacidade do software em despertar reações e comportamentos que expressam confiança nos seus conteúdos e resultados por ele propiciados.
Suporte à Colaboração (Baixo – Alto)	Refere-se ao apoio fornecido pelo software à realização de atividades de forma colaborativa, apoiando o compartilhamento de conhecimento e o desenvolvimento de habilidades sociais.
Objetividade (Baixa - Alta)	Refere-se à forma de funcionamento do software e dos procedimentos nele incorporados, ou seja, os quão bem definidos e padronizados eles são.
Apoio ao professor (Baixo – Alto)	Refere-se ao nível de apoio que o software oferece ao professor, que lhe permitirá atuar como provedor de informações e/ou de facilitador da aprendizagem.
Controle por parte do Estudante (Baixo – Alto)	Refere-se à possibilidade oferecida pelo software aos usuários, para este definir como explorar os módulos e conteúdos, ou seja, decidir que seções estudar, que caminhos seguir, que material utilizar e a ordem envolvida nessas decisões.
Motivação (Baixa – Alta)	Refere-se à capacidade de software em, por si só, motivar os usuários a explorar temas e conceitos, por meio de elementos como recursos. Multimídia, interação de boa qualidade, etc.
Acomodação das Diferenças Individuais (Baixa – Alta)	Refere-se à capacidade do software em considerar e facilitar a acomodação de diferenças individuais dos estudantes, ou seja, reforça a heterogeneidade em termos de atitudes, conhecimento e experiência anteriores, estilos de aprendizagem, etc.

Figura 38: Classificação TUP e PECTUS conforme os grupos estabelecidos.

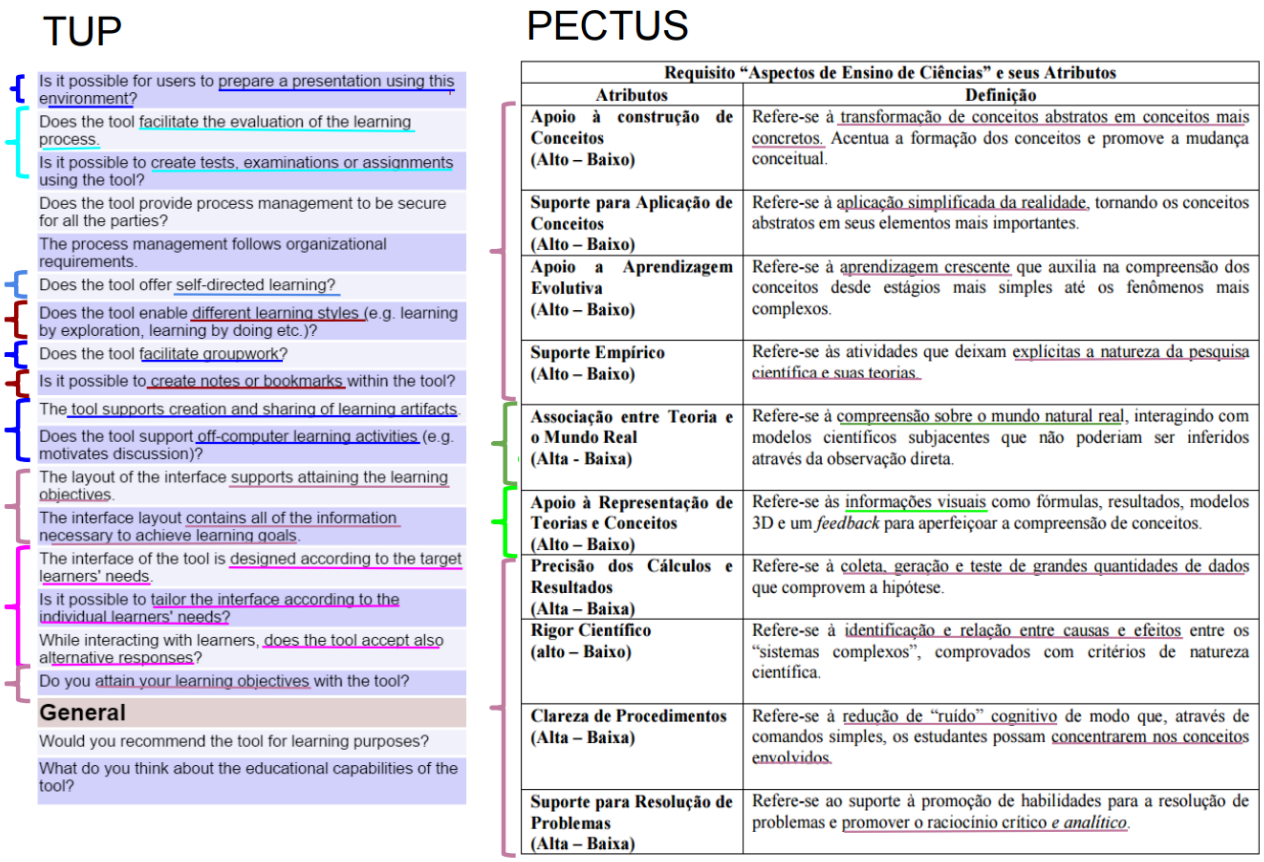


Figura 39: Classificação TUP e PECTUS conforme os grupos estabelecidos (continuação).

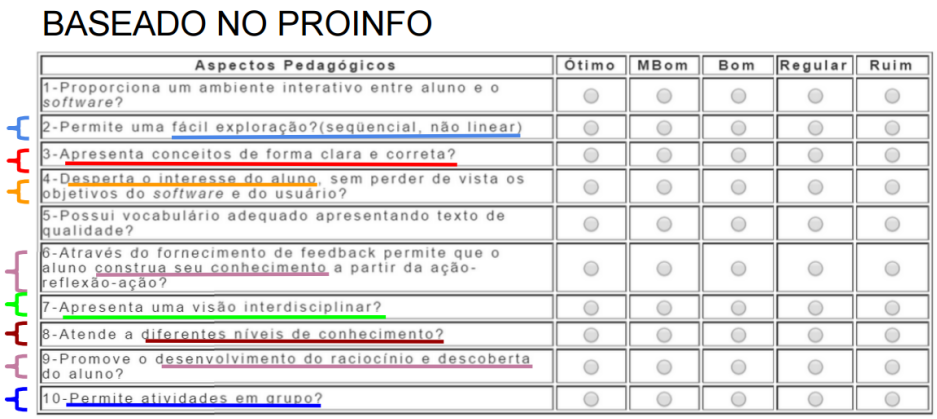


Figura 40: Classificação Modelo Baseado no ProInfo conforme os grupos estabelecidos.

4.3 Seleção dos Softwares Educacionais

A partir das ferramentas de ensino de programação apresentadas neste trabalho, optou-se por selecionar e avaliar um conjunto de três ferramentas pertencentes à categoria construtivista em prol da comportamentalista, uma vez que esta se adequa mais à visão metodológica de ensino escolhida.

Como já abordado, nesta categoria, as ferramentas apresentam o conceito de "micromundo informático", o qual além de possibilitar o exercício das capacidades cognitivas dos estudantes, também leva em conta os conhecimentos prévios dos mesmos, bem como seus interesses, expectativas e ritmos de aprendizagem (Pears et al., 2007). Além disso, estes micromundos tem por objetivo diminuir a distância entre os modelos mentais dos estudantes e a linguagem de programação.

As três ferramentas selecionadas dentre o conjunto contrutivista apresentado neste trabalho foram: Alice (Cooper et al., 2003), Pygame(Shinners, 2015) e Scratch(John Maloney et al., 2010). Estas três se destacam na medida em que são flexíveis quanto ao desenvolvimento de histórias animadas e jogos, com vizualização gráfica atrativa e imediata do código - sem necessariamente seguir um roteiro ou ambiente pré-definido, como RoboCode (IBM, 2003), ou depender de um únco personagem, como Karel the Robot (Pattis, 2015), GreenFoot(Canterbury, 2015) ou Jeroo (Nebraska, 2015).

Tendo em vista o fato de utilizar a linguagem LOGO (John Maloney et al., 2010), a qual foi desenvolvida por Seymour Papert com base na visão construcionista de ensino, a ferramenta educacional Scratch é uma das selecionadas para avaliação. Como já levantado, o aprendizado de programação no Scratch se dá por meio da criação de histórias animadas e jogos, no qual, para ajudar seus usuários a tornar seus projetos motivadores e significativos, facilita a importação ou criação de diversos tipos de mídias, além de proporcionar um contexto social para os usuários, os quais podem receber feedbacks dos projetos compartilhados, apoio de outros usuários e aprendizados advindos de projetos alheios.

Outra ferramenta educacional escolhida para avaliação é o Alice (Mellon, 2015), a qual visa a introdução da programação orientada à objetos com a criação de animações e histórias por meio de jogos interativos. Com uso de programação de animações em 3D, Alice tem o objetivo de satisfazer as expectativas iniciais destes estudantes ao cobrir tópicos abstratos de programação de maneira mais atrativa, o que contribui para a motivação e persistência dos mesmos.

Por fim, o Pygame (Shinners, 2015) foi escolhido por utilizar a linguagem Python (Craven, 2015) de programação, a qual, conforme apontado anteriormente, é considerada por alguns autores (Rebouças et al., 2010) (Marques et al., 2010) ideal para introdução da programação. Algumas de suas características são: a facilidade no desenvolvimento de jogos, sintaxe simples e flexível através de feedback imediato e módulos fáceis de usar.

A avaliação das três ferramentas selecionadas foi realizada por meio do questionário TUP para os aspectos tecnológicos, de usabilidade e pedagógicos, e seu resultado é apresentado no Apêndice A.

4.4 Contextualização Histórica da Oficina

As oficinas de programação em Python, destinadas a estudantes do ensino médio, inicialmente foram promovidas pelo grupo de estudantes que fazem parte do Programa de Educação Tutorial[8]- Computando Culturas em Equidade (PET- CoCe) para promover o incentivo à extensão.

Portanto, estas oficinas se originaram como atividade desse grupo, e anteriormente foram instruídas por outros estudantes, que em cada cersão da oficina utilizaram uma abordagem condizente com a prática com a qual o estudante responsável estava melhor adaptado, no que tange a transmissão de conteúdos e ensino de práticas de programação. Desta forma, estas oficinas visavam oferecer aos estudantes do ensino médio motivação e incentivo à área de tecnologia, através da visão de conceitos básicos que envolviam a computação.

De maneira geral, estas oficinas visavam abordar um público alvo de estudantes do ensino médio, e estudantes com altas habilidades e indícios de superdotação com idades entre 10 e 16 anos, variando o nível de escolaridade apresentado por estes conforme a edição da oficina. É importante ressaltar que não era requerido o conhecimento prévio na área de programação para participação.

Em 2012 a oficina de programação em Python foi realizada com a colaboração de dois estudantes, identificados neste trabalho como Instrutor 1 e Instrutor 2, participantes do grupo PET-CoCe e no período, instrutores da oficina. De acordo com o planejamento da oficina, a mesma teve início no dia 15/08/12, e se desenvolveu utilizando conceitos de IDLE3, strings, fluxogramas e reaproveitamento de código, sendo direcionada para o desenvolvimento de jogos. Além disso, contou com a utilização de alguns destes jogos para aproveitamento e assimilação do conteúdo, como o jogo "Adivinhe o número", "O reino do dragão", e o "Jogo da forca".

Em 2013 a edição da oficina de programação em Python foi instruída pelos estudantes identificados como Instrutor 2 e Instrutor 3, na época participantes do grupo PET-Coce, e contou com a participação de 15 estudantes, que foram convidados para participar da oficina por serem estudantes que obtiveram melhores resultados na olimpíada de matemática, e alguns por pertencerem ao programa de altas habilidades e superdotação vinculado ao Instituto de Educação do Paraná Professor Erasmo Pilotto.

[8]O PET (Programa de Educação Tutorial) é desenvolvido por grupos de estudantes, com tutoria de um docente, organizados a partir de formações em nível de graduação nas Instituições de Ensino Superior do País orientados pelo princípio da indissociabilidade entre ensino, pesquisa e extensão e da educação tutorial (Educação, 2013).

A fim de levantar dados sobre as oficinas realizadas, foi elaborado um questionário direcionado aos instrutores (Apêndice G), para que estes esclarecessem questões quanto à abordagem utilizada pelos mesmos e comportamento dos estudantes durante o período que a oficina foi ministrada.

Esse questionário foi respondido pelo Instrutor 2, o qual fez parte da oficina de programação em Python nos anos de 2012 e 2013. Quando questionado a respeito da utilização de alguma corrente filosófica de ensino na aplicação da oficina, o estudante revelou não dominar o assunto para poder avaliar a utilização da mesma, porém descreveu a prática de ensino utilizada, em que de acordo com este "O planejamento da oficina foi focado no desenvolvimento de atividades práticas de programação, sempre com um período inicial curto de teoria no começo de cada aula." Porém, o instrutor salientou que, durante o planejamento, esperou-se que os estudantes praticassem as atividades em casa.

Quando questionado sobre a motivação dos estudantes, o Instrutor 2 também informou que estes em sua maioria não estavam motivados, e por não possuírem o hábito de estudo fora das salas de aula, como era esperado pelos instrutores, encontraram dificuldades. Além disso, como afirma este, poucos dos estudantes encaravam as dificuldades como desafios e se motivavam a pesquisar sobre o assunto na internet ou em demais meios.

O nível de desistência foi descrito no questionário como alto, em justificativa o instrutor alegou que "o principal motivo foi a dificuldade de aprendizado que eles encontraram na oficina e não souberam lidar. O ritmo planejado não era o ritmo ideal para a maior parte dos estudantes". A colaboração entre os estudantes não foi em geral incentivada, e apesar de permitido que algumas atividades fossem realizadas em duplas, a preferência dos discentes era a realização das atividades de maneira individual.

A oficina de Python decorrida no primeiro semestre de 2015 foi desenvolvida e instruída por três estudantes, identificados como Instrutor 4, participante do Programa Institucional de Iniciação Científica, Instrutor 5, participante do programa Compute Você Mesm@ e Instrutor 6, integrante do PET- CoCe. Os envolvidos optaram por oferecer a oficina através de uma abordagem que além, de fornecer o conhecimento técnico computacional, proporcionasse aos estudantes oportunidades do exercício do pensamento crítico.

De acordo com o questionário sobre a oficina respondido pelos instrutores da mesma, para a oficina do ano de 2015 planejou-se uma abordagem já diferente das oficinas de Python anteriormente ministradas. Foram utilizadas bases do construtivismo social da tecnologia que visaram abordar a computação e suas ferramentas de formas distintas, pretendendo apresentar os conteúdos de maneira que englobassem questões do cotidiano, para que os estudantes pudessem assimilar as representações do mundo real nos exercícios e atividades computacionais. Conforme o Instrutor 5, além de se buscar uma abordagem construtivista, a oficina de 2015 também visou ser orientada à projetos, "apresentando o computador sob

um olhar da ação situada e dos estudos em cultura material".

O número de participantes desta edição da oficina de programação em Python é de cinco estudantes, quatro dos quais têm idades variando entre 15 e 16 anos e um apresenta idade de 11 anos. Apesar de todos já terem tido contato com linguagens de programação antes, seus níveis de conhecimento quanto maneira de programar são variados. Esta oficina também conta com a presença eventual de dois outros estudantes de magistério, entretanto pela presença se dar de maneira esporádica, os mesmos não são considerados como alvo da pesquisa nesse trabalho.

Conforme os Instrutores 5 e 6, as atividades foram desempenhadas tanto de maneira individual quanto em grupo, não havendo desistências. Segundo percepção do Instrutor 6, a motivação dos estudantes é avaliada como boa, pois os estudantes de maneira geral tinham vontade de resolver os desafios propostos.

Essa oficina foi dividida em 12 encontros, e em cada encontro foi utilizada a estratégia de ensino por projetos, onde o objetivo é envolver o estudante na atividade e colocá-lo como sujeito ativo, que por intermédio de experiências em seu meio pudessem resolver problemas e o conteúdo é apresentado dentro de um contexto familiar ao aprendiz. A partir dessa oficina, foi utilizado uma aula desses 12 encontros para validar os estudos discutidos e explanados nesse trabalho, conforme será descrito a seguir.

A tabela 6 apresenta a lista de atividades realizadas durante a oficina de programação em Python com base no plano de aula desenvolvido pelos instrutores.

Tabela 6: Atividades desenvolvidas na Oficina de Python de 2015

Nome da Aula	**Descrição/Abordagem**
Mestre Mandou	Aula de introdução ao curso de programação em Python, para que os participantes entendessem o conceito de imperatividade de uma linguagem de computador. Foram utilizadas figuras, em que um participante deveria falar as instruções, baseando-se na figura recebida, para que o outro participante pudesse reproduzi-la.

