Capítulo 4

Sobre cursos introdutórios de programação na modalidade MOOC utilizando Moodle

Leônidas de Oliveira Brandão e Romenig da Silva Ribeiro

Abstract

The Massive Open Online Course (MOOC) is a new step towards the knowledge democratisation. On one hand it is a result of the demand for (in)formation, on the other hand it is also a consequence for the ICT evolution that allows better interactions between users. Considering the introduction to programming there is a recent paradigm, the Visual Programming (VP), which has been used to reduce the learner's failure in this topic. In this text we present a set of systems to manage MOOC of programming using Moodle and the VP, but also programming under traditional textual language, as C. The supporting systems are the Moodle plugins iAssign and VPL, the first to allow the integration of the VP system iVProg, and the second for textual programming. Both options allow automatic evaluation for exercices.

Resumo

Neste trabalho discutimos dois tópicos relevantes à computação, como introduzir conceitos de programação e como gerenciar um curso Web aberto e massivo (MOOC). A popularidade dos MOOC resulta de uma demanda por (in)formação, mas também é resultado do avanços das Tecnologias de Informação e Comunicação (TIC). Quando se considera o ensino introdutório de programação, uma novidade é o uso do paradigma de Programação Visual (PV) que tem apresentado bons resultados para reduzir a taxa de falha nesta disciplina. Neste texto apresentamos um conjunto de sistemas para gerenciar cursos do tipo MOOC de introdução à programação, usando a PV e também a programação textual tradicional, com a linguagem C. Isso será feito com o sistema Moodle a partir dos pacotes iTarefa e VPL, respectivamente.

1. Introdução

A atividade de programação de computadores é um dos pilares para os grandes avanços tecnológicos observados nos últimos anos. Esse avanço tem implicado em aumento da demanda por profissionais de Tecnologia da Informação (TI), como se observa a partir de um relatório de 2014 da Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação (BRASSCOM). Esse relatório aponta que o mercado de *Tecnologia da Informação e Comunicação* (TIC) no Brasil foi o sexto maior do mundo em 2010, o quinto em 2011 e o quarto em 2012, movimentando mais de 169 bilhões de dólares em 2012[15].

Desse modo, as iniciativas para introduzir conceitos de algoritmos e programação para jovens aprendizes têm sido recomendadas desde a década de 1960 por Seymour Papert. Van Rossum (1999), por exemplo, propõe um projeto visando alunos do secundário e de graduação em áreas que não ciência da computação. Já em 2011, o governo britânico ressalta a importância de melhorar a capacitação de professores para que estes possam melhor ensinar tecnologias da informação e comunicação [OFSTED, 2011].

Entretanto, o aprendizado de programação em cursos de graduação, usualmente utilizando linguagens de programação tradicionais como *C* ou, mais recentemente, *Java*, apresenta um quadro de dificuldades que resultam em evasão. Isso tem motivado uma grande quantidade de pesquisas. Por exemplo, Robins *et al*. (2003), analisa os aspectos educacionais e psicológicos envolvidos no processo de aprendizagem de programação de computadores. Para isso, revisaram a literatura desde a década de 1970 buscando determinar os fatores inerentes ao processo de evolução que permita a um "programador iniciante" se tornar um "programador experiente". Os autores detectaram, dentre outras problemas, dificuldades na resolução de problemas, na compreensão de conceitos abstratos (e.g., variáveis e ponteiros), na utilização de recursos específicos de algumas linguagens de programação.

Em geral, aprender a resolver problemas computacionalmente envolve a aprendizagem de novas ferramentas, uma nova forma de pensar (pensamento sistemático) e uma nova forma de escrever a solução para o problema. Por outro lado, do ponto de vista didático, o aprendizado de programação pode ajudar o aprendiz a desenvolver sua capacidade de resolver problemas de modo mais sistemático, além da associação direta com vários conceitos matemáticos, tais como, soma e divisão, além de outros algoritmos (e.g., de Euclides), que de modo geral estão relacionados com o processo de indução matemática.

Uma possibilidade de redução de dificuldades é o emprego do paradigma de Programação Visual (PV). Para desenvolver algoritmos usando a PV, o aprendiz não precisa memorizar a sintaxe de uma linguagem de programação pois os sistemas de PV apresentam os comandos sob a forma de ícones. Além disso, se o ambiente de PV estiver integrado ao sistema gerenciador de cursos, a carga de trabalho adicional é reduzida, possibilitando ao aprendiz focar sua atenção no aprendizado de resolução de problemas utilizando-se de algoritmos.

15 URL: <http://www.brasscom.org.br/brasscom/Portugues/download.php?cod=593>, página 22.

Um exemplo de experimento com resultado positivo da PV é relatado por exemplo por Brandão et al. (2012), que apresentam um experimento em que dois grupos, com 3 aprendizes cada, deveriam resolver 2 problemas utilizando tanto a PV quanto a programação textual (em *C*), um dos grupos iniciando a tarefa com PV e o outro com C. No grupo que começou com a PV, todos conseguiram resolver os 2 problemas tanto na PV quanto em *C*, embora gastando em média 3 vezes mais tempo com a linguagem C. Já o grupo que iniciou com *C*, apenas 1 participante conseguiu resolver os 2 problemas em C e com a PV, mas gastou mais que o dobro do tempo do que os participantes do outro grupo. O fato dos alunos do primeiro grupo terem obtido melhores resultados e terem conseguido resolver os problemas usando também a linguagem *C*, indica que a PV pode também ser um bom mecanismo para reduzir as dificuldades com linguagens tradicionais (transferência).

Retomando o problema do déficit de pessoas aptas a participarem do mercado de desenvolvimento de sistemas, a Educação a Distância (EAD) apresenta-se como opção para aqueles que desejam complementar sua formação ou para quem não dispõe de um período livre para estudo.

Nesse cenário surgiu recentemente uma outra possibilidade EAD, que são os cursos Web abertos e massivos, em Inglês, sendo denominado por *Massive Open Online Course* (*MOOC*). Mas como esses cursos geralmente não oferecem certificados e a inscrição é voluntária, o índice de desistência é alto. A literatura também aponta outros fatores como a impulsividade, pessoas que se inscrevem provavelmente por alguma curiosidade e que posteriormente sequer entram uma vez no ambiente de curso. Um exemplo indicativo desse fenômeno é observado no trabalho de Seaton *et al*. (2014) que analisou um *MOOC* sobre circuitos e componentes eletrônicos promovido pelo MIT. Os autores constataram que a taxa de estudantes que nunca acessaram o sistema do curso era de 29%, dos 154000 inscritos, 46000 não entraram sequer uma vez no ambiente do curso[16].

Nesse contexto, este texto visa contribuir com o processo de aprendizagem de programação, apresentando um conjunto de ferramentas computacionais que possam ser utilizadas para promover cursos introdutórios de programação apoiados pela *World Wide Web* (*Web*) ou, mais especificamente, em cursos Web abertos e massivos (*MOOCs*).

As ferramentas aqui apresentadas, em geral reduzem a carga de novidades para os aprendizes pois o ambiente de programação está integrado ao *Sistema Gerenciador de Cursos* (*SGC*), no caso o sistema *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*). Além disso, elas dispõem de recurso de avaliação automática dos exercícios.

Quanto ao modelo de programação, será utilizado tanto a PV quanto a programação tradicional, escolhendo *C* como linguagem textual.

Na subseção seguinte, será apresentada a problemática que se pretende tratar no texto.

16 "The novelty and publicity surrounding MOOCs in early 2012 attracted a large number of registrants who were more curious than serious. We still take participation in assessment as an indication of serious intent. Of the 154,000 registrants in 6.002x in spring 2012, 46,000 never accessed the course, and the median time spent by all remaining participants was only one hour" [Seaton et al., 2014, pp 4].

1.1 Problemática

O problema central considerado neste trabalho diz respeito à ampliação do público atendido por cursos de introdução à programação a partir da *Web*. Isso passa por possibilitar o emprego de ferramentas que automatizem a avaliação de tarefas de programação e que permitam a autoria de maneira simplificada. Assim a questão da massificação do aprendizado de programação passa naturalmente pelo oferecimento de cursos na modalidade *MOOC*. Adicionalmente, por se observar uma alta taxa de evasão nestes cursos, uma discussão a esse respeito também se impõe.

Assim, a problemática envolvida neste trabalho é bastante ampla, sendo necessário o foco em alguns aspectos particulares. Aqui optou-se por abordar a questão da carga de trabalho necessária ao aprendizado inicial de programação e os modelos que permitam avaliação automática de exercícios.

Em relação à redução da carga de trabalho uma das discussões relevantes é a redução de algumas das barreiras envolvidas no aprendizado de programação, como os problemas de sintaxe, tais como identificados por Robins *et al*. (2006).

1.2 Justificativa

Os tópicos aqui considerados se justificam pela importância do tema: aprendizagem de algoritmos e programação, seja do ponto de vista de mercado, seja do ponto de vista didático. Isso pode ser observado a partir das iniciativas em alguns países visando introduzir esses conceitos desde a educação básica [NSTC, 2013; DE, 2013].

De outra parte existe a dificuldade encontrada pelos aprendizes em cursos de graduação com a disciplina de introdução à computação, apontando necessidade de melhor entender essa questão. Nesse sentido o emprego do paradigma de PV tem apresentado bons resultados, o que nos motiva a estudar e compreender seu papel nos vários cenários de aprendizado de programação.

Por fim, os cursos do tipo *MOOC* possibilitam um interessante modo de difusão do conhecimento, podendo, desta forma, ser empregado para promover o aprendizado de computação.

1.3 Objetivos

O objetivo aqui é contextualizar o ensino/aprendizagem dos primeiros conceitos de programação de computadores e o emprego de *MOOC* nessas atividades. Como objetivos específicos buscamos apresentar um conjunto de ferramentas computacionais para a construção e gerenciamento de cursos de programação em ambientes *Web* focando no modelo dos *MOOC*. Em particular, destacamos o uso de PV para reduzir algumas dificuldades frequentemente encontradas por estudantes no processo de aprendizagem de algoritmos e programação.