Continua na próxima página

Tabela 6 – *Continuada da página anterior*

Nome da Aula	**Descrição/Abordagem**
Mapa de distâncias	Aula para entender e reconhecer as operações matemáticas básicas, e entender como uma variável armazena valores. Nessa aula o material utilizado foi um mapa da cidade de Curitiba e uma régua para cada estudante, e os exercícios foram propostos em forma de desafio, os estudantes deveriam medir com uma régua no mapa a distância entre terminais de ônibus da cidade. Depois, deveriam implementar individualmente esta atividade gerando um programa computacional que medisse as distâncias entre os terminais de ônibus da cidade.
Que dia você nasceu?	Indagou-se aos estudantes se esses sabiam qual dia da semana eles tinham nascido. A partir disso foi apresentada uma fórmula que realiza esse cálculo, e a atividade realizada então propôs que eles transcrevessem a fórmula de maneira computacional individualmente, para encontrar o resultado, utilizando o conceito de variável.
Jokenpô	A aula teve o objetivo de fazer com que os estudantes entendessem o funcionamento do par de instruções *"if/else"* e operadores de comparação (==,¿,¡,!=). Para isso, foi utilizado o jogo Jokenpô, em que os participantes tiveram que montar as regras do jogo da maneira que preferissem, com o objetivo de demonstrar o funcionamento das condicionais para as regras que eles desenvolveram. Foram solicitados participantes para jogar Jokenpô enquanto um outro estudante, sem observar o jogo criou regras e teve de definir o vencedor com perguntas aos jogadores cuja a resposta seria somente "sim" ou "não". Como motivação, foi apresentado um vídeo onde uma mão mecânica jogava Jokenpô contra humanos e era programada para sempre ganhar. Após isso, cada estudante teve, com o auxilio do computador, transcrever o jogo em linguagem computacional.

Continua na próxima página

Tabela 6 – *Continuada da página anterior*

Nome da Aula	Descrição/Abordagem
Super Trunfo	Nessa atividade foram entregues duas cartas do jogo super trunfo do Adventure Time a cada estudante, o objetivo foi que eles jogassem em duplas, com os dois primeiros itens da carta, e definissem, dependendo do valor do item, o vencedor da partida. Eles foram instigados a pensar como um computador escolheria uma carta, para que a carta escolhida fosse a vencedora. Após jogarem, foi solicitado que desenvolvessem individualmente um programa computacional para resolução do problema proposto, utilizando conceitos de comparadores, *"if/else"*.
Advinha que número estou pensando	Para apresentar o conceito de laço de repetição aos estudantes, foi solicitado que eles desenvolvessem computacionalmente um programa que transcrevesse a brincadeira "advinha qual número estou pensando". Foi explicado como funciona o laço de repetição utilizando *while* através de um exemplo, e eles transcreveram e observaram como funcionamento do código antes de desenvolver o exercício proposto individualmente.
Código Morse	Foi entregue no material de apoio duas tabelas, uma com a tradução do código morse para as letras do alfabeto e outra numerais de 0 a 9, então foi solicitada uma sequência de exercícios onde o estudante teria que individualmente implementar computacionalmente um tradutor para o código morse, onde as entradas seriam frases e o resultado seria o código morse, e posteriormente o inverso afim de traduzir algumas frases que foram passadas/criptografadas.
Jokenpô (Spock - Lagarto)	Nesse encontro o objetivo foi revisar os conteúdos já vistos pelos estudantes até o momento. Foi utilizado o código feito pelos estudantes na aula "Jokenpô", e a partir deste eles teriam de implementar mudanças para adicionar dois novos itens "Spock" e "Lagarto" como novas regras. Desta forma, o jogo permitiu a revisão dos conteúdos que foram apresentados após a aula "Jokenpô". Para explicar o funcionamento do jogo, foi apresentado um vídeo da série *The Big Bang Theory*, em que os personagens da série jogam essa versão adaptada do mesmo.

Continua na próxima página

Tabela 6 – *Continuada da página anterior*

Nome da Aula	**Descrição/Abordagem**
Jogo da Velha	Essa atividade foi realizada em duas aulas, os estudantes jogaram no quadro o jogo da velha entre si, estimulados a perceber como o computador tomaria a decisão e receberia o comando por eles indicados em um programa computacional, influenciados a levar um problema real para o mundo computacional. Depois foi solicitado aos estudantes que implementassem em duplas, utilizando linguagem computacional o jogo da velha.

4.5 Planejamento da Oficina

Conforme apresentado dentro da visão sociocultural e construcionista de ensino, o ideal é a oficina se desenvolver sobre temas que tenham significância para os estudantes, de forma que este possa debater e compartilhar com seus colegas o tema escolhido para aprimorar a construção do conhecimento.

Portanto, dentro do público adolescente que se apresenta na oficina de Python desenvolvida na UTFPR, para incentivar o debate e dar a liberdade aos estudantes de escolher o tema que mais tenha significado para os próprios, que preferencialmente seja trazido de seus cotidianos, optou-se por trabalhar a narrativa de histórias. Ainda que trabalhando estruturas simples de programação, pois os estudantes ainda apresentam conhecimento básico, é possível trazer a reflexão de certos temas. Por exemplo, através de uma estrutura básica de *"if/else"* é possível fomentar a discussão e construção de implicações existentes nas escolhas que uma pessoa, um adolescente, ou um estudante faz.

Apesar da importância do debate e fomento de temas que sejam culturalmente ou politicamente relevantes para os estudantes, (Blinkstein, 2004) alerta para o risco previsto por Freire de ocorrer a decisão prévia de temas geradores sem de fato estes emergirem dos aprendizes. No caso, a falta de autenticidade dos temas poderia produzir efeitos "domesticadores", em que doutrinas são criadas pela falta de criticismo, o que acabaria por gerar um efeito manipulador nos estudantes. Portanto, as práticas de *Brainstorming*[9] e *BrainDraw*[10] foram escolhidas como suporte por permitir a livre geração e desenvolvimento de ideias ou temas que para estes tenham significância, sem que sejam predefinidos pelos instrutores da oficina.

Na ferramenta Pygame é possível trabalhar estruturas simples, como as de decisão, uti-

[9] *Brainstorming* é um método individual ou em grupo para gerar ideias, aumentar a eficácia criativa, ou encontrar soluções para os problemas, em que os participantes geram idéias sobre um determinado tema ou problema em um ambiente sem julgamento (Wilson, 2014).

[10] *BrainDraw* é um *brainstorming* em forma de rodízio, em que cada participante inicia com um desenho ou ideia inicial, e os outros continuam (Wilson, 2014).

lizando também componentes gráficos, em que imagens, futuramente, podem vir a dar lugar a animações (através de laços de repetição). Portanto, caso o estudante se sinta confortável quanto ao desenvolvimento do conteúdo, é possível avançar para a construção de uma animação apenas acrescentando algumas linhas de código. É interessante também a liberdade que o estudante pode ter de pesquisar suas próprias imagens online e utilizá-las, de maneira fácil, da forma que achar melhor ao montar a sua história ou descrever fatos por ele escolhidos. Na aula anterior à da oficina, buscou-se verificar com os cinco estudantes presentes quais destes poderiam comparecer, ocasião na qual todos confirmaram presença, com exceção do estudante de 11 anos. Também verificou-se a possibilidade de estender em uma hora a duração da oficina (que normalmente dura uma hora e vinte minutos), e todos concordaram. Ainda assim um email foi enviado durante a semana lembrando-os do tempo a mais na próxima aula da oficina e solicitando que avisassem seus responsáveis.

O planejamento da oficina se deu no estabelecimento de materiais e métodos a serem utilizados com os quatro estudantes, levando em consideração os conceitos já aprendidos por estes anteriormente na oficina.

4.5.1 Conceitos Abordados

A aula de "Contando Histórias com Animações" se caracterizou por ser a décima primeira aula de um conjunto de doze aulas propostas pela oficina de 2015.

Os conceitos abordados na aula proposta buscaram revisar e dar continuidade aos conceitos que já haviam sido apresentados nas dez aulas anteriores, como: entrada e saída de dado, condiconal (*if/else*) e laço de repetição (*while*). De conteúdo novo, foi brevemente introduzido o conceito de manipulação de imagens e eventos (para clique de botão).

4.5.2 Material

O material necessário para a aula da oficina foi definido de acordo com o método e atividades selecionadas da mesma, sendo estes:

- Laboratório com 4 máquinas iMAC;
- Exemplos de apoio no Pygame, suportado pela ferramenta Pycharm;
- Uma folha sulfite A3 e quatro folhas sulfite A4; e
- Canetas de colorir e lápis de cor.

Exemplo de Apoio: Como código de base/apoio para a aula da oficina, optou-se por desenvolver dois exemplos para os estudantes explorarem, e também por utilizar personagens dos filmes da saga Harry Potter, retiradas da internet, uma vez que esta série é popular

entre o público-alvo da oficina. O Exemplo 1 (Figura 41) buscou trabalhar estruturas de decisão (*if/else*) e entrada de dados. No cneário do exemplo, verificou-se as implicações de ter estudado para uma prova ou não.

A Figura 42 descreve o momento em que, após exibida a imagem de um estudante "pensativo", a seguinte pergunta é feita: "Você estudou para a prova?". Se a resposta é "Sim", as figuras se alteram, a imagem do personagem pensativo á substituída pela imagem de um personagem feliz, além da atividade indicando "férias" (Figura 43). Caso contrário, as figuras indicam um personagem triste e a atividade de estudos por conta da "recuperação" (Figura 44).

```
import sys, pygame
from pygame import *

pygame.init()

corBranca = 255,255,255
tamanhoJanela = width, height = 1200, 1200

tela = pygame.display.set_mode(tamanhoJanela)

#Imagens Utilizadas
pensando = pygame.image.load("pensando.jpg")
feliz = pygame.image.load("feliz.jpg")
triste = pygame.image.load("triste.jpg")

tela.fill(corBranca)
tela.blit(pensando,[0,0]) #posiciona a imagem na posicao X e Y
pygame.display.flip() #mostra a imagem

bemNaProva = input('Voce estudou para prova?')
rodando = True

while(rodando == True):

    if (bemNaProva == 'Sim'):
        print ('Ferias! :D ')
        tela.fill(corBranca)
        tela.blit(feliz,[0,0])
        pygame.display.flip()
    else:
        print ('Recuperacao! :( ')
        tela.fill(corBranca)
        tela.blit(triste,[0,0])
        pygame.display.flip()

    #Encerra o Programa
    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                rodando = False
```

Figura 41: Código do Exemplo 1 utilizado de base na aula da oficina. Fonte: Autoria Própria.

Figura 42: Imagem de personagens “pensativos” sobre a questão proposta. Fonte: Autoria Própria.

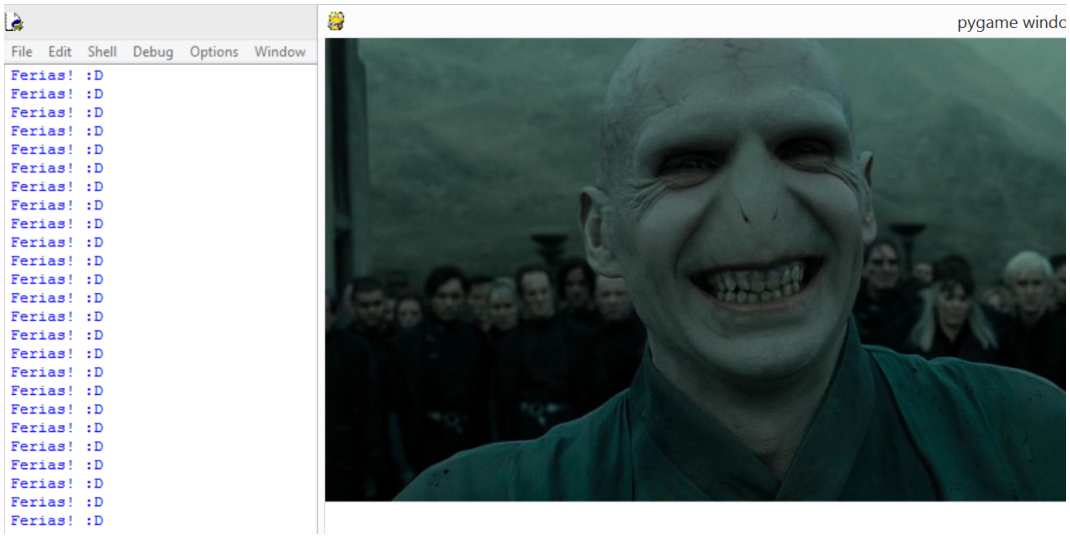


Figura 43: Imagem de personagem “feliz” como resultado de ter ido bem na prova. Fonte: Autoria Própria.

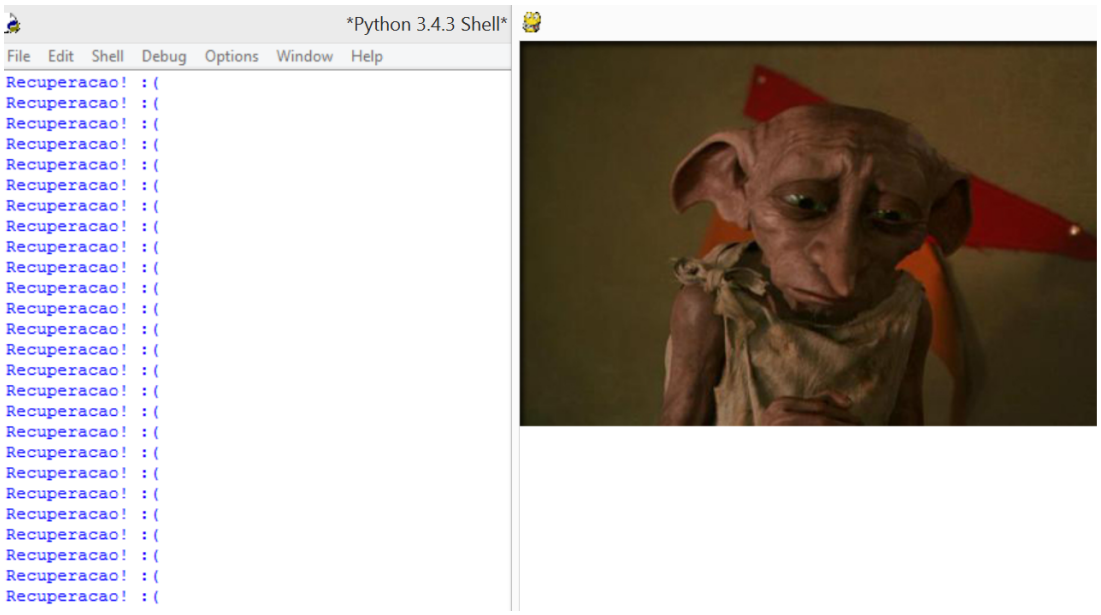


Figura 44: Imagem de personagem “triste” como resultado de não ter ido bem na prova. Fonte: Autoria Própria

O segundo exemplo de código (Figura 45) se propunha a trabalhar o laço de repetição, com o objetivo de demonstrar um animação. O código, quando executado, faz com que, uma vez que realizado o clique de mouse, a imagem se desloque para o sentido oposto (Figura 46).

```
import pygame
from pygame import *

pygame.init()

corVermelha = 255,0,0
tela = pygame.display.set_mode((1200, 800))

tela.fill(corVermelha)

imagem = pygame.image.load("imagem.jpg")
rodando = True

i = 0
f = 10

while (rodando == True):

    i = i + f

    tela.fill(corVermelha)

    if i > 500:
        i = 500
    if i < 0:
        i = 0

    tela.blit(imagem,[i,0]) #posiciona a imagem
    pygame.display.flip() #mostra a imagem

    # Se algum evento acontecer
    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                rodando = False
            elif event.type == MOUSEBUTTONDOWN:
                f = f * -1
```

Figura 45: Imagem do personagem move-se em sentido inverso cada vez que efetuado o clique de mouse. Fonte: Autoria Própria.

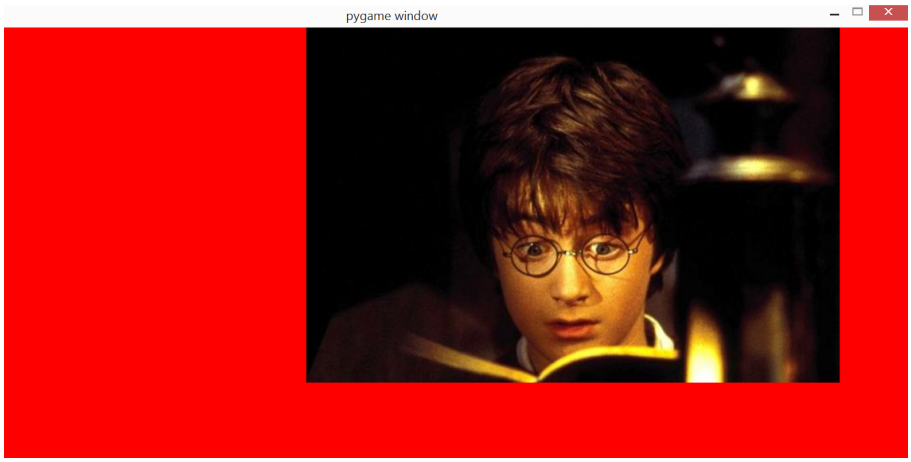


Figura 46: Imagem do personagem move-se em sentido inverso cada vez que efetuado o clique de mouse. Fonte: Autoria Própria

4.5.3 Método

A metodologia planejada para a oficina foi a definida nos passos de 1 a 8, pensada de acordo com o tempo disponível da aula, de duas horas e vinte minutos.

1. Recepção dos estudantes. Introduzir a atividade da oficina, que seria a criação de uma

história montada utilizando animação em Python. Tempo estimado de 5 minutos;

2. Prática de Brainstorming. Levantar tópicos significativos para os estudantes, para servir como base para a história. Colocar um papel A3 no meio da mesa e estudantes ao redor da mesma, e deixar que estes definiram/escrevam o que quiserem. Sugerir a pergunta "o quê vocês gostam?" como tema do brainstorming. Tempo estimado de 15 minutos;

3. Prática de BrainDraw. Elaborar em conjunto o esqueleto de uma história, que pode ser desenhada ou escrita. Cada estudante recebe uma folha A4 e inicia uma história. Cada rodada tem um tempo de quatro minutos, e depois desta o estudante deve passar o seu papel para o colega ao lado em forma circular, para que este continue sua história. Apresentar ao final quatro esqueletos de quatro histórias diferentes. Tempo estimado de 16 minutos;

4.Discutir e decidir qual história utilizar e o roteiro desta, e se querem usar a história desenvolvida ou preferem que cada um adapte ou crie uma história diferente. Tempo estimado de 10 minutos;

5.Explorar os exemplos preparados no Pygame. Não explicar o código linha a linha, mas sim deixar que rodem e verifiquem os resultados do código por si só, auxiliando em eventuais dúvidas. Explorar individualmente, ou seja, cada um em um computador. Incentivar que façam alterações no código, como por exemplo alterar uma imagem (buscada na Internet) ou alterar a cor da tela. Tempo estimado de 40 minutos (20 minutos para cada exemplo);

6. Pausar para descanso. Tempo estimado de 10 minutos;

7. Desenvolver história no Pygame. Sugerir que apresentem, ao fim, para os outros colegas a história preparada. Tempo estimado de 40 minutos; e

8. Solicitar preenchimento do questionário. Tempo estimado de 5 minutos.

Durante toda a aula da oficina, foram recolhidas observações quanto ao desenvolvimento e resultados obtidos, através de anotações ou *prints* de tela. Observou-se se todos os estudantes conseguiram finalizar a história, e se todos utilizaram os conceitos abordados.

5 Resultados Obtidos

A oficina saiu do horário inicialmente planejado, devido ao atraso de alguns estudantes para chegar ao local, fazendo com que as instrutoras tivessem que adaptar a duração das atividades no decorrer da oficina.

Após a apresentação de uma das integrantes da oficina e uma breve explicação do objetivo da aula, iniciou-se um exercício de *Brainstorming*, sugerindo que os estudantes pensassem a respeito do que gostam, uma pergunta genérica para que gerassem ideias livres de compromisso em um papel. Cada estudante ficou com uma folha de papel, diferenciando do planejado que era uma folha ao centro da mesa, para todos compartilharem as ideias, então as instrutoras solicitaram que a cada determinado tempo, os participantes trocassem suas

folhas com o estudante seguinte, configurando o exercicío em uma prática de *BrainDraw*.

Os temas e assuntos levantados pelos estudantes durante esta prática foram variados (Figura 47), envolvendo jogos de videogame (Super Mario, Pac-Man, Pokémon), programas e séries de televisão (Harry Potter, Naruto, Pretty Little Liars, Dragon Ball Z, Star Wars, Batman, Superman), personagens pontuais (Bob Esponja, Gandalf, Surfista Prateado, Flash), atividades (programar, dormir, tocar violão) e demais itens, como dinheiro, carro e comidas (*fast-food*). Além destes, um desenho representando o planeta Terra, expressando um descontentamento pelo Brasil.

Figura 47: Prática de BrainDraw para geração de ideias. Fonte: Autoria Própria.

Uma vez que esta prática foi finalizada, levantou-se uma discussão para que os estudantes percebessem os gostos em comum com seus colegas. As instrutoras fizeram questionamentos a cerca dos desenhos, incluindo o porquê da insatisfação com o Brasil. As respostas obtidas foram "o Brasil está todo ruim", "só aparece notícia ruim na televisão", "é muita corrupção", "é muito imposto", e "culpa da Dilma".

As instrutoras seguiram com as atividades previstas, solicitando aos estudantes que iniciassem uma história para que o colega seguinte continuasse, trocando as folhas entre si, e obtendo ao fim 4 diferentes histórias (Figura 48). Ao final desta atividade, cada estudante explicou a continuação desenvolvida na sua própria história, e foi possível observar que alguns personagens citados na atividade anterior reapareceram nas histórias. Além disso, apenas um dos estudantes preferiu iniciar a história de forma escrita, enquanto os outros o fizeram por meio de desenhos.

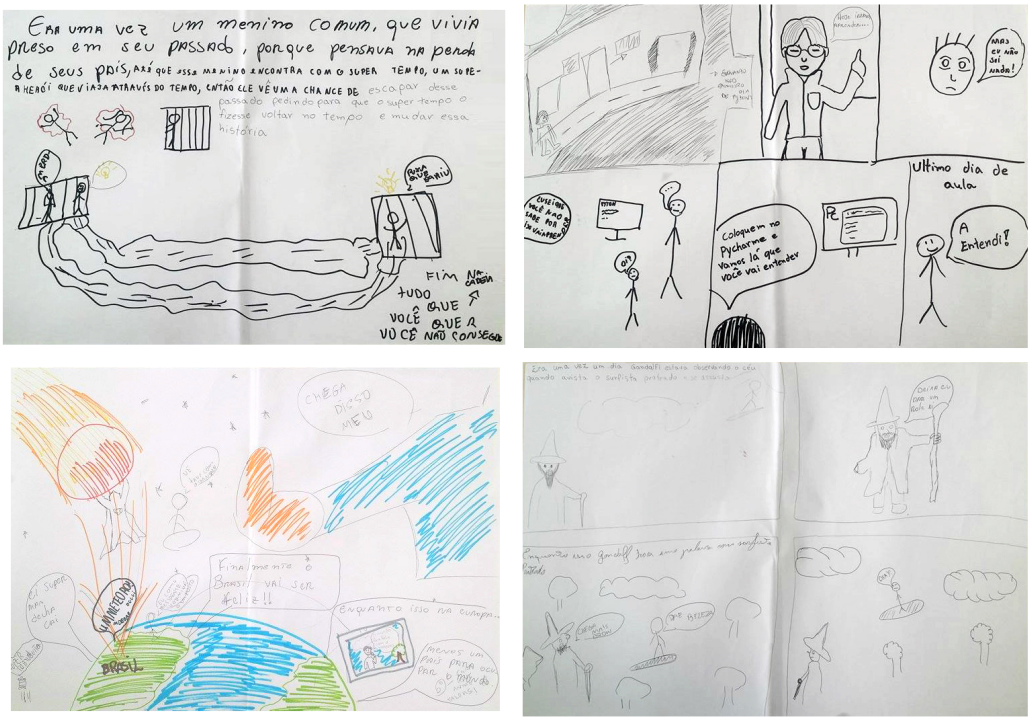


Figura 48: Prática de BrainDraw para geração de quatro histórias. Da esqueda para a direita, de cima para baixo: histórias A, B, C e D. Fonte: Autoria Própria

Além dos personagens levantados na prática anterior, surgiram também nas histórias personagens novos baseados nas experiências e vivências reais do cotidiano dos estudantes, como a representação de um instrutor da oficina, figurado pelo personagem "Super Kira" no ambiente de sala de aula da oficina (história B da Figura 48). Também foram citados outros elementos da vida dos estudantes, como o local em que os mesmos realizam a oficina (Sala do PET).

O Brasil voltou a aparecer como tema de uma história, na qual um meteoro é lançado em direção ao Brasil, com o intuito de destruir o país (história C da Figura 48). Apesar do personagem do Superman tentar impedir o meteoro, manifestações contrárias surgem, pedindo para o herói deixá-lo cair e celebrações como "Finalmente o Brasil vai ser feliz!". Uma representação da presidente também foi colocada na história em "Eu como presidente vou diminuir o imposto", além de representações de ideias quanto à repercussão da mídia internacional: "O mundo fica feliz com o meteoro" e "Menos um país para ocupar o mundo". Após a apresentação desta história, as instrutoras questionariam sobre a possível solução para a situação do Brasil. As respostas inicias foram "o problema são os políticos" e "os políticos são muito corruptos", até que outro estudante interveio alegando que em primeiro lugar "o povo tem que aprender a votar".

Uma das histórias se deu em torno de um menino que vivia preso em seu passado por pensar na perda dos pais (história A da Figura 48), e esta foi continuada de maneira que este pudesse retornar ao passado para alterá-la (entretanto sem sucesso ao fim). Por fim, outra história narrava a interação entre os personagens de Gandalf e Surfista Prateado (história D da Figura 48).

Depois desta prática, perguntou-se aos estudantes de que a melhor forma eles gostariam de trabalhar ao se pensar na história a ser desenvolvida no computador: Cada um desenvolver a sua própria história ou decidirem em conjunto sobre uma história em comum para que cada um desenvolvesse um pedaço da mesma no computador. Além disso, estes poderiam tanto utilizar uma das histórias que estavam prontas a partir da prática anterior, ou elaborar um nova, baseando-se ou não nas ideias das práticas anteriores. Estes optaram por unir elementos de várias histórias para montar uma nova história, e que todos iriam desenvolver no computador um pedaço diferente desta mesma história.

Assim, durante a elaboração da história os estudantes decidiram utilizar o personagem do "Super Kira", incluindo também um novo personagem baseado em um professor o qual eventualmente está presente nas oficinas de Python, o "Mister André", que seria o vilão tramando contra o "Super Kira". Desta forma, os estudantes definiram a história nos seguintes passos:

1. "Tudo começa quando Gandalf conta uma história sobre um meteoro".
2. "Gandalf estava numa partida de futebol com o Surfista Prateado".

3. “O Surfista acaba dando um chute muito forte e o meteoro vai em direção a Terra, mais precisamente no Brasil”.
4. “O Mister André manipula o meteoro para que ele vá onde o Super Kira está”.
5. “O Super Kira monta uma parede de códigos e manda um break para o meteoro, e com isso o meteoro vai em direção ao Mister André”.

Os estudantes ainda escreveram, ao fim dos passos determinados, a frase “Continua no próximo episódio..”, o que indicava que o último passo ainda não representava o final da história que estes estavam propondo.

5.1 Códigos Desenvolvidos no Pygame

Cada estudante escolheu uma parte da história, que seria desenvolvida de maneira individual no computador. Por perceberem que haviam definido cinco partes da história para apenas quatro participantes, estes negociaram e o Estudante 2 por fim concordou em desenvolver duas partes da história. Os Estudantes 1, 3, e 4 optaram por desenvolver a sua história a partir da modificação do exemplo passado, e o Estudante 2 optou por iniciar de um arquivo em branco. Todos os estudantes optaram por buscar imagens novas na internet para compor suas histórias.

A primeira parte da história foi desenvolvida pelo Estudante 3, a partir da proposta: “Tudo começa quando Gandalf conta uma história sobre um meteoro”. A principal dúvida deste estudante foi quanto à transformação da sua frase no formato de uma história, demorando principalmente para determinar qual seria a pergunta que este faria para o usuário. O estudante 3 também deu nomes significativos às variáveis do seu programa, além de buscar estruturar o código de maneira organizada (Figura 49). As Figuras 50 e 51 demonstram a execução do código produzido por este.

```
import sys, pygame
pygame.init()

corBranca = 0,0,0
tamanhoJanela = width, height = 800, 800

tela = pygame.display.set_mode(tamanhoJanela)

#Imagens Utilizadas
contador = pygame.image.load("Gandolf.jpeg")
meteoro = pygame.image.load("meteoro.jpeg")
tchau = pygame.image.load("triste.jpg")

tela.fill(corBranca)
tela.blit(contador,[0,0]) #posiciona o estudante posicao X e Y
pygame.display.flip() #mostra a imagem

historia = input('gostaria de saber sobre um meteoro que foi parado por super-Kira?')
rodando=True
while (rodando==True):
    if (historia == "sim" ):
        print('tudo comecou com o futebol...')
        tela.fill(corBranca) #pinta a tela
        tela.blit(meteoro,[0,0])
        pygame.display.flip()
    else:
        print('tudo bem entao')
        tela.fill(corBranca)
        tela.blit(tchau,[0,0])
        pygame.display.flip()

    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                rodando=False
```

Figura 49: Código fonte do programa do Estudante 3. Fonte: Estudante 3.

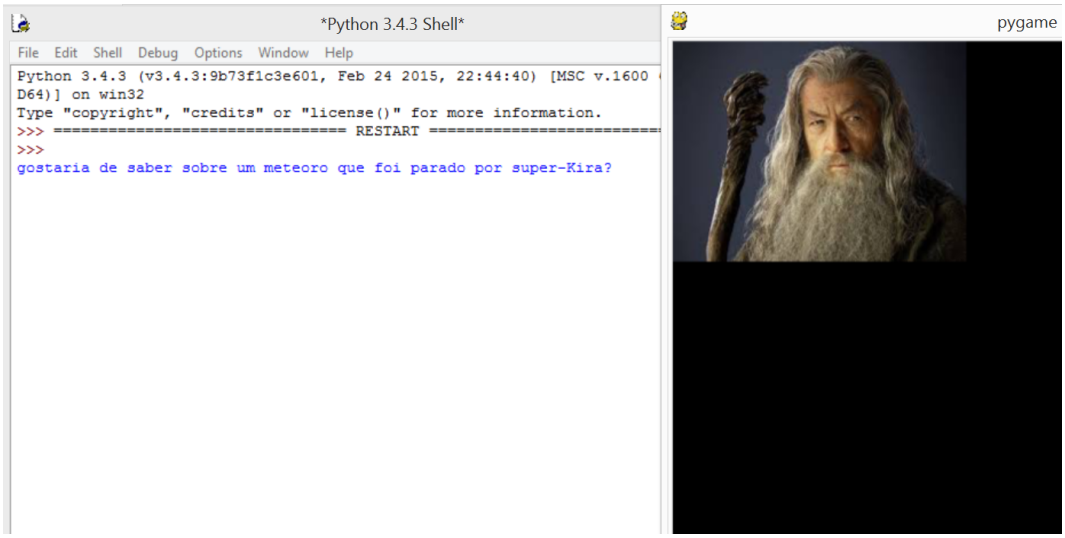


Figura 50: Parte 1 da história produzida pelo Estudante 3. Fonte: Estudante 3.

A segunda parte da história definida como "Gandalf estava numa partida de futebol com o Surfista Prateado" ficou sob responsabilidade do Estudante 4 para desenvolvimento. Este é o estudante que apresenta maior conhecimento prévio em programação. Ao explorar os exemplos passados, o estudante em poucos minutos assimilou o funcionamento destes, alterando o Exemplo 2 para fazer com que a imagem caminhasse na diagonal. Por ter buscado

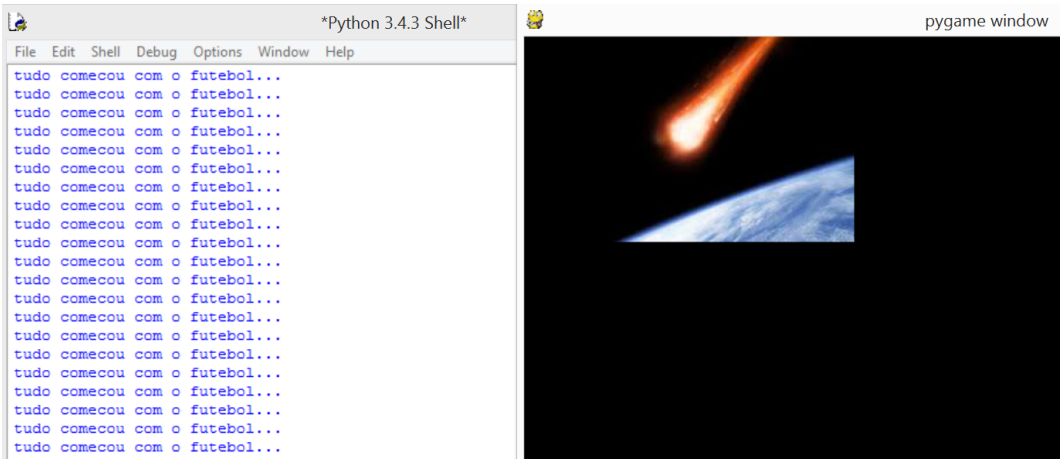


Figura 51: Parte 1 da história produzida pelo Estudante 3. Fonte: Estudante 3.

a elaboração de estruturas mais complexas no seu desenvolvimento (Figura 52), este foi o único estudante que não finalizou um trecho da história proposta no tempo da aula.

```
import sys, pygame
pygame.init()

corBranca = 0,255,255
tamanhoJanela = width, height = 1000, 800

tela = pygame.display.set_mode(tamanhoJanela)

#Imagens Utilizadas
pensando = pygame.image.load("1.jpg")
feliz = pygame.image.load("3.jpg")
triste = pygame.image.load("2.jpg")

#tela.fill(corBranca)
#tela.blit(pensando,[0,0]) #posiciona o estudante posicao X e Y
#pygame.display.flip() #mostra a imagem

bemNaProva = raw_input('Gandalf estava na partida??')
rodando=True
x=0
y=100
a=100
b=0
while (rodando==True):
    if (bemNaProva == 'sim'):
        x=x+1
        print('')
        tela.fill(corBranca) #pinta a tela
        tela.blit(feliz,[x,100])
        pygame.display.flip()
        if x>100:
            x=100
            a=a+5
            tela.blit(triste,[a,100])
            pygame.display.flip()
            if a>300:
                a=300
                if a=300:

            if a<0:
                a=0
        if x<0:
            x=0

   # else:
    #    print('Recuperacao :D')
     #   tela.fill(corBranca)
      #  tela.blit(triste,[0,0])
       # pygame.display.flip()
#Encerra o programa
    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                rodando=False
```

Figura 52: Código-fonte do programa do Estudante 4. Fonte: Estudante 4.