2. Fundamentação

Nesta seção apresentamos os conceitos associados a este trabalho, expondo de forma resumida o contexto do aprendizado de programação o modelo de educação a distância em geral e os *MOOC* em particular, além de se apresentar o modelo de programação visual para o desenvolvimento de algoritmos.

2.1 Contexto da aprendizagem de programação: sobre sua importância

Em um discurso à nação em 2011 a presidente brasileira Dilma Rousseff declarou que "Estou aqui para reafirmar o meu compromisso com a melhoria da educação", para logo depois dizer que "Nenhuma área pode unir melhor a sociedade que a Educação. Nenhuma ferramenta é mais decisiva do que ela para superarmos a pobreza e a miséria. Nenhum espaço pode realizar melhor o presente e projetar com mais esperança o futuro do que uma sala de aula bem equipada, onde professores possam ensinar bem, e alunos possam aprender cada vez melhor"[17].

Já o governo norte-americano tem destacado também o papel da matemática. No início de segundo mandato do presidente Barack Obama, em 2013, foi lançado o plano quinquenal da nação para a educação nas áreas de ciência, tecnologia, engenharia e matemática (em Inglês *Science*, *Technology*, *Engineering*, *and Mathematics* - *STEM*) [NSTC, 2013]. Esse documento inicia-se da seguinte forma: "Avanços na Ciência, Tecnologia, Engenharia e Matemática (STEM) têm há muito tempo um papel central na habilidade da Nação para a fabricação de produtos melhores e de modo mais inteligente, para a melhorar a saúde pública e fazer crescer a economia" [NSTC, 2013, pp 13].

Uma vez que os sistemas computacionais se tornam mais importantes para o desenvolvimento da ciência, tecnologia e indústria, o raciocínio lógico e a programação de computadores tornam-se habilidades fundamentais e sua introdução nos estágios iniciais da educação tem sido progressivamente mais adotado. Essas ações são importantes para o desenvolvimento da nação no sentido de atrair mais profissionais para a área de Ciência da Computação.

O Brasil está entre os maiores mercados de TIC do mundo, ocupando em 2012 o sétimo lugar, ao movimentar cerca de US$ 212 bilhões em 2011 – crescimento de 13% em relação à 2010 [BRASSCOM, 2012]. Além disso, segundo o relatório da BRASSCOM (Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação) "apesar do aumento no número de matrículas e de formandos nas áreas de ciências e engenharia, há receio de uma futura escassez de profissionais de TI qualificados para atender a demanda" [BRASSCOM, 2012, pp. 20]. Dessa forma, percebemos que o mercado de trabalho é receptivo para novos profissionais.

O que se busca evidenciar aqui é a importância das ciências para o desenvolvimento de uma nação e, em particular, a programação como componente curricular para colaborar no desenvolvimento dessas ciências. Entretanto, o processo de aprendizagem de programação pode ser árduo e moroso devido à quantidade de objetos de estudos com os quais os aprendizes precisam lidar em concomitância, além da dificuldade intrínseca de se resolver um problema por meio de um algoritmo[18].

Dada a grande valorização do raciocínio lógico e ao conhecimento de programação de computadores, o investimento em iniciativas para atrair novos

17 http://www2.planalto.gov.br/acompanhe-o-planalto/discursos/discursos-da-presidenta/pronunciamento-a-nacao-da-presidenta-da-republica-dilma-rousseff-em-cadeia-nacional-de-radio-e-tv (discurso de 10/02/2011).

18 Em geral um algoritmo que resolve uma determinada família de problemas, caracterizada pela seu conjunto de possíveis entradas de dados.

profissionais para a área e a ampliação da capacidade de formação dessas pessoas é primordial.

2.2 Contexto da aprendizagem de programação: sobre as dificuldades

Tradicionalmente, os cursos iniciais em programação possuem como componentes curriculares a introdução à lógica de programação, o desenvolvimento de algoritmos utilizando essa lógica e a aprendizagem de ambientes e linguagens para representar, construir e executar os algoritmos desenvolvidos. A experiência didática dos autores e a literatura da área indica que tantas novidades podem representar um excesso de informação no processo inicial da aprendizagem, podendo atrapalhar o desempenho dos aprendizes.

Um indicador da existência de dificuldades é a elevada taxa de evasão desses cursos. No Brasil, a taxa média de evasão das áreas de Ciências, Computação e Matemática foi de 28% entre os anos 2000 e 2005 [LOBO *et al*., 2007]. Dentre os motivos para o abandono, cita-se a dificuldade, frequentemente inesperada, que é sentida pelos aprendizes.

Entretanto, deve-se destacar um dos fatores relatados na literatura: a carga de trabalho adicional oriunda da aprendizagem de detalhes de sintaxe e de um novo ambiente (o compilador). Nesse sentido a PV se apresenta como boa ferramenta por naturalmente reduzir esse fator.

Como relatado por Lahtinen *et al*. (2005), que a partir da aplicação de um questionário a alunos e professores, constataram uma correlação forte entre dificuldades enfrentadas na aprendizagem das estruturas de programação (como laços e seleção), da sintaxe de linguagens de programação textuais, na compreensão de como construir um algoritmo para resolver um problema e na divisão de uma funcionalidade em pequenas funções.

Ao empregar outra metodologia, Robins *et al*. (2006) detectaram problemas semelhantes. Por um período de dois anos os autores fizeram um levantamento de disciplinas introdutórias de programação, com a utilização da linguagem *Java*, examinando os tipos de dúvidas dos aprendizes. Eles constataram que a grande maioria das dúvidas estavam relacionadas a pequenos problemas de sintaxe ou à ausência de boas práticas de programação.

Mas existem outros fatores determinantes na taxa de abandono, por exemplo, Beaubouef e Mason (2005) citam vários, dentre os quais: a assessoria ruim antes e durante a graduação (podendo levar o estudante a uma conclusão errada a respeito da área), a falta de habilidades em Matemática e em resolução de problemas, os cursos mal planejados, a falta de treino e de retroação por parte dos tutores, dentre outros.

Ao encontrarem muitas dificuldades nos cursos introdutórios de programação, os aprendizes ficam desmotivados, o que prejudica a aprendizagem. A consequência direta desse fato é a deficiência no aprendizado, o que acarreta em reprovação e gera altas taxas de evasão [Kinnunen e Malmi 2006].

Nesse sentido observa-se que um ambiente de PV pode reduzir as dificuldades apontadas por Lahtinen *et al*. (2005), Robins *et al*. (2006) e Beaubouef e Mason (2005). Apesar de um sistema de PV apresentar uma sintaxe associada, o seu modelo visual serve de diretriz para o aprendiz. Adicionalmente, se o ambiente de PV estiver integrado ao

gerenciador de curso e se ainda dispuser de recurso de avaliação automática, a qualidade de retroação fornecida ao aprendiz será maior, podendo melhorar sua compreensão sobre algoritmos.

2.3 EAD e MOOC

A Educação a Distância (EAD) se diferencia da educação presencial tanto no aspecto de tempo quanto de espaço. Na EAD "instrutores" e aprendizes estão na maior parte do tempo em espaços distintos e não existe a obrigação de sincronização de tempo para as atividades de todos os aprendizes.

Esta modalidade educacional existe a bastante tempo, sendo um de seus marcos um anúncio de curso de taquigrafia por correspondência publicado na Gazeta de Boston (*Boston Gazette*) por Caleb Philipps [Neto, 1998; Alves, 2011].

A EAD possibilita ao aprendiz uma maior flexibilidade para estudo, deste modo ela "vêm cumprir o importante papel, entre outras atribuições, de proporcionar acesso ao ensino superior para a população que se encontra em localidades mais remotas e de difícil acesso aos grandes centros urbanos" [Alvarez, 2013, pp 11].

No Brasil podemos destacar dois marcos da EAD, a Lei de Diretrizes e Bases da Educação Nacional n° 9.394, de 20 de dezembro de 1996, e a criação da Universidade Aberta do Brasil (UAB) em 2005.

A partir daí o número de alunos inscritos em cursos de graduação na modalidade EAD tem crescido, atingindo em 2012 mais de 1 milhão de alunos no Brasil, como indicam dados publicados pelo Instituto Nacional de Estudos e Pesquisas (INEP), mostrados na tabela 1.

Tabela 1 – Número de alunos de graduação matriculados em cursos presenciais e a distância

EAD	838.125	930.179	992.927	1.113.850
Presencial	5.115.896	5.449.120	5.746.762	5.923.838
Ano	2009	2010	2011	2012

Fonte: MEC/INEP [INEP-1, 2014, pp. 61]

Adicionalmente, o crescimento da EAD pode ter se beneficiado da implantação dos mecanismos de avaliação de desempenho dos estudantes, em 1995 o Exame Nacional de Cursos (Provão) e posteriormente o Exame Nacional de Desempenho dos Estudantes (Enade), que substituiu o Provão a partir de 2004. O Provão foi criado pela Lei no. 9.131, de 24 de novembro de 1995, e o Enade, foi regulamentado pela lei 10.861 de 2004 [Paiva, 2008].

A partir do Enade pode-se observar que o desempenho de estudantes formados em cursos EAD ou presencial é semelhante. Por exemplo, um relatório do INEP de 2013, com notas do Enade dos 2006 e 2008, permite comparar o desempenho de alunos nas duas modalidades, em 9 cursos distintos, percebendo-se que os desempenhos são bastante próximos. A tabela 2 apresenta estes dados, extraídos das páginas 70 e 71 de [INEP-2, 2009].