A terceira parte da história definida como “O Surfista acaba dando um chute muito forte e o meteoro vai em direção a Terra, mais precisamente no Brasil” foi desenvolvida pelo Estudante 2. Este também ficou responsável pela parte 4 da história, entretanto não teve tempo de finalizar a mesma. O Estudante 2 apresentou bastante interesse no funcionamento do processo de animação da imagem, e seu resultado foi, depois de encontrar uma imagem que representasse um “Surfista Preateado”, simulou o movimento similar ao que o super herói “Flash” faz, movendo a imagem da esquerda para a direita (Figura 49).

Por não ter o costume de dar nomes significativos para as variáveis e imagens que escolheu da internet, o estudante apresentou dificuldades em fazer com que o programa funcionasse. Mesmo depois de perceber os erros, este não buscou dar nomes mais significativos para suas variáveis e imagens, e percebe-se em seu código (Figura 53) que este utiliza, por exemplo, nome de cores, letras aleatórias e o próprio nome (representado pelo retângulo cinza) para imagens de personagens e outros que mais tarde precisariam ser referenciados em seu código. Os instrutores da oficina optaram por utilizar uma abordagem em que apenas orientou-se os estudantes quanto às boas práticas de programação, ficando livre a escolha destes em segui-las ou não.

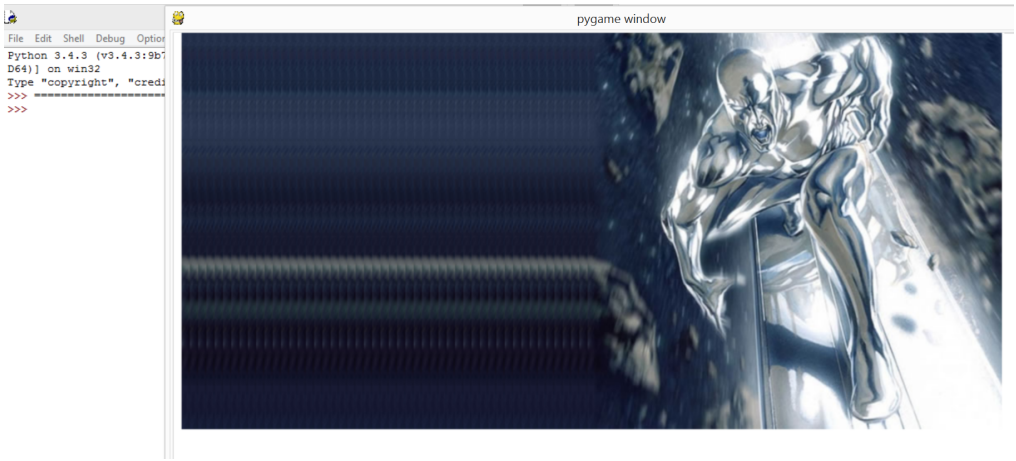


Figura 53: Parte 3 da história produzida pelo Estudante 2. Fonte: Estudante 2.

```
__author__ = 'dainf'
import sys, pygame
from pygame import *

pygame.init()
amarelo = 255,255,255
tamanhoJanela = width, height = 1000, 800
tela = pygame.display.set_mode(tamanhoJanela)
surfista = pygame.image.load("ffff.jpg")
vermelho = pygame.image.load("oi.jpg")
preto = pygame.image.load("ccccc.jpg")
laranja = pygame.image.load("     .jpeg")
tela.fill(amarelo)
tela.blit(surfista,[0,0]) #posiciona a imagem na posicao X e Y
pygame.display.flip() #mostra a imagem


import pygame
from pygame import *

pygame.init()

corVermelha = 255,255,255
tela = pygame.display.set_mode((1200, 1000))

tela.fill(corVermelha)

imagem = pygame.image.load("ffff.jpg")
rodando = True
xdosurfista = 0
velocidade = 10


while (rodando == True):
   xdosurfista = xdosurfista + velocidade
   tela.blit(imagem,[xdosurfista,0]) #posiciona a imagem
   pygame.display.flip() #mostra a imagem
```

Figura 54: Código-fonte do programa do Estudante 2. Fonte: Estudante 2.

O Estudante 1 foi o responsável pela última parte da história: "O Super Kira monta uma parede de códigos e manda um break para o meteoro, e com isso o meteoro vai em direção ao Mister André". Este estudante apresenta nível básico em conhecimentos de programação, e apresentou dificuldades ao estruturar a história na forma de estruturas de condição. Depois de sanadas as dúvidas, conseguiu estruturar seu código de maneira a utilizar mais figuras do que o exemplo de base apresentava.

De maneira geral, este estudante conseguiu elaborar um código organizado (Figura 55), apresentando resultados definidos pelas Figuras 56, 57 e 58.

```
import sys, pygame
from pygame import *

pygame.init()

corPreta = 0,0,0
tamanhoJanela = width, height = 1500, 1000

tela = pygame.display.set_mode(tamanhoJanela)

#Imagens Utilizadas
parede= pygame.image.load("paredeDeCodigo.jpg")
breeak= pygame.image.load("break.jpg")
kira = pygame.image.load("superKira.jpg")
fim =pygame.image.load("fim.jpg")
andreAssustado = pygame.image.load("andreAssustado.jpeg")
tela.fill(corPreta)
tela.blit(kira,[0,0]) #posiciona a imagem na posicao X e Y
pygame.display.flip() #mostra a imagem

kiraSalveMundo = input('Kira conseguiu mandar a parede de codigo?')
rodando = True

while(rodando == True):

   if ( kiraSalveMundo== 'sim'):
       print ('Parede de codigo!!')
       tela.fill(corPreta)
       tela.blit(parede,[0,0])
       tela.blit(breeak,[100,0])
       tela.blit(andreAssustado,[1000,0])
       pygame.display.flip()
   else:
       print (' :( ')
       tela.fill(corPreta)
       tela.blit(fim,[0,0])
       pygame.display.flip()

   #Encerra o Programa
   for event in pygame.event.get():
           if event.type == pygame.QUIT:
               rodando = False
```

Figura 55: Código-fonte do programa do Estudante 1. Fonte: Estudante 1.

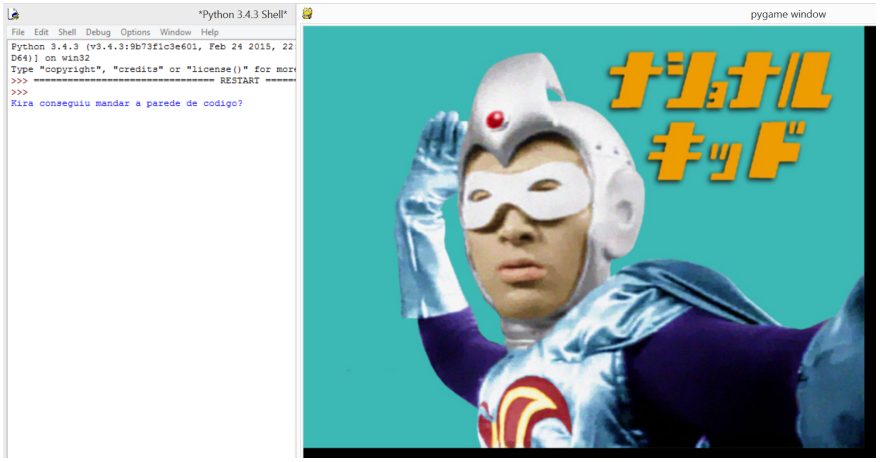


Figura 56: Parte 5 da história produzida pelo Estudante 1. Fonte: Estudante 1.

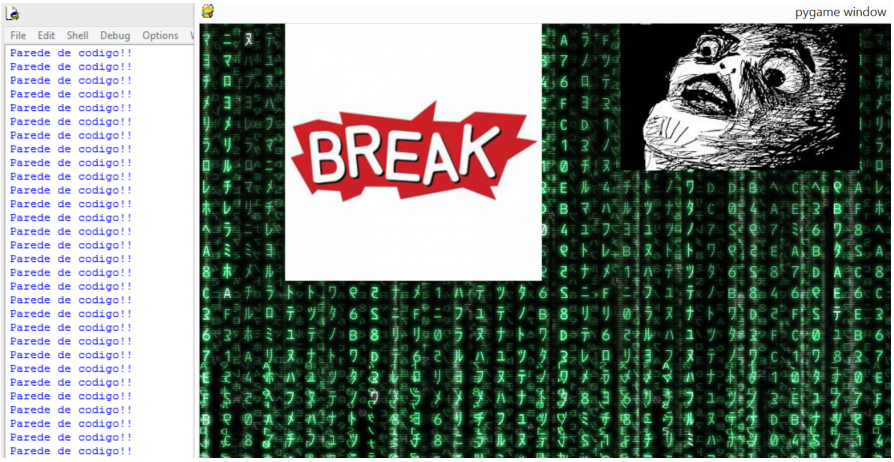


Figura 57: Parte 5 da história produzida pelo Estudante 1. Fonte: Estudante 1.

Figura 58: Parte 5 da história produzida pelo Estudante 1. Fonte: Estudante 1.

Por conta do interesse dos estudantes, ao fim desta aula optou-se por utilizar também a aula seguinte para dar continuidade aos trabalhos apresentados nesta, incluindo o objetivo de ao fim os estudantes apresentarem aos colegas o que produziram.

5.2 Questionário

Para complementar o estudo foi solicitado aos estudantes que, após o fim da oficina, respondessem a um questionário a respeito da participação dos mesmos na oficina de programação em Python.

Os questionamentos efetuados a respeito do interesse dos estudantes em linguagens de programação antes da oficina de programação em Python foram em suma positivos, dois dos quatro estudantes afirmaram ter contato com programação antes da realização da oficina de programação em Python em uma oficina de Arduíno, um estudante afirmou ter tido contato com programação em um curso de linguagem C na UTFPR, e outro estudante afirmou ter tido contato com programação anteriormente ao curso, no colégio.

Quando solicitados a avaliar o interesse que possuíam na área de Computação após a realização da oficina, um estudante avaliou como inalterado esse interesse, os demais estudantes afirmaram que houve um aumento no interesse que possuíam por essa área. Ao serem indagados sobre a preferência entre atividades com os conteúdos predefinidos e atividades que permitissem que eles decidissem o quê trabalhar durante as aulas, a maioria afirmou preferir a liberdade na escolha, justificando que poderiam desse modo expressar a criatividade de melhor maneira e mostrar as ideias em novos conteúdos, um dos estudantes afirmou preferir o oposto, pois para esse, as atividades definidas são melhores que as realizadas na hora.

Um dos estudantes afirmou preferir trabalhar individualmente em suas atividades, contrapondo aos demais colegas que afirmaram preferir atividades em dupla ou grupos pela maior facilidade de desenvolver o pensamento e por acharem mais divertido.

A aula da oficina realizada para esse trabalho foi avaliada pelos estudantes positivamente, foi citada como uma aula interativa e divertida, onde os estudantes destacaram a comunicação que houve entre eles como diferencial, um dos estudantes avaliou como positivo o fato da aula não ser somente presa à área de programação.

5.3 Avaliação da Ferramenta Pygame

Após a realização da oficina pode-se reaplicar o método de avaliação TUP sobre a ferramenta Pygame com o objetivo de comparar os resultados obtidos na primeira avaliação.

Conforme apresentado no Apêndice B, algumas questões apresentam discrepância em relação as respostas obtidas, pois após os testes realizados durante a oficina aqui descrita algumas informações novas se revelaram a respeito da ferramenta, como por exemplo, a

complexidade apresentada no momento da instalação dessa ferramenta em um sistema operacional de um computador MAC, e o fato da ferramenta apresentar travamentos durante a execução nesse ambiente.

Também foi possível verificar que a ferramenta apresenta imperfeições quanto a execução de programas nele desenvolvido, como por exemplo não parar aberta janelas durante a execução. Esses programas tiveram que ser editados, pára que através de adaptações no código desenvolvido o problema fosse corrigido, sendo que em outro sistema operacional esse mesmo problema não ocorreu e isso se fez desnecessário.

5.4 Discussão dos Resultados

A partir dos resultados obtidos, foi possível verificar que as oficinas decorridas nos anos de 2012 e 2013 assemelharam-se à uma metodologia tradicional behaviorista de ensino, em que a partir dos dados obtidos pode-se observar o baixo ou nulo incentivo à interação entre os estudantes. Como citado pelo Instrutor 2, as habilidades sociais não eram incentivadas nestas edições das oficinas, pois o foco se encontrava no ensino de habilidades técnicas. Nestas oficinas pôde-se verificar um grande número de desistentes, cuja justificativa para desistência apresentada pelo instrutor foi a desmotivação, e dificuldade dos estudantes em encararem os desafios propostos, por estes estarem desacostumados ao ritmo da oficina.

Portanto, observa-se que quando a metodologia de ensino aproxima-se de um modelo tradicional de ensino, no qual o estudante é apenas receptor do conteúdo, o discente pode não se sentir confortável no processo de aprendizagem, diferente de quando há um planejamento baseado em uma metodologia que leve a programação a um ambiente familiar ao estudante, fazendo-o se sentir confortável, para viabilizar o ato de "tornar-se sujeito ativo de sua própria aprendizagem".

Desta maneira, contrapondo-se a essas oficinas, a oficina desenvolvida em 2015 juntamente com a aula utilizada como validadora desse estudo tiveram inclusas práticas da filosofia construtivista, e foram estruturadas utilizando-se as vivências dos estudantes como auxílio no processo de aprendizagem. Como resultado, apontou-se nos questionários preenchidos pelos estudantes que a motivação da maioria destes quanto à área de computação aumentou após a oficina de programação em Python. Apesar do número menor de estudantes, esta oficina não apresentou nenhuma desistência.

Para a validação deste estudo, durante a atividade desenvolvida na aula da oficina proposta pelas pesquisadoras, foi encorajado, por meio de práticas participativas, que os próprios estudantes elencassem e debatessem temas que estes considerassem significativos, de forma a apresentar seus gostos e também interagir ao verificar gostos em comum com seus colegas. Além disso, durante a aula, buscou-se dar liberdade para que estes definissem e escolhessem a melhor maneira para construção e desenvolvimento da história, notando-se, por meio das

respostas dos questionários, uma preferência por esta abordagem mais livre, uma vez que, de acordo com os estudantes, favorece a prática de exercícios criativos e é uma maneira de exercitar suas próprias ideias em conteúdos novos.

Portanto, esta abordagem vai de acordo com a visão proposta pela linha sociocultural de ensino, em que os estudantes adquirem conhecimentos que estão adequados à realidade vivida socialmente, tornando-se politicamente conscientes uma vez que interagem com o aprendizado através de suas experiências do cotidiano (Santana, 2010). Foi possível verificar que a abertura desta abordagem fez de fato com que o descontentamento da situação política, econômica e social do país fosse trazida à tona pelos estudantes, possibilitando também a troca e aprofundamento de reflexões, em que, por exemplo, partiu-se de afirmações iniciais como "o Brasil vai muito mal" para "é que a televisão só mostra coisas ruins". Outro exemplo apontado foi quanto à culpabilização dos políticos ou da presidenta pela situação do país, passando também para diferentes opiniões como a de que "é o povo que tem que aprender a votar".

Além disso, pode-se refletir, a partir dos estudos sobre as práticas realizadas no ensino de programação e especificamente nas oficinas analisadas, que para o ensino de programação a nível médio a motivação é um item importante a ser desenvolvido e observado nos estudantes. Como já firmado por outros autores citados nesse trabalho como em (Leal, 2014), o estudante do nível médio ao se deparar com uma área com a qual não está adaptado, pode encontrar dificuldades, e demonstrar resistência ao aprendizado.

Desta forma, foi possível, de maneira geral, perceber que a motivação, além de estar atrelada ao trabalho de temas significativos para o estudante, também está relacionada a atividades que propiciem maior interação por meio da construção em conjunto. Ou seja, verifica-se o contrucionismo visado por Papert, em que para a construção do conhecimento acontecer de forma mais positiva os estudantes devem construir e compartilhar objetos publicamente, ou seja, os estudantes precisam estar conscientemente engajados na construção desta entidade pública (Papert and Harel, 2002), o que também contribui para uma interação social crítica dos indivíduos.

Ao se analisar as avaliações feitas pelos instrutores da oficina referentes a cada um dos estudantes (Apêndice F), foi possível traçar um perfil específico de modo a identificar as principais dificuldades que estes enfrentam. Além disso, foi possível relacionar, por meio do questionário que estes responderam, quais abordagens que melhor se adaptam quando se tratando da dificuldade que cada um destes enfrenta, conforme resumo apresentado pela Tabela 7.

Tabela 7: Práticas educacionais indicadas para dificuldades apresentadas

Aspecto a ser Trabalhado	**Abordagem/Prática Educacional**
Motivação	- Incentivo ao Conhecimento Significativo - Suporte à Colaboração (No sentido do trabalho em grupo) - Liberdade/Controle por Parte do Estudante (Como forma de incentivo à criatividade).
Foco	- Incentivo ao Conhecimento Significativo - Suporte à Colaboração (No sentido do trabalho em grupo)
Confiança	- Incentivo ao Conhecimento Significativo - Liberdade/Controle por Parte do Estudante
Adequação das Atividades ao Nível do Estudante	- Liberdade/Controle por Parte do Estudante (Como forma de incentivo à construção do conhecimento adequado) - Adaptação/Flexibilidade da Atividade

O Estudante 1 foi identificado como detentor de um baixo nível de conhecimento em programação, e na percepção dos avaliadores recomendaria-se para o mesmo a repetição da oficina de Python. Apesar de ambas as avaliações dos instrutores apontarem que este estudante tem como ponto positivo a facilidade na compreensão da solução lógica dos problemas, ou seja, tem um bom entendimento do que deve ser feito na atividade, a falta de motivação é apontada como pior dificuldade enfrentada pelo mesmo. Se algo parece desafiador, difícil ou desinteressante, este estudante tem tendência a ficar desmotivado a enfrentar o problema, diferentemente dos demais estudantes.

Desta forma, em sua avaliação, o Estudante 1 apontou alguns fatores diferenciais, e percebe-se que estes estão diretamente relacionados ao fator motivacional, como o nível de diversão, o uso da criatividade e da troca de ideias por meio de trabalho em grupo. Isto ficou evidenciado pelas respostas obtidas às perguntas 4, 5 e 6, sendo estas, respectivamente: preferência por "atividades que permitam maior liberdade para escolher o quê e como desenvolver durante a aula, porque gosto de usar a criatividade e ter uma discussão de ideias entre todos"; preferência por atividades em "dupla, pois fica mais divertido e um pode ajudar o outro"; e "Sim, pois é mais divertido e não é tão pesado só com programação", quando perguntado sobre o desejo em ter mais aulas como a realizada no dia.

Este estudante também enumerou que, dentre todas as aulas, a mais satisfatória foi a de "Contando Histórias com Animações" e a menos satisfatória do "Jogo da Velha". Desta forma, a abordagem proposta neste trabalho pode ser útil para estudantes com baixa motivação, uma vez que propõe maior liberdade, interação, e utilização de animações - contrariamente ao "Jogo da Velha", em que para se atingir o objetivo da atividade necessita-se obrigatoriamente que todos os passos sejam seguidos de forma fixa e predeterminada.

Quanto ao Estudante 2, este foi identificado como detentor de um baixo nível de conhecimento em programação, e na percepção dos avaliadores recomendaria-se para o mesmo a repetição da oficina de programação em Python. Ambos também identificaram como principal dificuldade deste a falta de foco, e nas palavras do Instrutor 6 este estudante tem "dificuldade quanto à lógica para a resolução dos problemas e a dispersão acaba dificultando o entendimento da necessidade de padronizar os códigos".

A dificuldade em se manter focado pode ser a explicação para a resposta dada por este quanto à preferência por atividades com roteiro e passos fixos/predefinidos em prol daquelas com maior liberdade, pois estas podem se configurar mais simples para estudantes com esta característica. Isto também reflete a escolha deste estudante como aula favorita a do "Jogo da Velha", uma vez que esta se configura como um conjunto de passos fortemente estruturados. Ainda assim, este também apontou a preferência por atividades em grupo, e em conjunto com o "Jogo da Velha", o mesmo apontou a aula de "Contando Histórias com Animações" como igualmente favorita. Em segundo lugar a aula de "Jokenpô" foi a escolhida, a qual também se relaciona por permitir que os estudantes interagissem entre si por meio de um jogo.

No que se refere ao Estudante 3, este foi identificado como detentor de um nível mediano em domínio de programação, precisando repetir a oficina apenas caso este deseje. No questionário, este estudante respondeu que de maneira geral prefere atividades que possibilitem maior liberdade, pois, de acordo com o mesmo, é possível assim "mostrar nossas ideias em um novo conteúdo". Este também classificou a aula de "Contando Histórias com Animações" como favorita, em conjunto com outras três: "Jogo da Velha", "Jokenpô" e "Código Morse".

A principal dificuldade do Estudante 3, de acordo com as avaliações dos instrutores da oficina, é a falta de confiança quanto aos conhecimentos que este possui. É possível que este fato justifique a preferência do Estudante 3 em trabalhar de maneira individual ao invés de em grupo, pois, uma vez que com baixa confiança, individualmente pode ser mais fácil de se reafirmar e colocar em prática o próprio conhecimento do que quando em dupla ou grupo.

O Estudante 4 é o que foi identificado com maior domínio de conhecimentos em programação pelos instrutores da oficina. Também verifica-se, no questionário avaliativo sobre este, que sua principal dificuldade é referente à adequação das atividades da oficina à experiência que ele já possuia. Portanto, é possível que a escolha da aula de "Jogo da Ve-

lha"como favorita tenha se dado por conta de ser mais longa e de complexa implementação, em termo de lógica de algoritmo, e assim desta forma, mais desafiadora.

Outro aspecto apontado como ponto a ser trabalhado pelo Estudante 4 foi a dispersão durante as atividades, o que pode se relacionar novamente com o fato de as atividades em geral se apresentarem em um nível de facilidade maior do que a capacidade do estudante. Quando perguntado sobre a abordagem favorita, este preferiu maior liberdade de escolha para desenvolver durante às aulas, pois de acordo com este as com passos fixos deixam "a aula muito objetiva, sem distrações". A aula de "Contando Histórias com Animações"foi elencada como quarta favorita deste estudante.

Por fim, as discussões aqui levantadas não devem ser tomadas como única via para a produção de resultados, uma vez que as variáveis estudadas nessa pesquisa não são estáticas por se tratar de seres humanos. Apesar da breve análise dos estudantes realizada com base na avaliação dos instrutores, ainda assim para resultados mais conclusivos seria necessário se levar em consideração alguns fatores aqui descartados, como o psicológico e questões pontuais de cada estudante, o que pode influenciar no dia a dia desses, e com isso alterar a percepção e sentimento em relação a uma determinada aula.

Deste modo, reitera-se também a influência no desenvolvimento quanto ao domínio de conhecimentos prévios à oficina, como facilidade em assimilar conceitos matemáticos, a facilidade ou não de interação nas atividades desenvolvidas em grupo, que pode tanto favorecer a afeição por uma atividade quanto provocar o efeito contrário, dependendo da personalidade do estudante, e as experiências pessoais de cada um, que podem favorecer ou não no processo de aprendizagem.

6 Considerações Finais

O presente trabalho visou o desenvolvimento de práticas para fomentar e desmistificar o ensino de programação no nível médio.

Para isto, proporcionou-se uma análise dos métodos de ensino de programação já utilizados nas oficinas para estudantes do ensino médio, que dispuseram da linguagem Python e foram ofertadas nos anos de 2012, 2013, e 2015 na UTFPR, no campus Curitiba por estudantes vinculados ao PET-CoCe e ao programa Compute Você Mesm@.

Desta forma, foram definidas as práticas educacionais compatíveis com cada método, verificando-se que a oficina mais atual abordada neste trabalho se difere das anteriores, uma vez que decorre de metodologias de ensino diferentes: as dos anos de 2012 e 2013 se voltaram para uma abordagem comportamentalista, enquanto a partir de 2015 optou-se por uma visão mais construtivista de ensino. Este trabalho também aliou cada método estudado a uma ferramenta de ensino de programação, as quais foram descritas na secção 3.2 conforme as

visões conducionistas e construtivistas do ensino.

Considerando-se as diretrizes educacionais para a informática na educação e as bases do curso de BSI, este trabalho julgou como apropriadas para o público alvo visado as práticas educacionais definidas pela intersecção das filosofias construcionistas e socioculturalistas de ensino. Desta forma, como apresentado na seção 4.1, indica-se como práticas norteadoras a liberdade oferecida ao estudante para decidir como ou o quê desenvolver, ou seja, o controle quanto à construção do seu conhecimento, aliando-se também às possibilidades de engajamento através da colaboração para a construção da entidade pública/compartilhada. Além disso, as práticas devem buscar envolver o cotidiano do estudante, de forma a respeitar sua individualidade ao possibilitar que o conhecimento abordado seja significativo e adequado para os mesmos.

Estas práticas, decorrentes da visão metodológica de ensino escolhida, também serviram como base para a seleção do modelo de avaliação do software educacional TUP, descrito na secção 4.2. Por meio deste, foi possível realizar a avaliação das três ferramentas educacionais que foram julgadas mais adequadas em 4.3: Alice, Pygame e Scratch.

Mediante aula realizada com o público alvo da oficina atual de Python, a metodologia selecionada pode ser aplicada conjuntamente com as práticas e a ferramenta Pygame. Desta forma, uma vez analisados os resultados obtidos, verificou-se a importância da utilização de uma metodologia de ensino ajustada às necessidades e perfis dos estudantes envolvidos. Esta verificação se deu tanto por meio das considerações quanto às oficinas anteriormente aplicadas, quanto por conta da aula prática realizada na oficina atual. Da mesma forma, foi possível concluir que as práticas verificadas não apenas cumpriram com os objetivos de aprendizagem visados, mas também permitiram a criação de espaços para que o pensamento crítico e reflexivo se desenvolvesse, além de contribuírem para a promoção da criatividade e motivação dos estudantes.

Como trabalhos futuros podem ser indicados a aplicação da metodologia de ensino selecionada por meio das outras duas ferramentas visadas neste trabalho: Alice e Scratch. Além disso, indica-se também a validação, por meio da realização de uma oficina, da visão metodológica de ensino escolhida neste trabalho, com o diferencial de se aplicar em uma turma de ensino médio que nunca teve contato com o ensino de programação. Desta forma, verificaria-se desde o início, dado um ambiente sem base ou prévia estruturação do conhecimento de programação, o quão eficaz a metodologia selecionada se apresenta, mesmo que em ambiente adverso ao que se deu neste trabalho.

Referências

Abramof, P. G. and Mota, A. C. (2007). *Recursos Computacionais na Integração de Disciplinas em Cursos de Engenharia*. da Vinci, Curitiba, v. 4 , n. 1, p. 147-158, 2007.

Abreu, A. S. et al. (1997). Abordagens do processo ensino-apreendizagem e o profeessor. `http://www.angelfire.com/ak2/jamalves/Abordagem.html`. [Online; acesso 5-abril-2015].

Aguiar, D. R. C. (2011). Pesquisando a flexibilidade curricular: um outro jeito de fazer educação, currículo e ensino na perspectiva freireana. `https://www.fe.unicamp.br/gtcurriculoanped/35RA/trabalhos/TE-Anped2012-flexibilidade_curr.pdf`. [Online; acesso 20-Maio-2015].

Allen, E. et al. (2012). Drjava: A lightweight pedagogic environment for java. `http://www.cs.rice.edu/~javaplt/papers/sigcse2002.pdf`. [Online; acesso 12-janeiro-2015].

Almeida, M. B. (2012). Noções básicas sobre metodologia de pesquisa. `http://mba.eci.ufmg.br/downloads/metodologia.pdf`. [Online; acesso 25-novembro-2014].

Alves, L. M. F. (2006). Análise de softwares educacionais. `http://www.uel.br/seed/nte/analisedesoftwares.html`. [Online; acesso 28-junho-2015].

Auburn, U. (2015). About jgrasp - an integrated development environment with visualizations for improving software comprehensibility. `http://www.jgrasp.org/index.html`. [Online; acesso 10-janeiro-2015].

Bahia, U. F. (2007). A construção do conhecimento na visão de piaget. `http://www.moodle.ufba.br/mod/book/view.php?id=10910&chapterid=9878`. [Online; acesso 12-Julho-2015].

Barreiros, E. F. S. et al. (2014). *Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência*. WEI – XXII Workshop sobre Educação em Computação.

Bednarik, R. (2002). Evaluation of educational environments - the tup model. `ftp://cs.joensuu.fi/pub/Theses/2002_MSc_Bednarik_Roman.pdf`. [Online; acesso 05-Julho-2015].

Bednarik, R. (2013). Tup online. `http://cs.joensuu.fi/~tup/`. [Online; acesso 10-Julho-2015].

Bergin, J. et al. (2002). A gentle introduction to the art of object-oriented programming in java. `http://www.csis.pace.edu/~bergin/KarelJava2ed/preface.html`. [Online; acesso 11-janeiro-2015].

Bertoldi, S. (1999). *Avaliação de Software Educacional - Impressões e Reflexões*. Universidade Federal de Santa Catarina - Florianópolis.

Blinkstein, P. (2004). Travels in troy with freire: Technology as an agent of emancipation. http://www.blikstein.com/paulo/documents/books/Blikstein-TravelsInTroyWithFreire.pdf. [Online; acesso 12-Julho-2015].

Bonin, L. F. R. (2008). Educação, consciência e cidadania. http://books.scielo.org/id/hn3q6/pdf/silveira-9788599662885-10.pdf. [Online; acesso 20-Maio-2015].

Bower, G. H. and Hilgard, E. R. (1981). *Theories of learning*. Englewood Cliffs, NJ: Prentice-Hall,Stanford University, CA.

Brasil, C. (2012). Ideias para a educação - revista da sociedade brasileira de computação. http://www.sbc.org.br/downloads/CB2012/computacao_ago_05_11.pdf. [Online; acesso 25-novembro-2014].

Campos, G. H. B. and Campos, F. C. A. (2001). *Qualidade de Software Educacional*. Qualidade de software: Teoria e Prática. Ed. Campinas: Makron, 2001.

Canterbury, U. K. (2015). Greenfoot - a programming education tool. http://www.greenfoot.org/. [Online; acesso 03-janeiro-2015].

Ciência, C. (2011). Paulo freire. http://www.canalciencia.ibict.br/notaveis/livros/paulo_freire_36.html. [Online; acesso 11-janeiro-2015].

Cooper, S. et al. (2003). Using animated 3d graphics to prepare novices for cs1. http://web.stanford.edu/~coopers/alice/scooper_csej.pdf. [Online; acesso 11-janeiro-2015].

Cortez and Moraes (1979). *Paulo Freire e Conscientização*. Câmara Brasileira do Livro.

Craven, P. V. (2015). Program arcade games with python and pygame. http://programarcadegames.com/index.php. [Online; acesso 20-janeiro-2015].

de Souza Rezende, C. (2013). Modelo de avaliação de qualidade de software educacional para o ensino de ciências. http://saturno.unifei.edu.br/bim/0040322.pdf. [Online; acesso 05-Julho-2015].

Dorn, B. and Sanders, D. (2003). Using jeroo to introduce object-oriented programming. http://www.cs.hartford.edu/~dorn/papers/FIE03.pdf. [Online; acesso 06-janeiro-2015].

Educação, M. M. D. (2013). Apresentação Programa de Educação Tutorial - PET. http://portal.mec.gov.br/index.php?option=com_content&view=article&id=12223&ativo=481&Itemid=480/. [Online; acesso 8-Julho-2015].

Exame, R. (2014). Há demanada crescente de profissionais de TI no país— – Editora Abril S.A. http://exame.abril.com.br/carreira/noticias/ha-demanda-crescente-por-profissionais-de-ti-no-pais. [Online; acesso 2-novembro-2014].

Fazenda, I. C. A. (2011). *Integração e Interdisciplinaridade no Ensino Brasileiro*. Editora Loyola.

Ferreira, M. L. S. (2008). A formação do cidadão crítico, criativo, participativo: um discurso aquem da prática. http://alb.com.br/arquivo-morto/edicoes_anteriores/anais16/sem12pdf/sm12ss11_01.pdf. [Online; acesso 20-Maio-2015].

Fiolhais, C. and Trindade, J. (2003). Física no computador: o computador como uma no ensino e na aprendizagem das ciências físicas. Revista Brasileira de Ensino de Física vol. 25, no. 3, Setembro 259.

Fior, C. A. and Mercuri, E. (2009). Formação universitária e flexibilidade curricular: importância das atividades obrigatórias e não obrigatórias. http://pepsic.bvsalud.org/pdf/psie/n29/n29a10.pdf. [Online; acesso 19-Maio-2015].

FNDE (2012). Apresentação proinfo. http://www.fnde.gov.br/programas/programa-nacional-de-tecnologia-educacional-proinfo. [Online; acesso 05-Junho-2015].

Fontes, C. (2014). Teorias de aprendizagem e software educativo. http://educar.no.sapo.pt/teorias.htm. [Online; acesso 12-janeiro-2015].

Fortes, C. C. (2007). *Interdisciplinaridade: Origem, Conceito e Valor*.

Foundation, L. (2015a). Logo and learning. http://el.media.mit.edu/logo-foundation/what_is_logo/logo_and_learning.html. [Online; acesso 12-Maio-2015].

Foundation, L. (2015b). Logo history. http://el.media.mit.edu/logo-foundation/what_is_logo/history.html. [Online; acesso 12-Maio-2015].

Foundation, L. (2015c). A logo primer. http://el.media.mit.edu/logo-foundation/what_is_logo/logo_primer.html. [Online; acesso 12-Maio-2015].

Foundation, L. (2015d). Logo programming. http://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html. [Online; acesso 12-Maio-2015].

Foundation, N. S. et al. (2015). About drjava. http://mba.eci.ufmg.br/downloads/metodologia.pdf. [Online; acesso 12-janeiro-2015].

França, R. S. D. et al. (Março 2013). *Despertando o interesse pela ciência da computação: Práticas na educação básica*. VIII International Conference on Engineering and Computer Education.