Tabela 1.2 – Desempenho alunos de graduação presencial e EAD 2006 e 2008

Cursos	2006				2008			
	Ingressantes		Concluintes		Ingressantes		Concluintes	
	Pres.	EAD	Pres.	EAD	Pres.	EAD	Pres.	EAD
Biologia	30,4	32,8	32,7	32,8	35,3	31,2	40,8	31,9
Ciências Sociais	38,4	52,9	41,2	52,9	35	37,2	41,7	62,3
Filosofia	29,8	30,4	32,5	30,4	31,4	34,4	34,4	33,5
Física	30,6	39,6	32,5	39,6	29,7	33	37,8	37,2
Geografia	36,8	32,6	39	32,6	36	35,6	38,8	33,9
História	36,5	31,6	38,5	31,6	36,6	33,3	40,2	29,6
Letras	34	33	35,7	33,1	41,5	37,6	46,1	35,3
Matemática	28,8	34	31,7	34,2	31	31,2	35,2	28,9
Pedagogia	39,9	46,0	43,4	46,1	42,2	41	49,5	46,2
Média	34,6	37,0	36,8	37,0	35,5	37,0	41,0	38,3

Fonte: MEC/INEP [INEP-2, 2009]

O desempenho semelhante entre formados pela EAD e pelos cursos presenciais, como ilustrado na tabela 2, serviu como uma validação do modelo de EAD como alternativa real para a formação de muitos. Sob o ponto de vista tecnológico, percebemos que a relevância adquirida pela EAD está também ligada à evolução das Tecnologias da Informação e Comunicação (TIC). Enquanto nos primeiros modelos de cursos EAD as cartas eram o suporte, com interações mínimas entre "instrutores" e aprendizes, hoje as TIC possibilitam interações rápidas, inclusive entre os aprendizes. Em particularmente, isso se deve aos recursos ligados à *Web*, como navegadores *Web* e Sistema Gerenciadores de Cursos (SGC).

Dentre os recursos essenciais aos ambientes EAD atuais, devem ser destacados os mecanismos de comunicação (*chat* e fórum), além ferramentas para avaliação automática. E vale registrar, que mesmo estes recursos não foram novidades trazidas pela *Web*, pois o sistema PLATO já propunha seu emprego desde meados da década de 1960 [Bitzer et al., 1965].

Mais recentemente, estes recursos de personalização da aprendizagem e de avaliação automática possibilitaram o surgimento de cursos abertos em larga escala, os *Massive Open Online Courses* (*MOOC*). O *MOOC* aparece com este nome em 2008, com o oferecimento de um curso na Universidade de Manitoba, Canadá. George Siemens e Stephen Downes ofereceram o curso *Connectivism and Connective knowledge* para dois grupos distintos, um pequeno grupo de 24 alunos da universidade e outro bem maior (mais de 2000) que se inscreveram via Internet, originários de vários países [Mackness et al., 2010].

A partir desse evento, a popularização do *MOOC* foi rápida. Em 2011 e 2012 aparecem mais cursos nessa modalidade, atraindo um número muito grande de estudantes, como os cursos da Universidade Stanford e do Instituto Tecnológico de Massachusetts (MIT). Stanford lançou 3 cursos em 2011, surgindo daí o *Coursera*. O MIT lançou a plataforma *MITx* e quando a Universidade de Havard aliou-se à iniciativa livre do MIT, passaram a denominar o projeto por *edX* [Souza, 2015].

As iniciativas de *MOOC* se multiplicaram, existindo hoje várias universidades e empresas oferecendo cursos nessa modalidade e, da mesma forma que o Brasil tem larga fatia de participação em redes sociais, nos *MOOC* isso também ocorre. Regalado (2013) cita dados do *Coursera*, a partir dos quais pode-se perceber que o Brasil aparece como o país externo com o maior número de inscritos em seus cursos, como mostrado na tabela 3.

Tabela 3 – Legião estrangeira. De onde vem os inscritos em cursos universitários *online*[19].

País	**Percentual**
Estados Unidos	38,5%
Brasil	5,9%
Índia	5,2%
China	4,1%
Canadá	4,1%
Reino Unido	4%
Rússia	2,4%
189 outros países	35,8%

Fonte: [Regalado, 2013]

Do mesmo modo, também se multiplicaram o número de artigos analisando os *MOOC*, principalmente sua alta taxa de desistência, na faixa dos 90% [Taneja e Goel, 2014].

Se de um lado a taxa é alta, de outro o número de aprendizes envolvidos pode compensá-la. Assim, um curso com 10 mil participantes, que tenha uma taxa de sucesso de 10%, pode contribuir para a melhoria de formação de 1000 pessoas. Além disso, o mesmo curso pode ser oferecido mais vezes sem a demanda trabalho adicional. Se ainda considerarmos a capacidade de difusão do conhecimento, a contribuição de um *MOOC* é bastante positiva.

2.4 Programação Visual

Os princípios da *Programação Visual* (PV) podem ser identificados na década de 1970, com o projeto *Pict* [Glinert e Tanimoto, 1984]. Seus autores perceberam que muitos se

19 Foreign Legion. Where people signing up for free online college courses come from.

interessavam por produzir sistemas que suprissem suas necessidades, mas que não eram bem-sucedidos na tarefa de programação. Daí pensaram em uma "linguagem de programação" mais simples, com forte apoio em elementos gráficos.

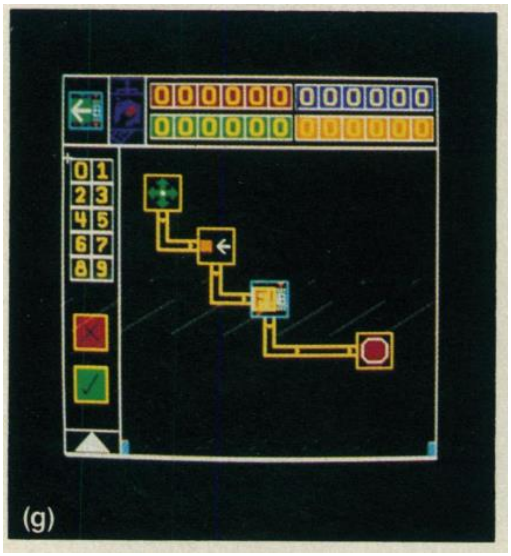

Figura 1. Imagem sobre cálculo de fatorial no *Pict*.

(Fonte: Glinert e Tanimoto, 1984)

Em geral, a PV fornece ao "programador" recursos como ícones, fluxogramas, blocos de código ou representações digitais de objetos do mundo real para projetar seu algoritmo. Uma definição de PV mais abstrata é proposta por Myers (1990) como "qualquer sistema que permite ao usuário representar um programa em duas (ou mais) dimensões". No mesmo parágrafo, Brad Myers explica que "linguagens de programação tradicionais [texto] não são consideradas de duas dimensões porque os compiladores ou interpretadores as processam como uma cadeia de caracteres unidimensional".

Apesar de não terem sido criados com o objetivo de servir, necessariamente, aos processos de ensino e de aprendizagem de programação, atualmente existem vários sistemas de PV com esse fim, como o *Alice*[20] (Figura 2) e o *Scratch*[21] (Figura 3), que foram objetos de estudo de vários artigos, apresentando bons resultados de aprendizado.

Adams (2007), por exemplo, relata o sudo do *Alice* na desconstrução de estereótipos sobre a Ciência da Computação durante acampamentos de 1 semana. A atitude e percepção dos aprendizes pós-acampamento era mais positiva do que no início, tanto para os garotos quanto para as garotas.

Já Dann *et al*. (2012) empregaram o *Alice* servindo como ponte para ensinar *Java*. Os autores analisaram dados de um curso de *Java* sem o *Alice* em 2006 com outras duas edições usando o *Alice*, em 2009 e 2010. Eles relatam resultados estatisticamente significantes em favor do modelo de transferência de conhecimento usando o *Alice* como ponte.

---

20 Sítio do sistema *Alice* <http://www.alice.org>. Último acesso foi realizado em 04/07/2015.

21 Sítio do sistema Scratch <https://scratch.mit.edu/>. Último acesso foi realizado em 04/07/2015.

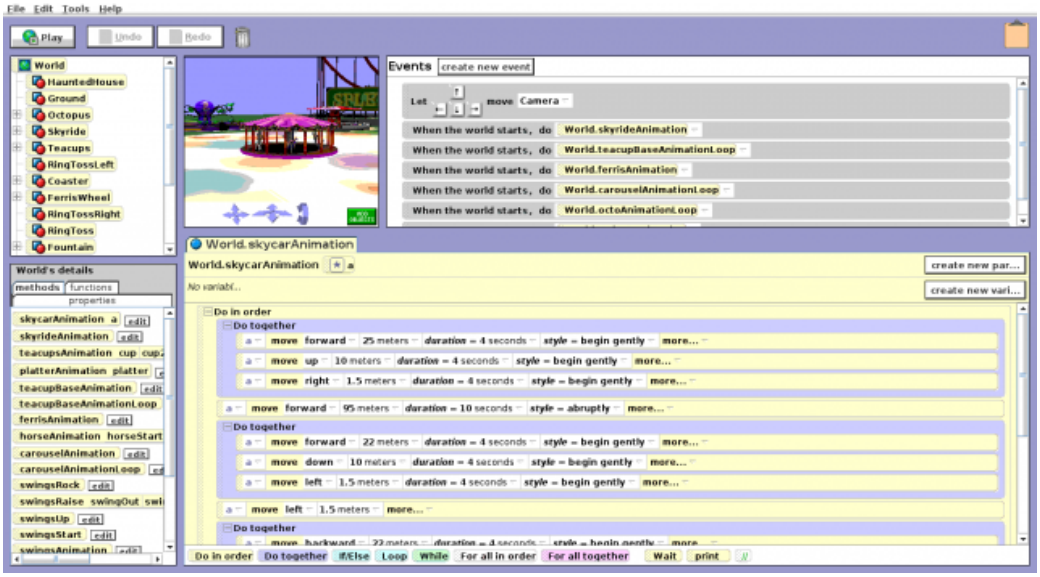

Figura 2. Tela do *Alice* 3 em sua versão para a área de trabalho.

Outro resultado positivo de PV foi detectado por Armoni *et al*. (2015) que examinou 5 diferentes grupos de alunos em Israel em nível escolar equivalente ao primeiro ano do ensino médio brasileiro. Os autores relatam a existência de uma disciplina de programação optativa nessa série e perceberam que as turmas de alunos que tiveram experiência anterior com programação, usando o *Scratch*, tiveram maior facilidade para entendimento dos conceitos de programação usando *Java* ou *C#*. Relataram, também, um maior interesse das turmas com experiência prévia com *Scratch* pelo curso de programação (quando ocorreu o treinamento prévio com *Scratch* foram necessárias 2 classes, contra 1 do ano anterior).

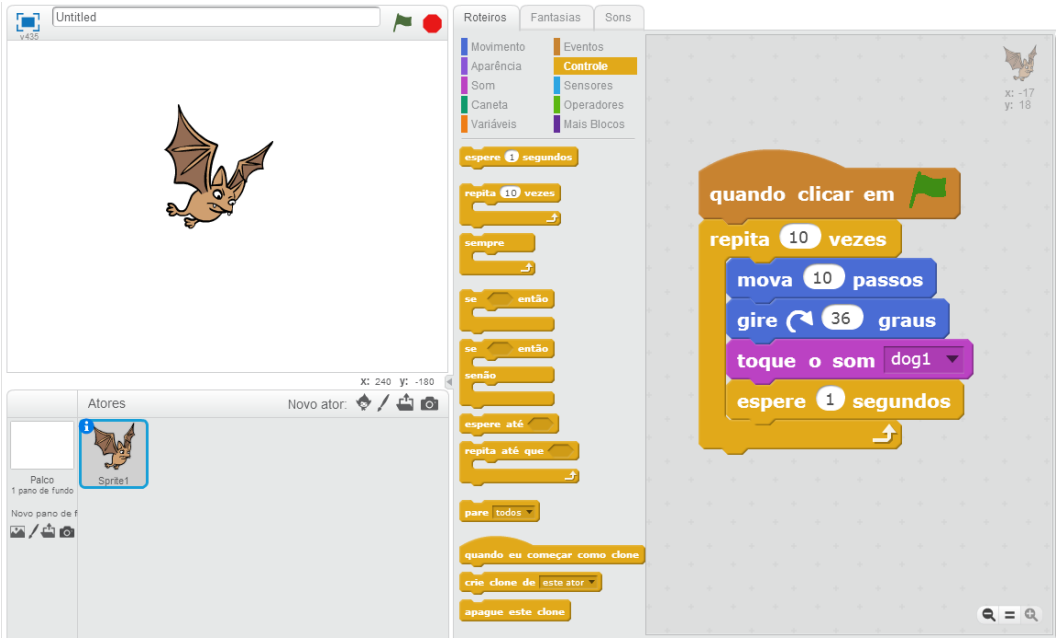

Figura 3. Tela do *Scratch* em navegador; é necessário o *Flash Player*.

Utilizando uma extensão do *Scratch*, o *BYOB*, Stefano (2011) alterou sua camada de apresentação para que os blocos de comandos tivessem a estrutura da linguagem *C* e utilizou esse ambiente como ponte para o aprendizado de *C*. O autor relatou boa

aceitação por parte dos aprendizes, mas citou que o trabalho de analisar o desempenho deles com o aprendizado de *C* ainda estava sob investigação.

Vale também citar dois outros trabalhos que apontam a viabilidade do uso de PV com aprendizes desde o ensino fundamental, Freudenberg, et al.(2009) e Repenning e Ioannidou (2004).

De modo geral, os sistemas de PV são utilizados como instrumentos para motivar e ensinar conceitos básicos de programação para várias faixas etárias. Os relatos na literatura apontam para que a PV aumenta a motivação e o interesse pela área de Computação e ainda reduz as dificuldades no processo de aprendizagem de programação.

Uma vez que a maioria dos sistemas de PV apresentam recursos para desenho bi ou tridimensionais, eles são sistemas grandes, funcionando como aplicativos, portanto não podendo ser integrados a um navegador *Web*. Por isso optou-se aqui pelo sistema *iVProg* que não apresenta ainda recursos para desenhos, mas dispõe de toda a estrutura necessária para introduzir os conceitos iniciais de programação. Além disso, utilizando o pacote *iTarefa*, o *iVProg* pode ser integrado facilmente ao sistema *Moodle* [Brandão e Rodrigues, 2013].

Na seção seguinte, discute-se a utilização de avaliação automática de exercícios de programação no ambiente *Moodle*.

3. Validação automática de exercícios de programação no *Moodle*

Nesta seção descreve-se resumidamente o sistema *Moodle* e os pacotes *iTarefa* e *VPL*, ambos disponíveis no sítio *Web* do Moodle.org. Também apresenta-se o sistema *iVProg*, para PV, em sua versão 2.

Como citado anteriormente, uma funcionalidade essencial para que sistemas educacionais *Web* possam ser empregados em cursos MOOC é a avaliação automática de tarefas. Para isso, o *Moodle* dispõe principalmente de questionários do tipo múltipla escolha, enquanto os sistemas *VPL* e *iTarefa*/*iVProg* implementam uma heurística para validar os exercícios de programação.

A técnica implementada em ambos é baseada na utilização de "casos de testes", por meio de um gabarito fornecido pelo professor. Esse gabarito é composto por um conjunto de dados de entradas e seus correspondentes dados de saídas, obtidos a partir de um algoritmo correto para o problema. Quando o aprendiz envia sua resposta, o sistema utiliza os dados de entrada do gabarito, submetendo-os ao algoritmo do aprendiz e comparando suas saídas com as respostas esperadas (no gabarito). O resultado da avaliação é produzido por uma heurísitica a partir das distâncias entre os dois conjuntos de saídas. Este processo é ilustrado na figura 4.

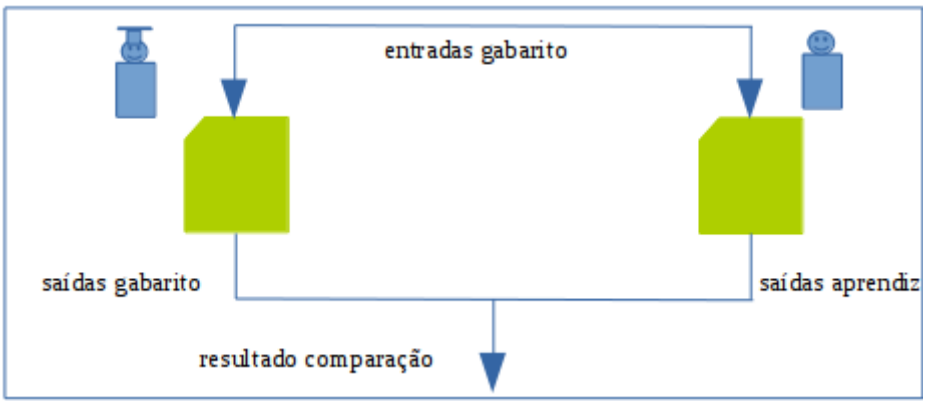

Figura 4. Modelo de validação baseado em gabarito.

3.1 O Sistema *Moodle*

O sistema *Moodle* foi iniciado em 1999 por Martin Dougiamas, sendo um acrônimo para *Modular Object-Oriented Dynamic Learning Environment*, embora o próprio Dougiamas afirme em uma entrada do fórum do *Moodle* que tenha pensado inicialmente no M como Martin's. A figura 5 apresenta esta entrada datada de 17 de julho de 2005.

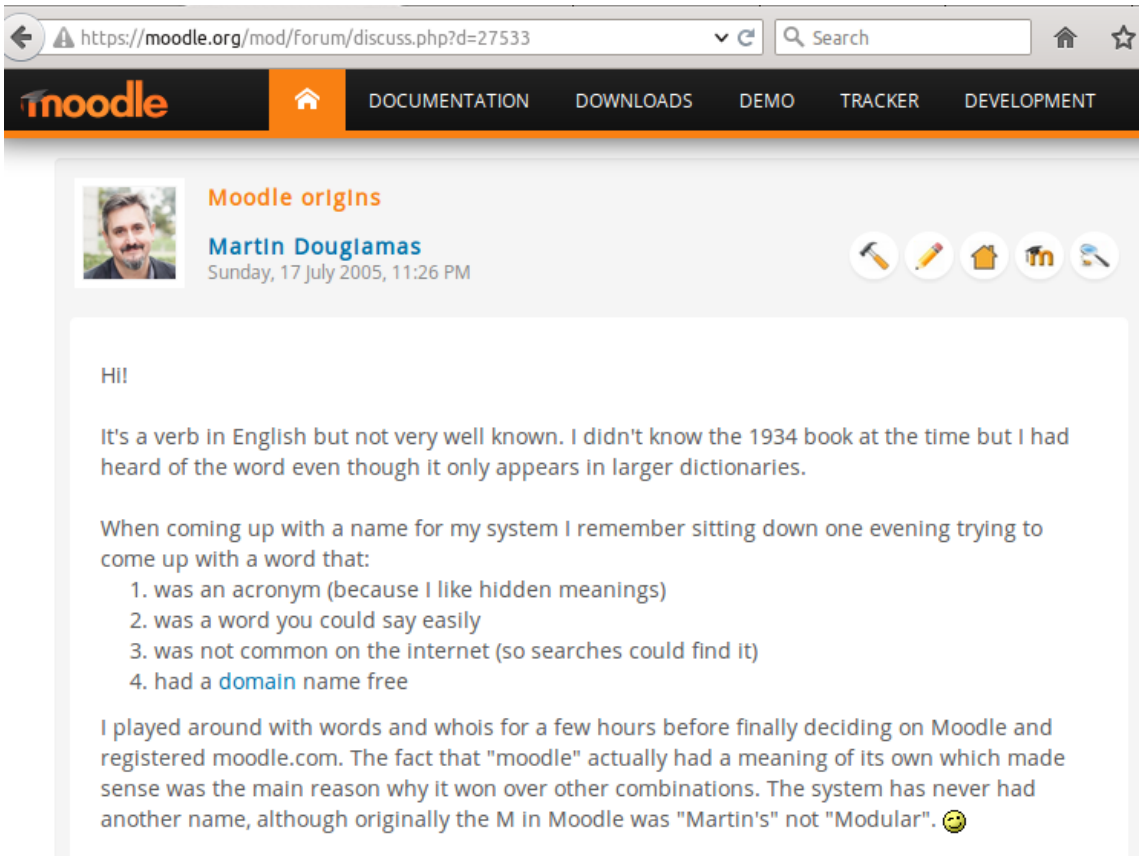

Figura 5. Mensagem no fórum do *Moodle* do principal responsável pelo sistema. Capturada em 05/07/2015.