Giraffa, L. M. M. (2009). *Uma odisséia no ciberespaço: O software educacional dos tutoriais aos mundos virtuais*. Revista Brasileira de Informática na Educação, Volume 17, Número 1, 2009.

Graeml, A. et al. (2008). *Projeto de Abertura do Curso de Bacharelado em Sistemas de Informação*. Universidade Tecnologica Federal do Parana.

Hendrix, T. D. et al. (2004). An extensible framework for providing dynamic data structure visualizations in a lightweight ide. http://www.jgrasp.org/papers/sigcse2004paper.hendrix.cross.barowski.pdf. [Online; acesso 10-janeiro-2015].

Henriksen, P. and Kolling, M. (2003). Greenfoot: Combining object visualisation with interaction. https://kar.kent.ac.uk/14059/1/greenfoot.pdf. [Online; acesso 04-janeiro-2015].

IBM (2003). Robocode tutorial. http://www.ibm.com/developerworks/java/library/j-robocode/j-robocode-pdf.pdf. [Online; acesso 06-janeiro-2015].

IDC (2014). About idc latin america. http://www.idclatin.com/about/whyidc. [Online; acesso 2-novembro-2014].

in St. Louis, W. U. (2015). The jpie project. http://jpie.cse.wustl.edu/. [Online; acesso 10-janeiro-2015].

Jenkins, T. (Setembro 2002). *On the Difficulty of Learning to Program*. University of Leeds.

John Maloney, M. R. et al. (2010). The scratch programming language and environment. http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf. [Online; acesso 06-janeiro-2015].

Krausz, M. (2011). *Onde as disciplinas se encontram*. revistaeducacao.

Lab, M. M. (2015). Scratch - a creative learning community. http://scratch.mit.edu/. [Online; acesso 06-janeiro-2015].

Leal, A. V. D. A. (Março 2014). *Ensino de Programação no Ensino Médio Integrado:Uma Abordagem Utilizando Padrões e Jogos com Materiais Concretos*. Universidade Federal de Goiás.

Marques, D. L. et al. (2010). Atraindo alunos do ensino médio para a computação: Uma experiência prática de introdução a programação utilizando jogos e python. http://www.br-ie.org/pub/index.php/wie/article/view/1954/1713. [Online; acesso 20-janeiro-2015].

Marques, N. L. R. (2009). Teorias de aprendizagem. http://www.nelsonreyes.com.br/MET_ENS_FII_T_%20APRENDIZAGEM.pdf. [Online; acesso 09-Abril-2015].

Marques, N. L. R. et al. (2009). *Teorias de Aprendizagem*. Instituto Federal De Educação,Ciência e Tecnologia Do Rio Grande Do Sul – CAMPUS CAVG.

Mellon, U. C. (2015). Alice - an educational software that teaches students computer programming in a 3d environment. http://www.alice.org/index.php. [Online; acesso 11-janeiro-2015].

Melo, A. P. D. et al. (2010). *Diretrizes para o Uso de Tecnologias Educacionais no Estado do Paraná*. Governo do Estado do Paraná, Secretaria de Estado da Educação.

Micheletto, I. B. P. (2007). Ação-reflexão-ação: Processo de formação continuada. http://www.diaadiaeducacao.pr.gov.br/portals/pde/arquivos/1448-6.pdf. [Online; acesso 12-Julho-2015].

Mizukami, M. G. N. (1986). *Ensino: As abordagens do processo*. E.P.U.

Moreira, M. A. (2009). *comportamentalismo,Construtivismo e Humanismo*. Subsídios Teóricos para o Professor Pesquisador em Ensino de Ciências.

Moreira, M. A. (2011). *Teorias de Aprendizagem*. Editora Pedagógica e Universitária - EPU.

Nebraska, U. (2015). Jeroo learning tool. http://home.cc.gatech.edu/dorn/38. [Online; acesso 06-janeiro-2015].

Nelson, M. (2015). Robocode programming game. http://robocode.sourceforge.net/. [Online; acesso 08-janeiro-2015].

Neto, A. C. et al. (2004). *Flexibilização curricular: cenários e desafios*. EDUFRN - Editora da Universidade Federal do Rio Grande do Norte.

of Kent, U. (2012). Bluej. http://www.bluej.org/. [Online; acesso 12-janeiro-2015].

Oliveira, E. (2010). *Interdisciplinaridade*. infoescola.com.

Ostermann, F. et al. (2010). *Teorias de Aprendizagem*. Universidade Federal Do Rio Grande Do Sul – Instituto De Física.

Papert, S. and Harel, I. (2002). Situating constructionism. http://web.media.mit.edu/~calla/web_comunidad/Reading-En/situating_constructionism.pdf. [Online; acesso 15-Julho-2015].

Pattis, R. E. (2015). Karel the robot. https://www.cs.mtsu.edu/~untch/karel/fundamentals.html. [Online; acesso 11-janeiro-2015].

Pears, A. et al. (2007). *A Survey of Literature on the Teaching of Introductory Programming*. University of Pennsylvania.

Pessoa, T. and Nogueira, F. (2009). *Flexibilidade Cognitiva nas Vivêncuas e Práticas Educativas*. EDUFBA - Editora da Universidade Federal da Bahia.

Pimentel, U. (2002). *Repensando a Educação na era da Internet - Teorias de Aprendizagem*. Universidade Federal do Estado do Rio de Janeiro.

Prass, A. R. (2012). *Teorias de Aprendizagem*. ScriniaLibris.com.

PythonBrasil, C. (2015). Python: O que é? por que usar? http://pyscience-brasil.wikidot.com/python:python-oq-e-pq. [Online; acesso 12-Fevereiro-2015].

Píccolo, H. L. et al. (2010). Ambiente interativo e adaptável para ensino de programação. http://www.br-ie.org/pub/index.php/sbie/article/view/1496/1261. [Online; acesso 11-janeiro-2015].

Quig, B. et al. (2003). The bluej system and its pedagogy. http://www.bluej.org/papers/2003-12-CSEd-bluej.pdf. [Online; acesso 12-janeiro-2015].

Rebouças, A. D. D. S. et al. (2010). Aprendendo a ensinar programação combinando jogos e python. http://www.lbd.dcc.ufmg.br/colecoes/sbie/2010/009.pdf. [Online; acesso 20-janeiro-2015].

Report, M. (Setembro 2012). *A National Talent Strategy. Ideas for securing U.S competitiveness and economy growth*. Microsoft.

Revista, N. E. (1995). *50 Questões Básicas sobre Construtivismo*. Revista Nova Escola.

Ribeiro, M. N. B. (2013a). Avaliação da qualidade de software educacional. http://www.fafism.com.br/avaliacao/autor.html. [Online; acesso 05-Julho-2015].

Ribeiro, M. N. B. (2013b). Ferramenta de avaliação da qualidade de software educacional. http://www.fafism.com.br/avaliacao/avaliacao.html. [Online; acesso 10-Julho-2015].

Ribeiro, M. N. B. (2013c). Informática educacional. http://www.fafism.com.br/avaliacao/index.html. [Online; acesso 10-Julho-2015].

Ribeiro, M. N. B. (2013d). Software educacional. http://www.fafism.com.br/avaliacao/software_edu.html. [Online; acesso 10-Julho-2015].

Rocha, A. R. and de Campos, G. H. B. (1993). *Avaliação da qualidade de software educacional*. Em Aberto, Brasília, ano 12, n.57, jan./mar. 1993.

Rocha, S. F. M. and Rocha, J. H. M. V. (2010). *A interdisciplinaridade em Paulo Freire: Reflexões em defesa do diálogo disciplinar na Educação.* infoescola.com.

Rogers, C. R. (1978). Liberdade de aprender. Interlivros,Trad.De Edgar Godoi de Mata Machado.

Rossetto, E. and Brabo, G. (2007). A constituição do sujeito e a subjetividade a partir de vygotsky: algumas reflexões. http://e-revista.unioeste.br/index.php/travessias/article/download/3238/2553. [Online; acesso 20-Junho-2015].

Salles, C. M. C. (2012). *A Aprendizagem Significativa e as Novas Tecnologias na Educação a Distância*. Universiodade FUMEC - FACE.

Santana, A. L. (2010). Metodo de Educação crítico-social. http://www.infoescola.com/pedagogia/metodo-de-educacao-critico-social/. [Online; acesso 26-maio-2015].

Santos, C. M. B. and Marques, J. T. (2009). Buscando a construção e (re)construção da práxis pedagógica. http://www.faesi.com.br/nucleo-de-pesquisa-cientifica/75-portal-do-saber/234-buscando-a-construcao-e-reconstrucao-da-praxis-pedagogica. [Online; acesso 10-Julho-2015].

Santos, R. V. (2005). Abordagens do ensino e aprendizagem. ftp://www.usjt.br/pub/revint/19_40.pdf. [Online; acesso 09-Abril-2015].

Scaico, P. D. et al. (2013). *Ensino de Programação no Ensino Médio: Uma Abordagem Orientada ao Design com a linguagem Scratch*. XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014.

Shinners, P. (2015). Writing games with pygame. http://www.pygame.org/. [Online; acesso 08-janeiro-2015].

Tonet, B. (2007). *Tutorial Visualg*. NAPRO - Núcleo de Apoio e Aprendizagem de Programação.

Treviso, V. C. and de Almeida, J. L. V. (2014). O conhecimento em jean piaget e a educação escolar. http://unifafibe.com.br/revistasonline/arquivos/cadernodeeducacao/sumario/31/04042014074544.pdf. [Online; acesso 19-Maio-2015].

Untch, R. H. (1999). Teaching programming using the karel the robot paradigm realized with aconventional language. http://files.eric.ed.gov/fulltext/ED329231.pdf#page=251. [Online; acesso 11-janeiro-2015].

Vieira, F. M. S. (2010). *Avaliação de Software Educativo: Reflexões para uma Análise Criteriosa*.

Wilson, C. (2014). Brainstorming and beyond: A user-centered design method. http://scitechconnect.elsevier.com/wp-content/uploads/2015/05/Brainstorm-and-beyond-ch1.pdf. [Online; acesso 15-Julho-2015].

Wing, J. M. (Março 2006). Communications of the acm. https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf.

Zacharias, V. L. C. (2008). *Burrhus Frederic Skinner*. Universidade Federal do Estado do Rio de Janeiro.

A Comparação das Ferramentas de Software Educacional

	ALICE		PYGAME		SCRATCH	
Question	**Answer**	**Comment**	**Answer**	**Comment**	**Answer**	**Comment**
Technology						
Does the system support the import/export of external resources?	YES		YES		YES	
Is it possible to use the tool on different operating systems?	YES		YES		YES	
Does the tool need additional software or components to be installed? If YES, please specify which ones.	NO		NO		YES	Adobe AIR for Scrath 2.0
Does the tool satisfy the requirements on a open source software (e.g. GPL)?	YES		YES		YES	
Does the system need some additional equipment to function properly?	NO		NO		NO	
Do you need to use unusual external interfaces, devices or media?	NO		NO		NO	
Is the required equipment readily available?	NO		NO		NO	
It is easy to set up the environments necessary for the tool.	Strongly agree		Mostly disagree		Neutral	
It is easy to install the tool (e.g. by an installation wizard).	Strongly agree		Mostly disagree		Mostly agree	
Is the system adaptable to the needs of disabled people?	NO		NO		NO	
Does the system distinguish between the different age groups of users?	NO		NO		NO	
Does the system follow the regional setting, e.g. format of numbers, time format, keyboard?	YES		YES		YES	
Is it possible to change the language of the environment?	YES	Algumas palavras ainda permanecem em inglês	NO		YES	
Is it easy to maintain the system?	YES		YES		YES	
The system requires much maintenance.	Mostly disagree		Mostly disagree		Mostly agree	
The system is easy to administrate.	Strongly agree		Mostly agree		Mostly agree	
Is it necessary to train personnel in order to use the tool?	NO		NO		YES	
Is it expensive to purchase and run the system?	NO		NO		NO	
Does the system fit into the organization's technological structure?	YES		YES		YES	
Is the privacy for users guaranteed?	NO		NO		NO	
Are the security measures adequate for the system?	NA		NA		NA	
Did you experience any health risks while using the tool?	NO		NO		NO	
If any fault occurs, does it cause the breakdown of the system?	NO		YES		NO	
Does the system actively prevent the faults?	NO		NA		NO	
The system is free of technological defects leading to malfunctions.	YES		NO		NO	
The system offers diagnostics tools in order to find possible hardware or software defects.	NO		NO		NO	
Usability						
Users can rapidly start working with the tool without a long period of training.	Strongly agree		Mostly agree		Mostly agree	
Does the usage of the tool impose heavy cognitive load to the users and thus causing problems?	NO		NO		NO	
Is is possible to select advanced modes of control according to the user's experience?	NO		NO		YES	
Does the tool offer various interaction modes (e.g. sound, haptic channel) ?	Mostly agree		Mostly agree		Strongly agree	
The level of interaction with the tool corresponds to the users' characteristics	Strongly agree		Mostly agree		Mostly agree	
Are the means of interaction properly selected?	YES		NO		YES	
Do the users have the ability to 'undo' and 'redo' their actions?	YES		YES		YES	
Is the feedback offered by the tool within a reasonable time?	YES		YES		YES	
Does the system offer shortcuts for the most often invoked commands or sequences?	YES		NO		YES	
Does the system offer any help facility?	YES		YES		YES	

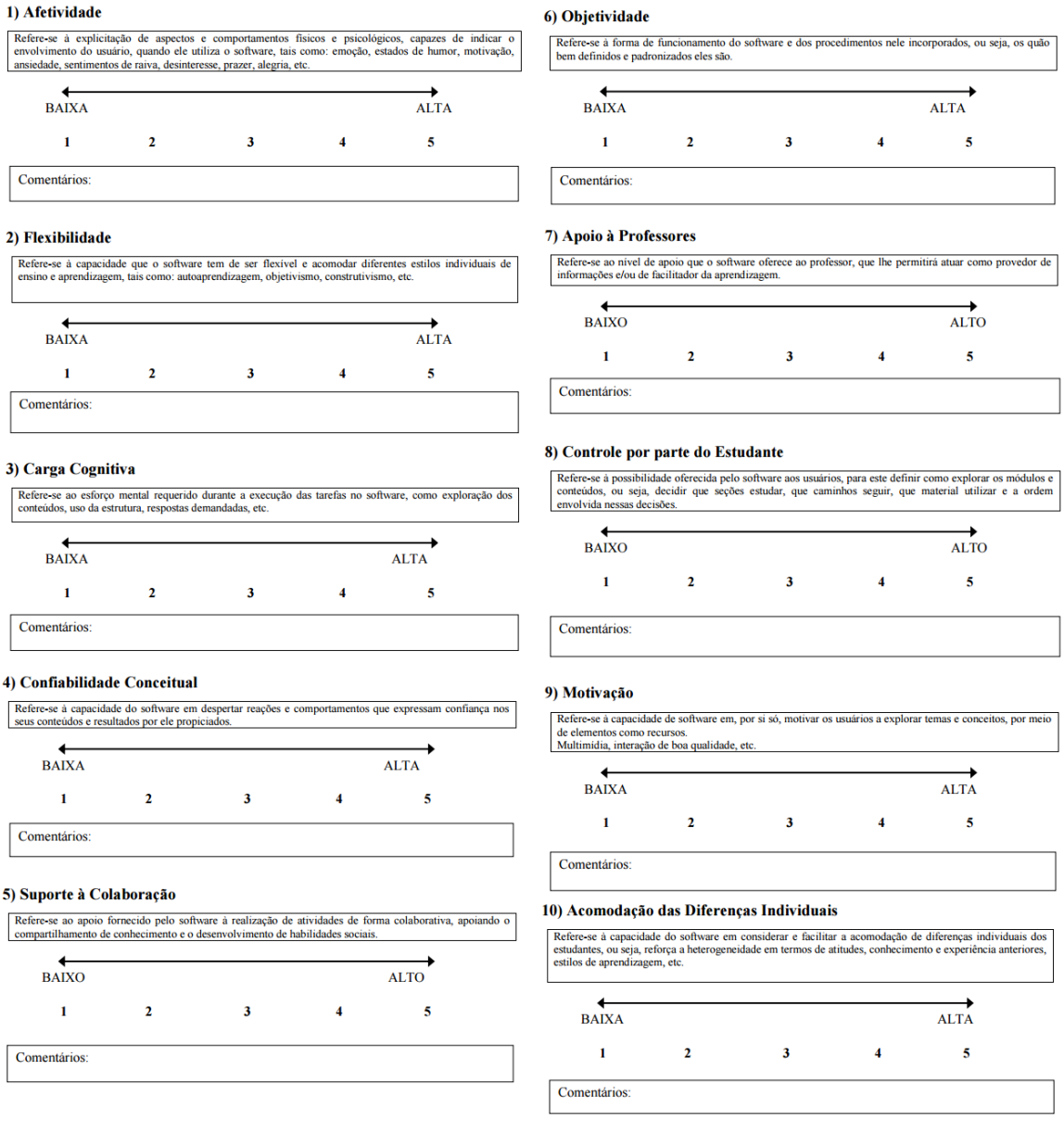
Is the help easily accessible?	YES		YES		NO	
Are users always properly informed about their position in the system structure?	YES		NO		NO	
Navigation within the system is easy and natural for users (e.g. the structure of menu).	Strongly agree		Neutral		Neutral	
Are users facilitated to search within the environment?	YES		NO		NO	
Does the tool require re-training of usage after breaks?	NO		NO		NO	
Does the environment require users to memorize facts unrelated to the learning objectives?	NO		YES		NO	
The design of the interface is aesthetic.	Mostly agree		Mostly disagree		Neutral	
Is the text used in the environment clearly legible?	YES		YES		YES	
The tool uses graphics appropriately.	Mostly agree		Mostly disagree		Neutral	
The displayed information is properly organized.	Strongly agree		Mostly disagree		Neutral	
The tool uses sound appropriately.	Neutral		Neutral		Strongly agree	
Is the environment consistent in terms of design, navigation and terminology?	YES		YES		YES	
The designer's model of system corresponds with the user's view, expectations and perception of it.	Strongly agree		Mostly agree		Mostly agree	
The system complies with the local setting and habits.	Mostly agree		Neutral		Mostly agree	
Is it possible to use the system by multiple nationalities without restrictions?	YES		YES		YES	
Is it possible to use the system by various age groups?	YES		YES		YES	
Pedagogy						
Is the subject of learning presented in an authentic context?	YES		NO		YES	
Usage of the tool fits into the learning setting (e.g. class versus individual learning)	Strongly agree		Mostly agree		Strongly agree	
The tool conveniently complements other class activities.	Strongly agree		Strongly agree		Strongly agree	
The content fits into the institutional curriculum.	Strongly agree		Strongly agree		Mostly agree	
Does the system distinguish between the roles of the users in the learning process?	Strongly disagree		Strongly disagree		Mostly disagree	
Is the environment self-adaptable in order to reflect learners' growth?	YES		NO		YES	
Does the system enable the customization of the perspectives of learning objects according to the learners' n	YES		YES		YES	
Cultural issues are addressed and properly handled by the environment.	Neutral		Mostly disagree		Neutral	
Are the resources and references used in the tool credible?	YES		YES		YES	
Is the information presented by the tool up to date?	YES		YES		NA	
Do you feel the tool maintains trust in it?	YES		NO		YES	
Learners' ownership of learning is maintained by the environment.	Mostly agree		Mostly agree		Mostly agree	
Are the tasks designed to maintain (or increase) learners' motivation?	YES		YES		YES	
Is the selection of the learning goals appropriate?	YES		NO		YES	
Is it possible to modify the learning goals?	YES		YES		YES	
Does tool enable multiple paths to achieve the learning goals?	YES		NO		NO	
Are the task sequences adaptable according to the learners' growth?	YES		YES		YES	
The level of abstraction corresponds with the learners' growth.	Strongly agree		Mostly agree		Neutral	
Learning tasks correspond with the real world matching.	Strongly agree		Mostly agree		Neutral	
Does the system provide enough means for the proper knowledge representation?	YES		YES		YES	
Does the environment provide multiple representation of knowledge?	YES		YES		NO	
Can the teacher prepare and modify learning materials using the tool?	YES		YES		YES	
Can the teachers share learning materials using the tool?	YES		NO		YES	