Desde seu início o *Moodle* adotou o modelo de software livre e em novembro de 2001 uma versão do próprio *Moodle* passou a ser o gerenciador sob o endereço moodle.com. Dougiamas estruturou o crescimento do *Moodle* a partir de uma equipe

contratada, sendo atualmente disponibilizada uma nova distribuição *Moodle* a cada 6 meses[22].

O Moodle é desenvolvido em PHP e, sob o ponto de vista prático para os desenvolvedores, o interessante é que o *Moodle* adota uma arquitetura extensível, sendo possível adicionar novas funcionalidades por meio de *pacotes adicionais* (*plugins*).

Nas figuras 6 e 7 são apresentadas as interfaces da página inicial de cursos de um usuário do tipo professor registrado, do *Moodle*, respectivamente, em nas versões 1.9 e 2.7.

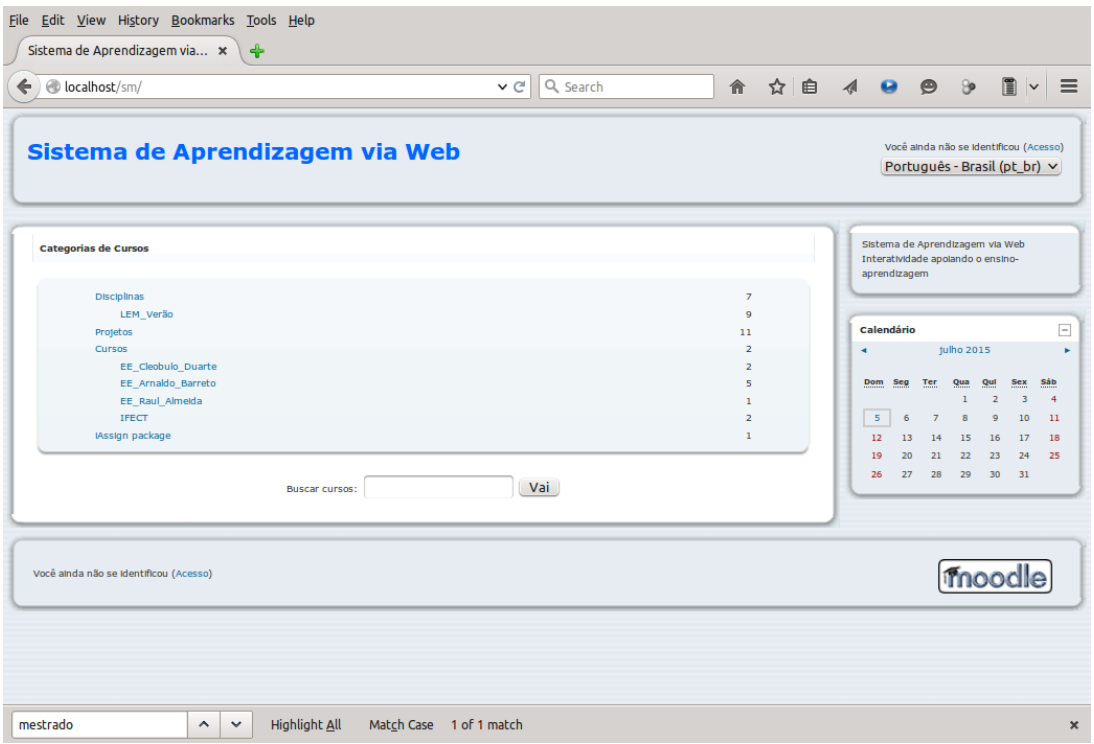

Figura 6. Interface inicial do *Moodle* em sua versão 1.9

22 Estas informações podem ser encontradas no moodle.org, hoje sob o endereço https://docs.moodle.org/29/en/History.

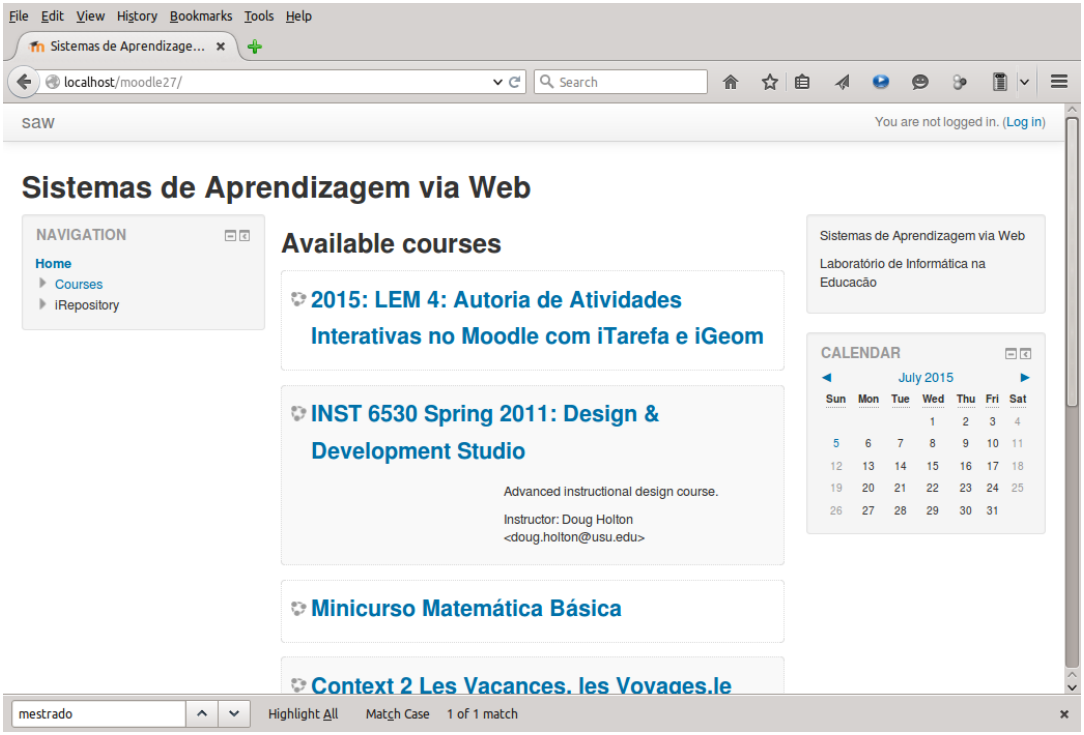

Figura 7. Interface inicial do *Moodle* em sua versão 2.7.

Por questões de segurança, apenas o administrador do ambiente *Moodle* tem permissão para adicionar (ou remover) esses pacotes adicionais, como por exemplo, os pacotes *iTarefa* e o *VPL*.

3.2 O pacote *iTarefa*

O pacote *iTarefa* (*Tarefas Interativas na Internet*) foi iniciado no segundo semestre de 2008, para substituir o sistema *SAW* (Sistema de Aprendizagem via *Web*), um gerenciador de cursos desenvolvido desde 2000 no IME-USP [Moura e Brandão, 2005]. O *iTarefa* fez parte do projeto de mestrado de Patrícia Alves Rodrigues [Rodrigues, 2011].

A ideia do projeto *iTarefa* é proporcionar a integração de sistemas para ensino/aprendizagem de tópicos específicos (como geometria ou programação) de modo flexível. Para isso implementa os requisitos para Módulos de Aprendizagem Interativa (*iMA*), definidos a partir do projeto *SAW*. Exemplos de *iMA* são os sistemas *iGeom* para Geometria Interativa [Brandão e Isotani, 2003] e *iVProg* para programação/construção de algoritmos [Kamiya e Brandão, 2009; Ribeiro et al., 2012].

Na figura 8 aparece a interface do *iTarefa*, em sua versão para Moodle 2.X, para visualizar uma tarefa interativa, no exemplo uma atividade para construir a mediatriz entre dois pontos utilizando o *iGeom*.

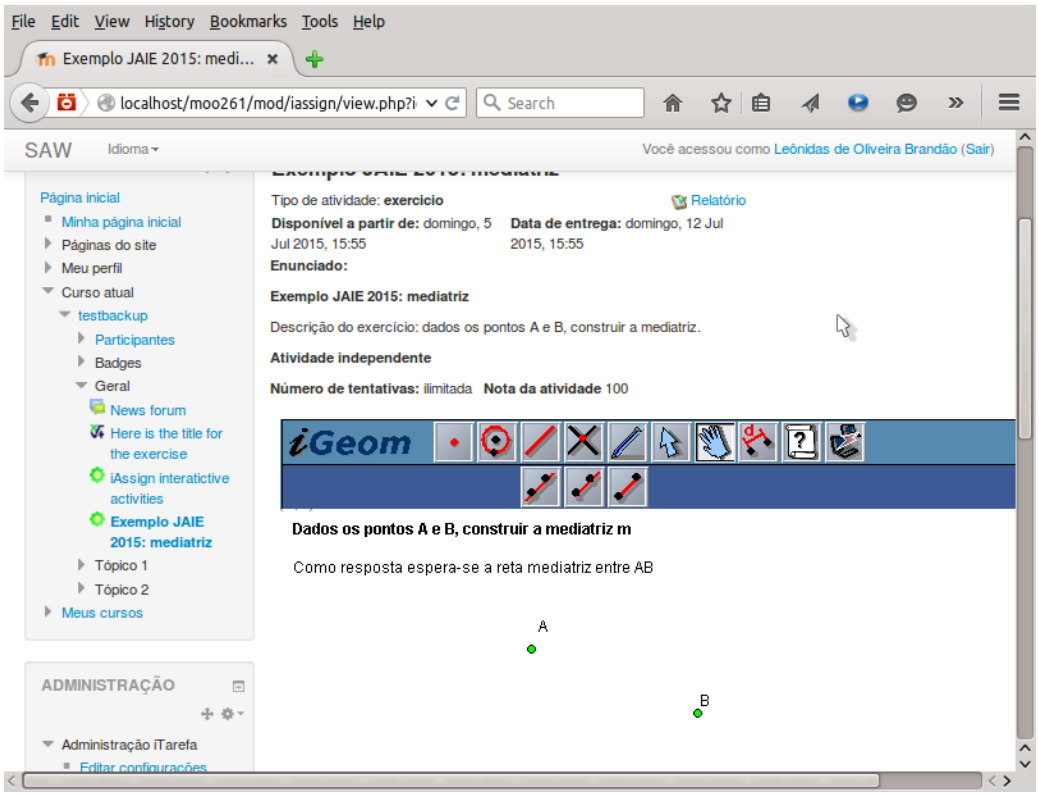

Figura 8. *iGeom* executando dentro do módulo *iTarefa* no *Moodle*.

Deste modo o *administrador* pode adicionar novos *iMA*. Em princípio qualquer *applet* ou pacote *HTML* poderia ser integrado, desde que seu autor implemente os métodos de *comunicação* requeridos.

Com o *iTarefa* o *professor* pode inserir novos exercícios, pode compartilhar com outros professores seus exercícios. Existe também um relatório que apresenta todos os exercícios *iTarefa* do curso, na forma de uma tabela cujas linhas são os alunos e as colunas os exercícios. Além do resumo de todos os envios, se o professor clicar numa das entradas da tabela, poderá examinar o que o aprendiz enviou para o correspondente exercício. O sistema de notas do *iTarefa* é integrado à grade de notas do *Moodle* (*gradebook*).

O *iTarefa* tem uma outra extensão que possibilita que o professor insira tarefas interativas em meio a qualquer texto (extensão do tipo *filtro*).

Para o *aluno/aprendiz*, o envio de suas atividades pode ser realizado com um clique. Ele também pode examinar seu último envio e, dependendo de como o professor configurou o exercício, pode enviar sua resposta mais de uma vez.

Mais detalhes sobre o pacote *iTarefa* podem ser encontrados no texto de Brandão e Rodrigues (2013).

3.3 O sistema iVProg

O *iVProg* – Programação Visual Interativa na Internet, foi desenvolvido a partir do sistema *Alice* para ser um *iMA*, tendo utilizado a versão 2.0 do Alice.

Como já citado, o *Alice* é um ambiente de programação criado na universidade Carnegie Mellon. Até sua versão 2.4, era mantido unicamente em versão inglês, sem

opção de internacionalização. O *Alice* é implementado em Java e dispõe de interfaces para representações gráficas tridimensionais, podendo ser utilizado para a criação de jogos.

Entretanto, para tornar viável carregar o *iVProg* como *applet*, foi cortada toda a parte de animação do Alice 2.0, deste modo reduzindo o tamanho de 120 megabytes no *Alice* para menos de 1 megabyte no *iVProg* [Kamiya, 2009]. Outra diferença no *iVProg* é ele ter sido desenhado para introduzir algoritmos a partir do modelo de programação estruturada, enquanto o *Alice* foca Orientação a Objetos.

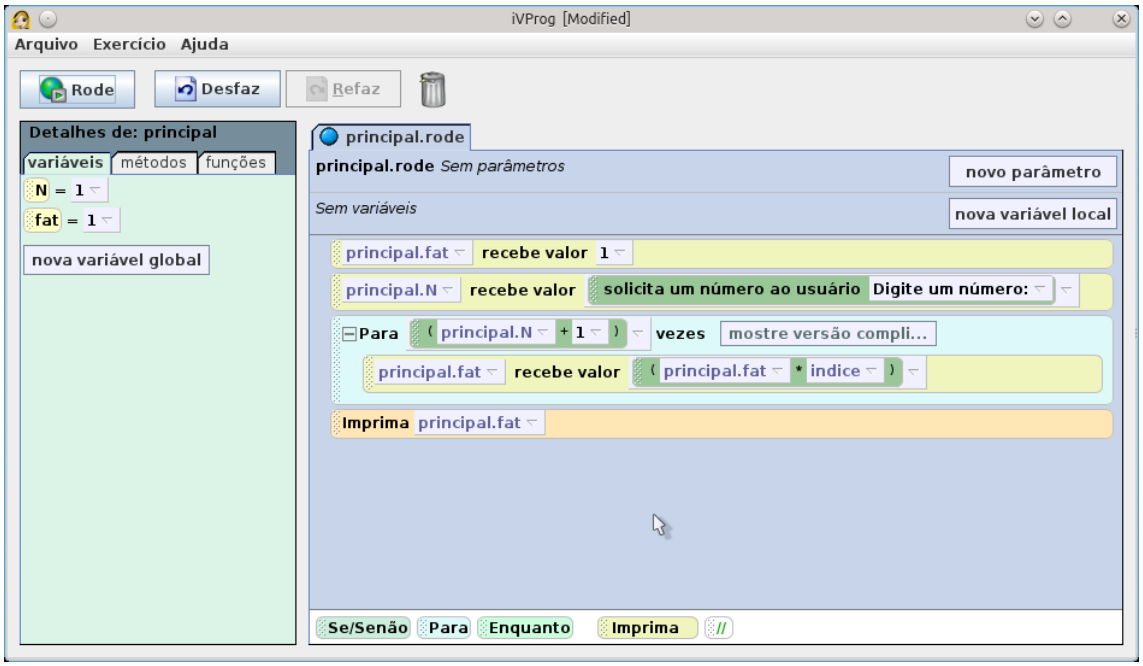

Figura 9. Imagem do *iVProg* versão 0.9 com algoritmo para fatorial.

A ideia do *iVProg* é possibilitar aos aprendizes construir seus algoritmos a partir de blocos representando comandos e variáveis. Para executar o código basta um clique no botão "rode". Na Figura 9 está uma imagem da versão 0.9 do *iVProg*.

Nessa versão, utilizava-se o *mouse* exclusivamente com a opção de "arrastar" (*drag-and-drop*) para a construção do algoritmo a partir de seus blocos de comandos.

A partir de 2011 foi gerada uma versão completamente nova do sistema iVProg, que passou a ser denominado *iVProg 2*. Essa versão empregou um arcabouço produzido a partir de uma Linha de Produto de Software (LPS) para *iMA* [Dalmon e Brandão 2012].

Em sua segunda versão ainda é possível utilizar mecanismos de interação do tipo "arrastar" (*drag-and-drop*), porém o iVProg 2 também implementa o modo "clique-pega, clique-cola" (*point-and-click*), que é mais eficaz para usuários experientes [Inkpen, 2001].

Mesmo tendo sido inspirado no *Alice*, ambas as versões foram projetadas para o ensino de programação estruturada, enquanto o *Alice* é destinado ao ensino de programação orientada a objetos.

Quanto à tecnologia, o *iVProg 2* é um *iMA*, implementado em Java na forma de *applet,* portanto podendo ser integrado a um navegador *Web*. A figura 10 é uma tela do

sistema *iVProg* 2, com um algoritmo para obtenção de fatorial de número natural, com a janela para entrada de dados em primeiro plano.

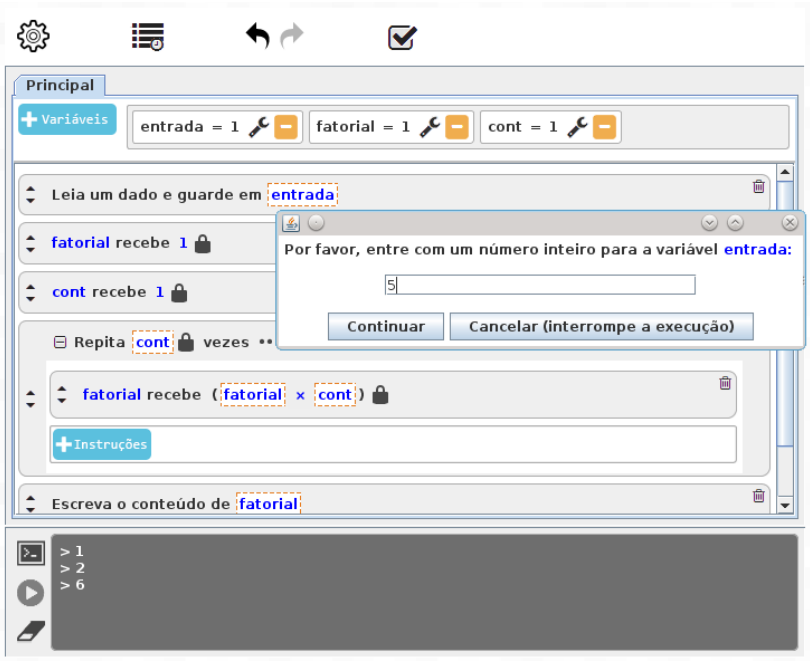

Figura 10. Tela do *iVProg* 2 rodando um algoritmo para cálculo de fatorial.

O iVProg 2 apresenta várias novidades em relação à versão anterior, sendo a principal seu mecanismo para a avaliação automática de atividades. Outra novidade visível é a integração da área para saída de dados, o que elimina a necessidade de lançamento de uma janela para isso. O mecanismo de avaliação automática é baseada no modelo de "casos de teste" explicada na subseção 3.4.