Is it possible for users to prepare a presentation using this environment?	YES		YES		YES	
Does the tool facilitate the evaluation of the learning process.	YES		NO		YES	
Is it possible to create tests, examinations or assignments using the tool?	YES		YES		YES	
Does the tool provide process management to be secure for all the parties?	NA		NA		NA	
The process management follows organizational requirements.	Neutral		Neutral		Neutral	
Does the tool offer self-directed learning?	NO		NO		NO	
Does the tool enable different learning styles (e.g. learning by exploration, learning by doing etc.)?	YES		YES		YES	
Does the tool facilitate groupwork?	YES		NO		NO	
Is it possible to create notes or bookmarks within the tool?	NO		NO		YES	
The tool supports creation and sharing of learning artifacts.	Strongly agree		Neutral		Neutral	
Does the tool support off-computer learning activities (e.g. motivates discussion)?	NO		NO		YES	
The layout of the interface supports attaining the learning objectives.	Mostly agree		Mostly agree		Neutral	
The interface layout contains all of the information necessary to achieve learning goals.	Mostly agree		Neutral		Mostly agree	
The interface of the tool is designed according to the target learners' needs.	Mostly agree		Neutral		Mostly agree	
Is it possible to tailor the interface according to the individual learners' needs?	YES		NO		YES	
While interacting with learners, does the tool accept also alternative responses?	NO		NO		YES	
Do you attain your learning objectives with the tool?	YES		YES		YES	
General						
Would you recommend the tool for learning purposes?	YES		YES		YES	
What do you think about the educational capabilities of the tool?	Por ter representação visual em 3D, tem um forte apelo visual, o que pode ser bastante motivador. É rico em recursos e simples de usar, o que potencializa a sua capacidade educacional.		Apresenta recursos que facilitam a aprendizagem, como a facilidade de programar animações. Entretanto, apesar de ainda em um nível básico, este contato pode não vir a ser tão intuitivo para alunos que ainda não tiveram nenhum contato com a programação.		A versão avaliada permite explorar a criatividade, além de existirem outras versões para outras faixas etárias (como o scratch para pais). Essa versão permite a interação e aprendizagem em qualquer nível.	

B Avaliação da Ferramenta Pygame

	PYGAME - Antes Oficina		PYGAME - Depois Oficina	
Question	**Answer**	**Comment**	**Answer**	**Comment**
Technology				
Does the tool need additional software or components to be installed? If YES, please specify which on	NO		YES	Para MAC: HOMEBREW, PIP, SMPEG, HG
It is easy to set up the environments necessary for the tool.	Mostly disagree		Strongly disagree	Não, pois tudo deve ser feito por linha de comando, além de compatibilidade de versões.
It is easy to install the tool (e.g. by an installation wizard).	Mostly disagree		Strongly disagree	
Is it easy to maintain the system?	YES		NO	
The system requires much maintenance.	Mostly disagree		Mostly disagree	
The system is easy to administrate.	Mostly agree		Mostly disagree	
Is it necessary to train personnel in order to use the tool?	NO		YES	
Usability				
Users can rapidly start working with the tool without a long period of training.	Mostly agree		Neutral	
Navigation within the system is easy and natural for users (e.g. the structure of menu).	Neutral		Mostly disagree	
Does the tool require re-training of usage after breaks?	NO		YES	
Pedagogy				
The tool supports creation and sharing of learning artifacts.	Neutral		Mostly disagree	
The layout of the interface supports attaining the learning objectives.	Mostly agree		Neutral	
The interface layout contains all of the information necessary to achieve learning goals.	Neutral		Mostly disagree	
The interface of the tool is designed according to the target learners' needs.	Neutral		Mostly disagree	
General				
Would you recommend the tool for learning purposes?	YES		NO	
What do you think about the educational capabilities of the tool?	Apresenta recursos que facilitam a aprendizagem, como a facilidade de programar animações. Entretanto, apesar de ainda em um nível básico, este contato pode não vir a ser tão intuitivo para alunos que ainda não tiveram nenhum contato com a programação.		Apresenta recursos que facilitam a aprendizagem, como a facilidade de programar animações. Entretanto, apesar de ainda em um nível básico, este contato pode não vir a ser tão intuitivo para alunos que ainda não tiveram nenhum contato com a programação. Também não apresenta fácil instalação.	

C Questionário Pectus



1) Afetividade

Refere-se à explicitação de aspectos e comportamentos físicos e psicológicos, capazes de indicar o envolvimento do usuário, quando ele utiliza o software, tais como: emoção, estados de humor, motivação, ansiedade, sentimentos de raiva, desinteresse, prazer, alegria, etc.

BAIXA — ALTA

1 2 3 4 5

Comentários:

2) Flexibilidade

Refere-se à capacidade que o software tem de ser flexível e acomodar diferentes estilos individuais de ensino e aprendizagem, tais como: autoaprendizagem, objetivismo, construtivismo, etc.

BAIXA — ALTA

1 2 3 4 5

Comentários:

3) Carga Cognitiva

Refere-se ao esforço mental requerido durante a execução das tarefas no software, como exploração dos conteúdos, uso da estrutura, respostas demandadas, etc.

BAIXA — ALTA

1 2 3 4 5

Comentários:

4) Confiabilidade Conceitual

Refere-se à capacidade do software em despertar reações e comportamentos que expressam confiança nos seus conteúdos e resultados por ele propiciados.

BAIXA — ALTA

1 2 3 4 5

Comentários:

5) Suporte à Colaboração

Refere-se ao apoio fornecido pelo software à realização de atividades de forma colaborativa, apoiando o compartilhamento de conhecimento e o desenvolvimento de habilidades sociais.

BAIXO — ALTO

1 2 3 4 5

Comentários:

6) Objetividade

Refere-se à forma de funcionamento do software e dos procedimentos nele incorporados, ou seja, os quão bem definidos e padronizados eles são.

BAIXA — ALTA

1 2 3 4 5

Comentários:

7) Apoio à Professores

Refere-se ao nível de apoio que o software oferece ao professor, que lhe permitirá atuar como provedor de informações e/ou de facilitador da aprendizagem.

BAIXO — ALTO

1 2 3 4 5

Comentários:

8) Controle por parte do Estudante

Refere-se à possibilidade oferecida pelo software aos usuários, para este definir como explorar os módulos e conteúdos, ou seja, decidir que seções estudar, que caminhos seguir, que material utilizar e a ordem envolvida nessas decisões.

BAIXO — ALTO

1 2 3 4 5

Comentários:

9) Motivação

Refere-se à capacidade de software em, por si só, motivar os usuários a explorar temas e conceitos, por meio de elementos como recursos.
Multimídia, interação de boa qualidade, etc.

BAIXA — ALTA

1 2 3 4 5

Comentários:

10) Acomodação das Diferenças Individuais

Refere-se à capacidade do software em considerar e facilitar a acomodação de diferenças individuais dos estudantes, ou seja, reforça a heterogeneidade em termos de atitudes, conhecimento e experiência anteriores, estilos de aprendizagem, etc.

BAIXA — ALTA

1 2 3 4 5

Comentários:

Figura 59: Questionário Aspectos Pedagógicos: Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

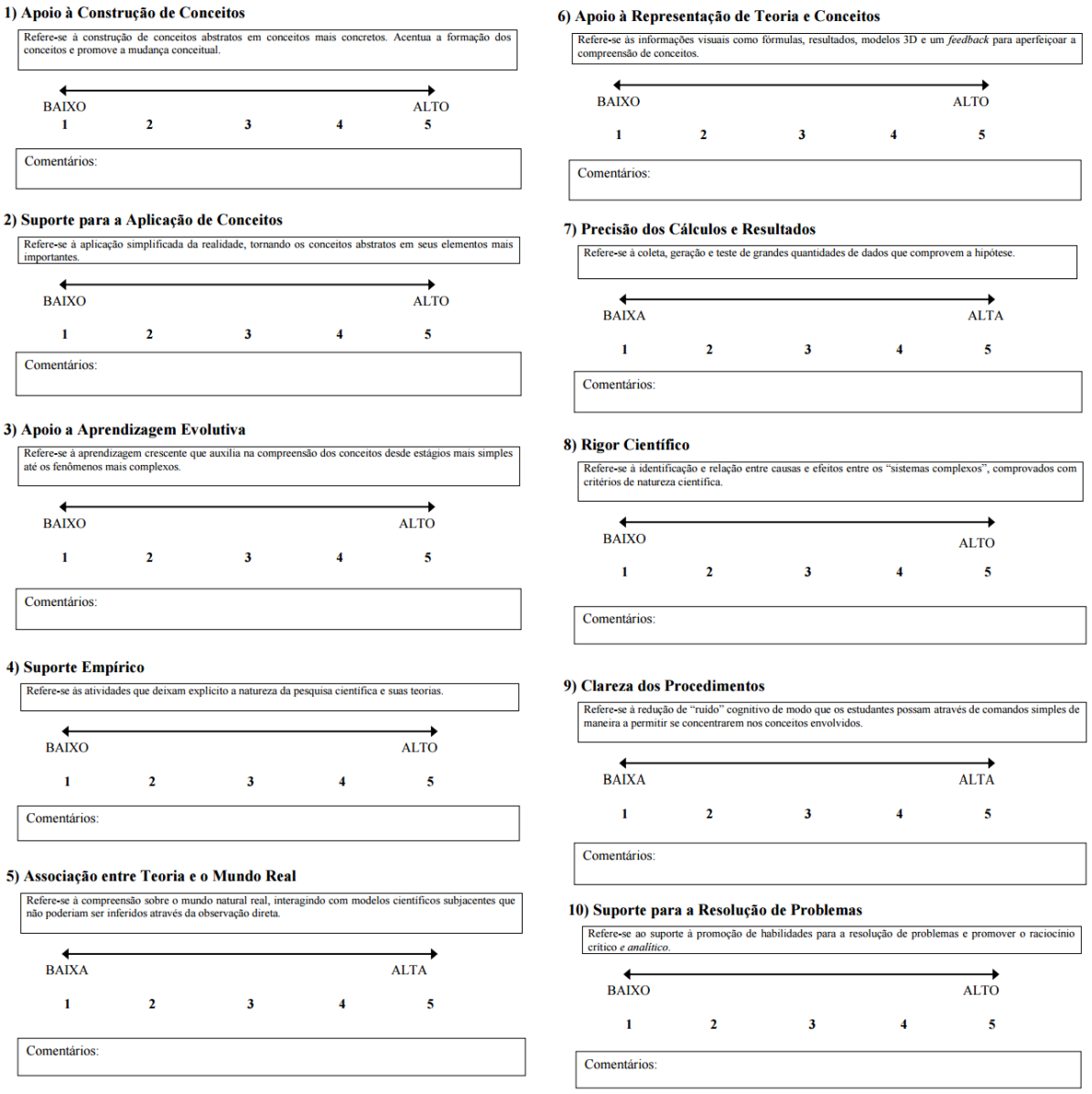


Figura 60: Questionário Aspectos Ensino de Ciências: Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

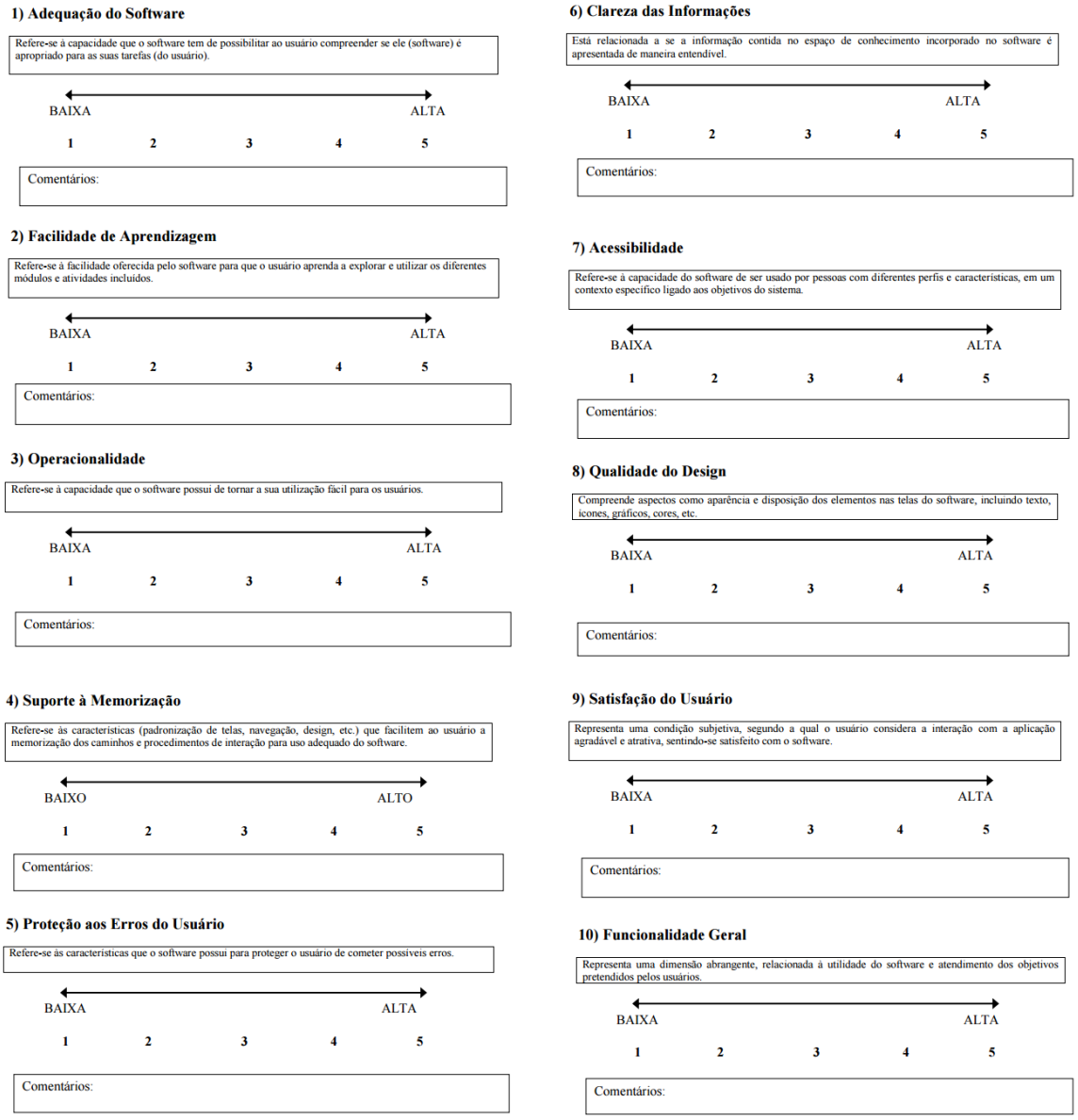


Figura 61: Questionário Aspectos Usabilidade: Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