A subseção seguinte apresenta o pacote VPL para *Moodle*, que permite a criação de algoritmos em várias linguagens de programação.

3.4 O sistema VPL

O VPL é um dos módulos do *Moodle* e está disponível na área de *downloads*: <*https://moodle.org/plugins/view.php?plugin=mod_vpl*>. Ele é desenvolvido por membros da *Universidad de las Palmas Gran Canaria*, na Espanha e permite a construção de algoritmos em diversas linguagens de programação.

Para o funcionamento desse ambiente é necessário um trabalho adicional além daquele usualmente realizado por um administrador do *Moodle*. Por razões de segurança, é preciso fazer uma instalação de um segundo servidor, o que precisa ser feito por pessoa que entenda da administração de redes de computadores, geralmente uma competência além da formação exigida para um administrador de ambiente de curso.

Tal servidor engaiolará os compiladores utilizados pelo VPL e receberá todas as requisições de execução dos exercícios. Desse modo, se um aprendiz cometer algum erro involuntário ou no caso dele submeter um código malicioso que consiga burlar todas as medidas de segurança implementadas pelo *Moodle* e pelo VPL, a integridade do servidor pode ser mas facilmente garantida. No pior dos casos apenas a máquina virtual contendo o compilador será danificada, podendo ela ser reiniciada rapidamente. Esta é a razão pela qual tal tipo de solução é denominada por *gaiola ou jaula* (*jail system*, em Inglês).

Na documentação do VPL, encontrada no site oficial do módulo, explica-se sobre a instalação do sistema de gaiola[23] e algumas de suas configurações[24] realizadas no ambiente Ubuntu. A instalação do módulo do *Moodle* segue o padrão de instalação de qualquer módulo de atividades.

Depois de instalado o módulo VPL e iniciado o servidor com os compiladores, é necessário configurar o endereço de IP do servidor da gaiola no módulo. O intervalo das portas de comunicação que receberão as requisições a partir do *Moodle* e outros detalhes. A Figura 11 mostra o trecho dessas configurações no qual deve ser inserido o endereço IP do segundo servidor (gaiola) e quais portas devem ser usadas. Essas são as principais configurações, as demais estão relacionadas a detalhes sobre os arquivos a serem enviados pelos aprendizes, como o tamanho do arquivo submetido e tempo máximo de execução de cada requisição.

Para permitir que o usuário (do tipo *aluno*) digite seu código existe um *applet Java* integrada ao módulo de atividades do VPL instalada no *Moodle*, como indicado na Figura 12. O *applet* provê algumas facilidades para o estudante, como seleção do tamanho da fonte e iluminação da sintaxe da linguagem utilizada.

Figura 11. Interface do menu de configuração do VPL no *Moodle*.

23 http://vpl.dis.ulpgc.es/index.php/support/2014-09-27-15-06-22/40-how-to-install-vpl-jail-system

24 http://vpl.dis.ulpgc.es/index.php/support/2014-09-27-15-06-22/66-execution-server-configuration

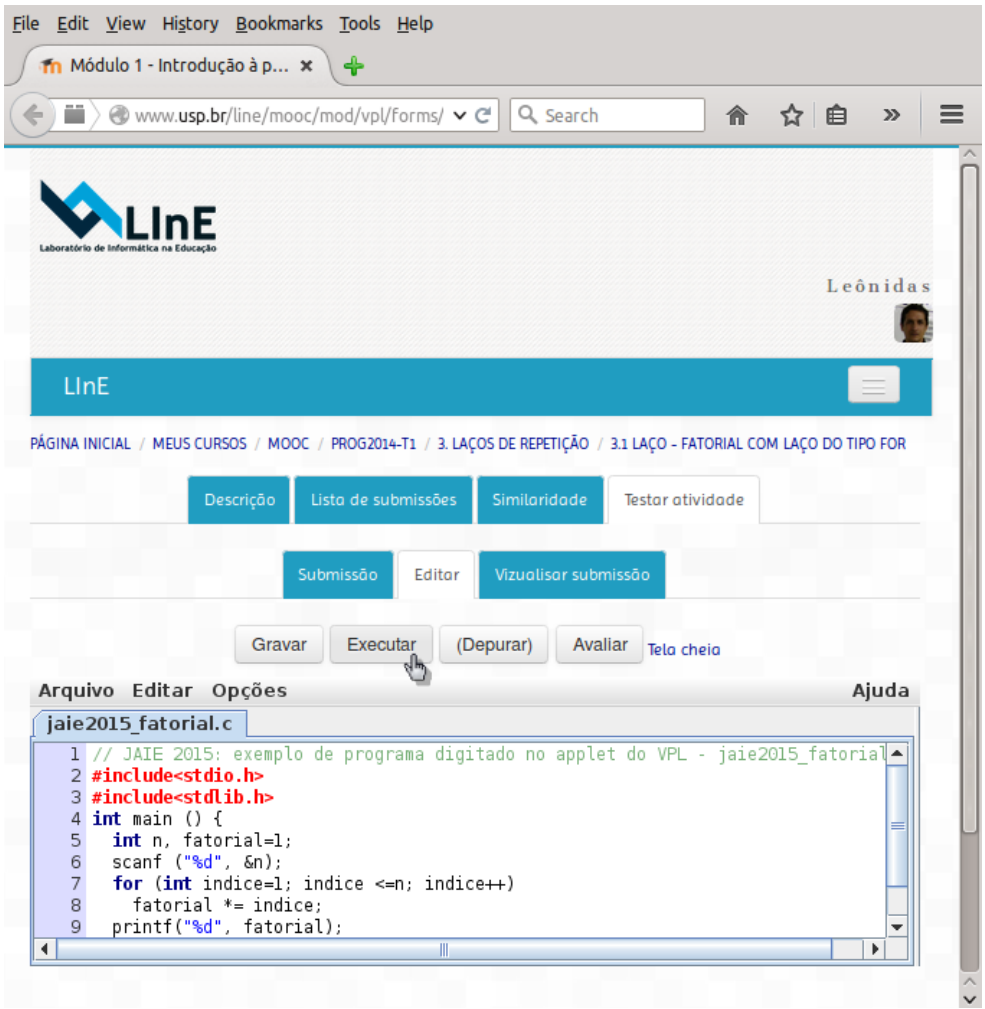

Figura 12. Interface principal do VPL como *applet* para editar código.

Para o *cadastramento de exercícios* no VPL, um usuário do tipo *professor*, deve cadastrar um gabarito contendo as entradas para testes e as correspondentes saídas esperadas, como no modelo explicado na seção 3. Note que a construção do gabarito é um processo manual, não existindo (ainda) uma interface que dado o algoritmo "correto" possa facilmente gerar as entradas e suas saídas.

Existe um formato específico para o gabarito, que é uma tripla contendo cada "caso de teste", sendo composta por 3 rótulos, *case*, *input* e *output*, O primeiro é o nome do teste, o segundo uma lista de valores para entradas e o último a correspondente lista de saída. Pode existir tantas triplas quanto forem necessários os casos para testes. A documentação do VPL descreve este formato (http://vpl.dis.ulpgc.es/index.php/support) e a Figura 13 apresenta a primeira tripla para testes para o exercício do cálculo de fatorial (para n=0, devendo resultar o valor 1).

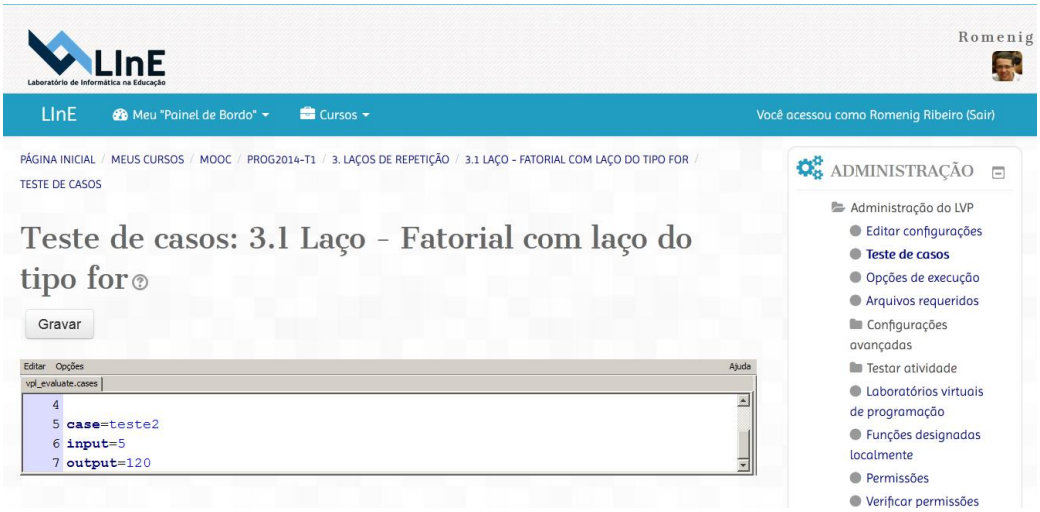

Figura 13. Captura de tela da configuração de teste de caso do VPL.

Do ponto de vista do aprendiz, a tarefa é muito mais simples, pois ele pode digitar seu código dentro do *applet* com o editor de texto para código (vide Figura 14). Pode, com um clique obter os resultados da compilação de seu programa ou o resultado dele a partir do gabarito do professor (e consequentemente, conhecer sua "nota" para o exercício - com a dada solução).

Como indicado nas figuras 12 e 14, o aprendiz pode gravar seu código, pode executá-lo, examinar o resultado da compilação (eventuais mensagens de erro) ou solicitar sua avaliação (automática). Para existir esta última opção é necessário que o professor tenha cadastrado o gabarito do exercício.

Assim como no pacote *iTarefa*, o professor também pode ter acesso a todas as submissões, com uma pequena diferença, o VPL mantém o registro de todas as submissões de um aluno para determinado exercício, enquanto o *iTarefa* registra apenas última.