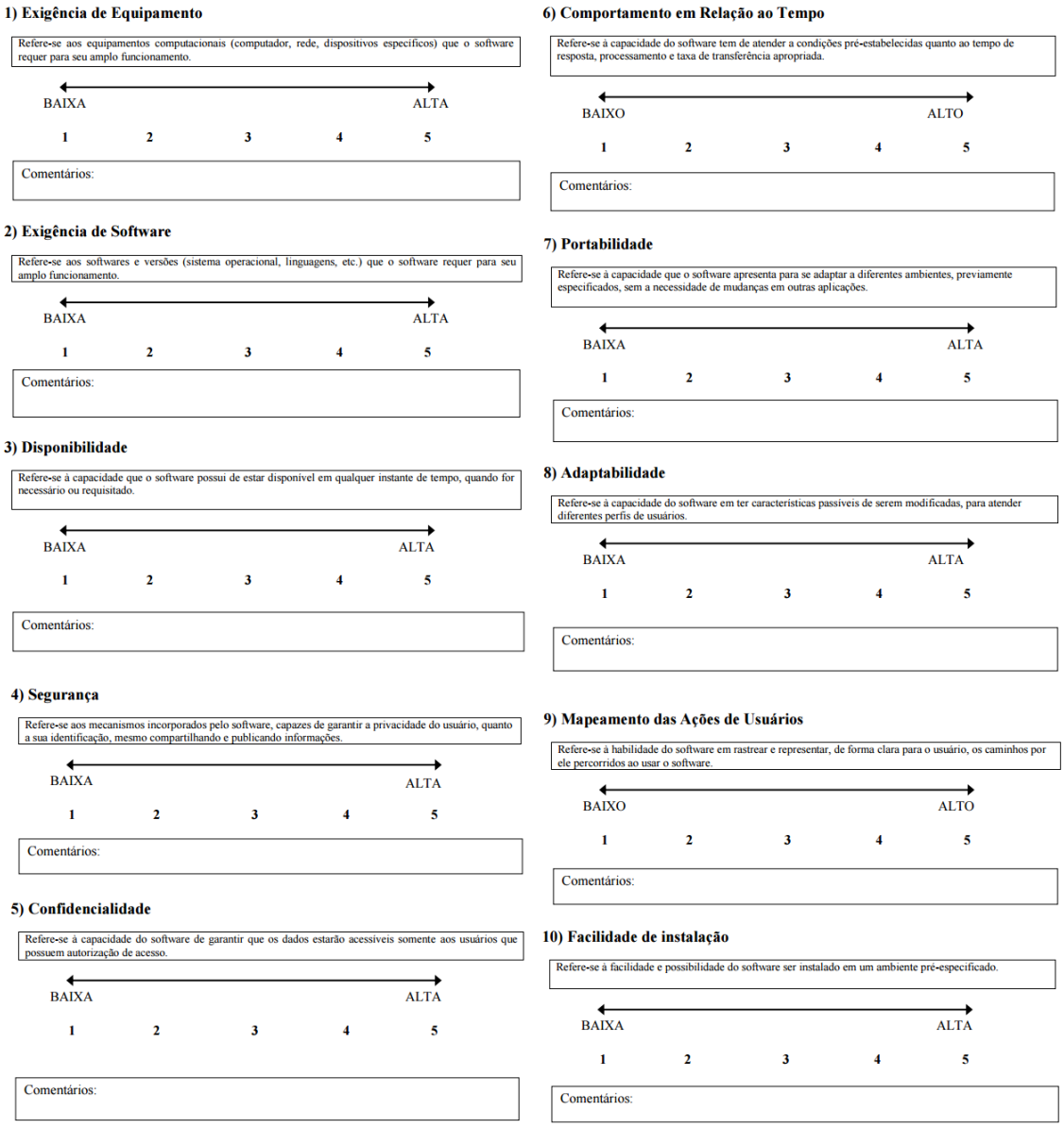


Figura 62: Questionário Aspectos Tecnológicos: Modelo PECTUS. Fonte: (de Souza Rezende, 2013)

D Questionário PROINFO

Aspectos Pedagógicos	Ótimo	MBom	Bom	Regular	Ruim
1-Proporciona um ambiente interativo entre aluno e o *software*?	○	○	○	○	○
2-Permite uma fácil exploração?(seqüencial, não linear)	○	○	○	○	○
3-Apresenta conceitos de forma clara e correta?	○	○	○	○	○
4-Desperta o interesse do aluno, sem perder de vista os objetivos do *software* e do usuário?	○	○	○	○	○
5-Possui vocabulário adequado apresentando texto de qualidade?	○	○	○	○	○
6-Através do fornecimento de feedback permite que o aluno construa seu conhecimento a partir da ação-reflexão-ação?	○	○	○	○	○
7-Apresenta uma visão interdisciplinar?	○	○	○	○	○
8-Atende a diferentes níveis de conhecimento?	○	○	○	○	○
9-Promove o desenvolvimento do raciocínio e descoberta do aluno?	○	○	○	○	○
10-Permite atividades em grupo?	○	○	○	○	○

Aspectos Técnicos	Ótimo	MBom	Bom	Regula	Ruim
1- Os aspectos técnicos especificados no software são compatíveis com a configuração dos equipamentos da escola?	○	○	○	○	○
2- O software é de fácil instalação e desinstalação?	○	○	○	○	○
3- Permite a utilização em rede?	○	○	○	○	○
4- Disponibiliza ajuda de forma eficiente?	○	○	○	○	○
5- O software apresenta facilidade de navegação?	○	○	○	○	○
6- Possui clareza nos comandos?	○	○	○	○	○
7- Apresenta encarte com explicações sobre objetivos, conteúdos, equipe de desenvolvimento e sugestões metodológicas para a utilização?	○	○	○	○	○
8- O software tem como idioma o português?	○	○	○	○	○
9- Apresenta processamento eficaz , sem travamentos?	○	○	○	○	○
10- Possui recursos de multimídia (imagem e/ou som)?	○	○	○	○	○

E Questionário TUP

Pedagogical aspects Subgroup	Question	Type of answer
Context		
Context	Is the subject of learning presented in the authentic context?	
	The usage of the tool fits into the learning setting (e.g. class versus individual learning)	E
	The tool conveniently complements other class activities.	E
	The content fits into the institutional curriculum.	E
Roles	Does the system distinguish between the roles of the users in the learning process?	
Personalization	Is the environment self-adaptable in order to reflect learners' growth?	
Customization	Does the system enable the customization of the perspectives of learning objects according to the learners' needs?	
Cultural diversity	Cultural issues are addressed and properly handled by the environment.	E
Credibility	Are the resources and references used in the tool credible?	
	Is the information presented by the tool up to date?	
Trust	Do you feel the tool maintains trust in it?	
	Learners' ownership of learning is maintained by the environment.	E
Task		
Motivation	Are the tasks designed to maintain (or increase) learners' motivation?	
Goals	Is the selection of the learning goals appropriate?	
	Is it possible to modify the learning goals?	
Task sequence	Does tool enable multiple paths to achieve the learning goals?	
	Are the task sequences adaptable according to the learners' growth?	
Task abstraction	The level of abstraction corresponds with the learners' growth.	E
Real world match	Learning tasks correspond with the real world tasks?	E
Knowledge representation match	Does the system provide enough means for the proper knowledge representation?	
	Does the environment provide multiple representation of knowledge?	

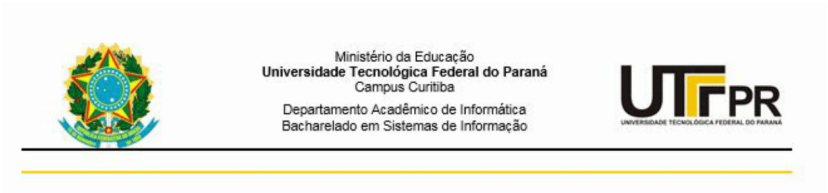
Pedagogical aspects Subgroup	Question	Type of answer
Tools		
Materials management	Can the teacher prepare and modify learning materials using the tool?	
	Can the teachers share learning materials using the tool?	
	Is it possible for users to prepare a presentation using this environment?	
Process management	Does the tool facilitate the evaluation of the learning process?	
	Is it possible to create tests, examinations or assignments using the tool?	
	Does the tool provide process management to be secure for all the parties?	
	The process management follows organizational requirements.	E
Learning styles	Does the tool offer self-directed learning?	
	Does the tool enable different learning styles (e.g learning by exploration, learning by doing etc.)?	
	Does the tool facilitate groupwork?	
	Is it possible to create notes or bookmarks within the tool?	
	The tool supports the creation and sharing of learning artifacts?	E
	Does the tool support off-computer learning activities? (e.g. motivates discussion)	
Interfaces		
Layout	The layout of the interface supports attaining the learning objectives.	E
	The interface layout contains all of the information necessary to achieve learning goals.	E
Tailorization	The interface of the tool is designed according to the target learners' needs.	E
	Is it possible to tailor the interface according to the individual learners' needs?	
	While interacting with learners, does the tool accept also alternative responses?	
Overall pedagogy	Do you attain your learning objectives with the tool?	
	Would you recommend the tool for learning purposes?	A
	What do you think about the educational capabilities of the tool?	A

Technological aspects Subgroup	Questions	Type of answer
Availability and Compatibility		
Software Compatibility	Does the system support the import/export of external resources?	
	Is it possible to use the tool on different operating systems?	
	Does the tool need additional software or components to be installed?	
	Does the tool satisfy the requirements on an free open source software?	
Hardware Compatibility	Does the system need some additional equipment to function properly?	
	Do you need to use unusual external interfaces, devices or media?	
Availability	Is the required equipment readily available?	
	It is easy to set up the environments necessary for the tool.	E
	It is easy to install the tool (e.g. by an installation wizard).	E
Accessibility		
Support for disabled	Is the system adaptable to the needs of disabled people?	
Support for the age groups	Does the system distinguish between the different age groups of users?	
Localization	Does the system follow the regional setting, e.g. format of numbers, time format, keyboard?	
Multilanguage support	Is it possible to change the language of the environment?	B
Organizational aspects		
Maintenance	Is it easy to maintain the system?	
	The system requires much maintenance.	E
Administration	The system is easy to administrate.	E
Training	Is it necessary to train personnel in order to use the tool?	
Finance	Is it expensive to purchase and run the system?	
Integration	Does the system fit into the organization's technological structure?	
Reliability		
Privacy	Is the privacy for users guaranteed?	
Security	Are the security measures adequate for the system?	
Safety	Did you experience any health risks while using the tool?	
Fault tolerance	If any fault occurs, does it cause the breakdown of the system?	
Fault prevention	Does the system actively prevent the faults?	
Defects	The system is free of technological defects leading to malfunctions.	
	The system offers diagnostics tools in order to find possible hardware or software defects.	
General technological notes		
	Did you notice any technological problems causing your learning objectives not to be attained?	A

Usability aspects Subgroup	Question	Type of answer
Learnability		
	Users can rapidly start working with the tool without a long period of training.	E
	Does the usage of the tool impose heavy cognitive load to the users causing problems?	
	Is it possible to select advanced modes of control according to the users' experience?	
Interaction		
Modes of interaction	Does the tool offer various interaction modes (e.g. sound, haptic channel) ?	
	The level of interaction with the tool corresponds to the users' characteristics.	E
	Are the means of interaction properly selected?	
Reversibility	Do the users have the ability to 'undo' and 'redo' their actions?	
Response time	Is the feedback offered by the tool within a reasonable time?	
Shortcuts	Does the system offer shortcuts for the most often invoked commands or sequences?	
Help	Does the system offer any help facility?	
	Is the help easily accessible?	
Navigation	Are the users always properly informed about their position in the system structure?	
	Navigation within the environment is easy and natural for users (e.g. the structure of menu).	E
	Are users facilitated to search within the environment?	
Memorability	Does the tool require re-training of usage after breaks?	
	Does the environment require users to memorize facts unrelated to the learning objectives?	
Aesthetics	The design of the interface is aesthetic.	E
Visual aspects		
Text	Is text in the system clearly legible?	
Graphics	The tool uses graphics appropriately.	E
Organization	The displayed information is properly organized.	E
Audio	The tool uses sound appropriately.	E
Overall usability		
Consistency	Is the environment consistent in terms of design, navigation and terminology?	
	The designer's model of system corresponds with the users' view, expectations and perception of it.	E
Localization	The system complies with the local setting and habits.	E
	Is it possible to use the system by multiple nationalities without restrictions?	
Support for age groups	Is it possible to use the system by various age groups?	
General comments on usability	Which problems have you noticed while working with the tool?	A

Type of answer	Values	Denotation
Open ended	Free form text	A
Boolean	YES / NO	B
Boolean tick	Box ✔ / Cross ✘	B
Short numerical[2]	5..1	C1
Short numerical + zero[2]	5..0	C2
Long numerical[2]	10..1	D1
Attitude	Strong confirmation .. Strong disagreement	E

F Questionário Aplicado aos Instrutores



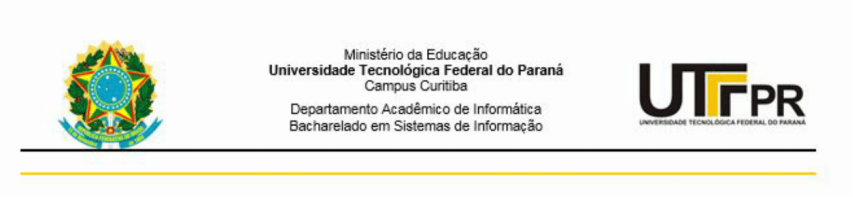
Questionário Professores
Oficina de Python
06/07/2015

Professor(a):_____

Avaliação dos Estudantes:

	Estudante (1)	Estudante (2)	Estudante (3)	Estudante (4)
Nome				
Nota				
Nível de programação do estudante				
Precisaria repetir a oficina de Python?				
Principais dificuldades/pontos a serem trabalhados				
Principais facilidades/pontos positivos				

G Questionário Sobre Visões Metodológicas

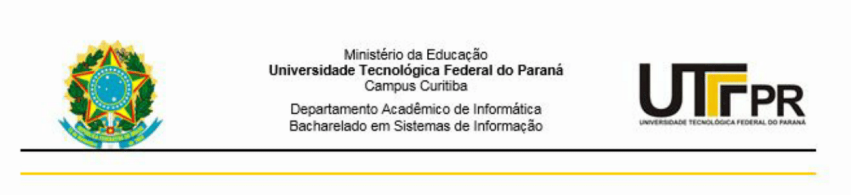


Questionário Visão Metodológica do Ensino
Oficina de Python

Nome:_____
__

1. Qual período/ano você ministrou a oficina de Python?
2. Você se baseou em alguma corrente filosófica de ensino? (Ex: sócio-cultural, construtivista, behaviorista...)
3. De maneira geral, a oficina se baseava em atividades com roteiro e passos fixos/pré-definidos para se alcançar um objetivo ou atividades que permitiam maior liberdade para este escolher o quê ou como desenvolver durante a aula?
4. Durante as aulas, como você avalia a motivação dos estudantes?
5. Houveram desistências? Se sim, o que você avalia ser o motivo?
6. Havia colaboração entre os estudantes durante a execução das tarefas?
7. Era feito uso de algum material educativo para apoio às aulas? Quais?
8. Era incentivado o desenvolvimento de habilidades sociais durante a aplicação da oficina? Se sim, como?
9. Foram realizadas atividades ou exercícios em grupo?

H Questionário Aplicado aos Estudantes



Questionário Estudante

Oficina de Python

06/07/2015

Estudante (Opcional):_____

1) Você já teve contato ou interesse em linguagens de programação antes da oficina de Python?
   a. Sim
   b. Não

Se sim, cite onde ou como foi este contato.
_____
_____
_____

2) Depois da oficina de Python, seu interesse na área de computação:
   a. Aumentou
   b. Se manteve o mesmo
   c. Diminuiu

3) Enumere (de 1 a 10) as aulas da oficina de acordo com sua satisfação, em que 1 significa a que você mais gostou e 10 a que você menos gostou (não considerar as aulas que você não compareceu):

____ - Mestre Mandou.
____ - Mapa de distâncias.
____ - Que dia você nasceu?
____ - Jokenpô.
____ - Super Trunfo.
____ - Advinha que numero estou pensando.
____ - Código Morse.
____ - Jokenpô (Spock - Lagarto).
____ - Jogo da Velha.
____ - Contando Histórias com Animações.

4) De maneira geral, você prefere que os professores tragam atividades com roteiro e passos fixos/pré-definidos para se alcançar um objetivo ou atividades que permitam maior liberdade para você escolher o quê ou como desenvolver durante a aula? Porquê?

_____
_____
_____
_____

5) De maneira geral, você prefere realizar atividades de maneira individual ou em dupla/grupo? Porquê?

_____
_____
_____
_____

6) Você gostaria de ter mais aulas como a de hoje? Explique o porquê.

_____
_____
_____
_____