Figura 14. Tela do VPL executando no *Moodle*.

4. Considerações finais

Neste trabalho foi apresentado uma contextualização para cursos do tipo *Massive Open Online Course* (*MOOC*) para introdução de programação.

Em particular foram citados alguns dos trabalhos que abordam as dificuldades envolvidas na tarefa de introduzir os conceitos básicos e a importância do tema para a atualidade. Como forma de reduzir alguns dos problemas citados, foi apresentado o modelo de programação visual, em particular o sistema *iVProg*.

Também foi apresentado o contexto histórico da EAD, focando principalmente sua situação no Brasil. Vale destacar os dados do Exame Nacional de Desempenho dos Estudantes (Enade) que tem apontado para desempenhos similares entre alunos egressos de cursos tradicionais (presenciais) e aqueles oriundos de cursos na modalidade EAD. A questão de evasão nos cursos *MOOC* também foi citada.

Depois foram apresentados o sistema *Moodle* e seus pacotes adicionais *iTarefa* e VPL, os quais possibilitam a construção de ambientes para cursos de programação, uma vez que dispõem de recursos para avaliação automática de exercícios, seja de programação tradicional (com *C*) ou visual (com *iVProg*).

Por fim deve-se destacar a possibilidade de utilização desse ambiente *Moodle*+*iTarefa*+VPL para rodar experimentos que permitam um melhor entendimento do papel da programação visual. Em particular, estudar em quais situações poderia ela ser mais bem empregada.

Referências

Adams, J. C. “Alice, middle schoolers & the imaginary worlds camps.” *SIGCSE Bulletin*, Março de 2007: 307-311.

Alvarez, A. M. T. “Produto 1 - Panorama e diagnóstico da oferta e qualidade da Educação Superior brasileira.” *Ministério da Educação*. 15 de Abril de 2013. http://portal.mec.gov.br/index.php?option=com_docman&task=doc_details&gid=13944&Itemid= (acesso em 04 de Julho de 2015).

Alves, L. “Educação a distância: conceitos e história no Brasil e no mundo.” *Revista Brasileira de Aprendizagem Aberta e a Distância*, 2011: 83-92.

Armoni, M., Meerbaum-Salant, O., Ben-Ari, M. “From Scratch to "Real" Programming.” *ACM Transactions on Computing Education*, Fevereiro 2015: 25:1-25:15.

BRASSCOM. “www.brasscom.org.” *Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação.* 2012. http://www.brasscom.org.br/brasscom/Portugues/download.php?cod=437 (acesso em 15 de 07 de 2015).

Beaubouef, T., e Mason, J. “Why the high attrition rate for.” *SIGCSE Bulletin*, 2005: 103-106.

Bitzer, D., Lyman, E., e Easley, J. The Uses of PLATO: a computer controlled teaching system. Relatório Técnico, Illinois: Coordinated Science Laboratory - University of Illinois, 1965.

Brandão, L. O.; Ribeiro, R. S.; Brandão, A. A. F. "A system to help teaching and learning algorithms." *Frontiers in Education Conference (FIE), 2012*. IEEE, 2012.

Brandão, L. O., S. Isotani. "Uma ferramenta para o ensino de Geometria Dinâmica na Internet: iGeom." *Anais do IX Workshop de Informática na Educação*, 2003: 146-148.

Brandão, L. O., Rodrigues, P. A.. "Incrementando a Interatividade em Cursos Web via Moodle: examinando o sistema iTarefa com o iGeom." *Jornada de Atualização em Informática na Educação*, 2013.

Dalmon, D. L., e L. O. Brandão. "Uma Linha de Produto de Software para Módulos de Aprendizagem Interativa." *Anais do Simpósio Brasileiro de Informática na Educação*, 2012.

Dann, W., D. Cosgrove, D. Slater, D. Culyba, e S. Cooper. "Mediated Transfer: Alice 3 to Java." Proceedings of the 43rd *ACM Technical Symposium on Computer Science Education*, 2012: 141-146.

DE. "The national curriculum in England – Key stages 1 and 2 framework document". Department for Education. 2013. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/425601/PRIMARY_national_curriculum.pdf (acesso em 10 de Agosto de 2015).

Freudenberg, B., Y. Ohshima, e S. Wallace. "Etoys for One Laptop Per Child." *Creating, Connecting and Collaborating through Computing*, 2009. C5 '09, 19-22 de Janeiro de 2009: 57-64.

Glinert, E. P., e S. L. Tanimoto. "Pict: An interactive graphical programming." *Computer*, Novembro de 1984: 7-25.

INEP-1. "Censo da educação superior 2012 – resumo técnico". *Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira* (INEP). Julho de 2014. http://download.inep.gov.br/download/superior/censo/2012/resumo_tecnico_censo_educacao_superior_2012.pdf (acesso em 04 de Julho de 2015).

INEP-2. "Resumo Técnico: Censo da Educação Superior 2008." *Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira*. 2009. http://download.inep.gov.br/download/censo/2008/resumo_tecnico_2008_15_12_09.pdf (acesso em 04 de Julho de 2015).

Inkpen, K. M. "Drag-and-drop versus point-and-click mouse interaction styles for children." *ACM Transactions on Computer-Human Interaction* (TOCHI), Março de 2001: 1-33.

Kamiya, R. R. "iVProg: um sistema visual para ensino/aprendizagem de programação via Web." Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação. Instituto de Matemática e Estatística: Universidade de São Paulo - São Paulo, 2009.

Kamiya, R. R., e L. O. Brandão. "iVProg-um sistema para introdução à Programação através de um modelo Visual na Internet." *Anais do XX Simpósio Brasileiro de Informática na Educação*, 2009.

Kinnunen, P., e L. Malmi. "Why students drop out CS1 course?" *Proceedings of the second international workshop on Computing education research*, 2006: 97-108.

Lahtinen, E., K. Ala-Mutka, e J. Hannu-Matti. *Proceedings of the 10th annual SIGCSE conference on*, 2005: 14-18.

Lobo, R. L., P. R. Motejunas, O. Hipólito, e M. B. C. Lobo. "A evasão no ensino superior brasileiro." *Cadernos de Pesquisa*, Setembro-Dezembro de 2007: 641-659.

Mackness, J., S. Mak, e R. Williams. "The ideals and reality of participating in a MOOC." *Proceedings of the 7th International Conference on Networked Learning*, 2010: 266-275.

Moura, J. G., e L. O. Brandão. "Aplicações no SAW - Sistema de Aprendizagem pela Web." *Anais do XVI Simpósio Brasileiro de Informática na Educação* (SBIE), 2005.

Myers, B. A. "Taxonomies of visual programming and program visualization." *Journal of Visual Languages & Computing*, Março de 1990: 97-123.

Neto, F. J. S. L. "Educação a Distância: Regulamentação, Condições de Êxito e Perspectivas". 11 de Fevereiro de 1998. URL: http://www.feg.unesp.br/~saad/zip/RegulamentacaodaEducacaoaDistancia_lobo.htm (último acesso em 04/07/2015).

NSTC. "Federal science, technology, engineering, and mathematics (STEM) education 5-year strategic plan: A report from the committee on STEM education national science and technology council", Washington, 2013. URL:.https://www.whitehouse.gov/sites/default/files/microsites/ostp/stem_stratplan_2013.pdf (último acesso em 07/07/2015).

OFSTED 2011. "ICT in schools 2008-11: An evaluation of information and communication technology education in schools in England 2008-11", *London: Office for Standards in Education*, 2011.

Paiva, G. S. "Avaliação do desempenho dos estudantes da educação superior: a questão da equidade e obrigatoriedade no Provão e Enade". *Ensaio: Avaliação e Políticas Públicas em Educação*, jan./mar. de 2008: 31-46.

Regalado, A. "The Most Important Education Technology in 200 Years. A Business Report on Digital Education", *MIT Technology*, 2013.

Repenning, A., e A. Ioannidou. "Agent-based end-user development." *Communications of the ACM - End-user development: tools that empower users to create their own software solutions*, 2004: 43-46.

Ribeiro, R. S., L. O. Brandão, P. A. Rodrigues, S. Isotani, e A. A. F. Brandão. "iVProg e iTarefa: aprimorando o ensino de algoritmos e programação para iniciantes." *Anais dos Workshops do Congresso Brasileiro de Informática na Educação* (APPLETS), 2012.

Robins, A., J. Rountree, e N. Rountree. "Learning and Teaching Programming: A Review and Discussion." *Computer Science Education* , 2003: 137-172.

Robins, A., P. Haden, e S. Garner. "Problem distribution in a CS1 course." *Proceedings of the 8th Australasian Conference on Computing Education*, 2006: 165-173.

Rodrigues, P. A. iTarefa: componente Moodle para incorporar módulos de aprendizagem interativa em cursos Web. *Dissertação de Mestrado*, São Paulo: Universidade de São Paulo - Pós-Graduação em Ciência da Computação, 2011.

Seaton, D. T., Y. Bergner, I. Chuang, P. Mitros, e D. E. Pritchard. "Who does what in a massive open online course?" *Communications of the ACM*, Abril de 2014: 58-65.

Souza, M. J. G. MOOC de Geometria: discussões e proposta de um modelo para a educação básica. *Dissertação de Mestrado*, São Paulo: Universidade de São Paulo - Pós-Graduação em Ciência da Computação, 2015.

Stefano, F. "A Minimal, Extensible, Drag-and-drop Implementation of the C Programming Language." *Proceedings of the 2011 Conference on Information Technology Education*, 2011: 191-196.

Taneja, S., e A. Goel. "MOOC Providers and their Strategies". *International Journal of Computer Science and Mobile Computing* 3.5, 2014: 222-228.

Van Rossum, Guido. "Computer programming for everybody". *Proposal to the Corporation for National Research Initiatives*, 1999. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.6836&rep=rep1&type=pdf (última visita em 10/07/2015).