Saci – ainda outro ambiente para o ensino de programação

Ricardo Anido

Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
13983-970 – Campinas – SP – Brazil

ranido@ic.unicamp.br

Abstract. *This paper presents an integrated system for teaching and learning computer programming. The system, named Saci, comprises a sub-system for managing users, a sub-system for managing classes, and the main teaching environment which is implemented as a single-page Web application. The learning environment incorporates tools for visualizing the class contents (video, supporting documents), a set of practical exercises, tools for editing the source-code, executing and verifying the program correctness, and some simple help tools. After a "class", pre-built by a teacher, is loaded into a student's Web browser, all processing happens in the browser, with no configuration or installation needed in the student's computer. A first course using the Saci system was deployed mid-March by the Brazilian Olimpyad in Informatics.*

Resumo. *Este artigo apresenta um sistema integrado para o ensino e aprendizagem de programação de computadores. O sistema, chamado Saci, compreende um subsistema de gerência de usuários, um subsistema de gerência de aulas, e o ambiente de aprendizagem propriamente dito, implementado como um aplicativo Web de página única. O ambiente de aprendizagem incorpora ferramentas para visualização do conteúdo da aula (vídeo, documentos de apoio), edição do programa-fonte, execução e verificação da correção do programa, além de ferramentas de ajuda ao aluno. Após o carregamento de uma "aula", pré-montada por um professor, no navegador Web de um aluno, todo o processamento ocorre no navegador, não sendo necessária nenhuma configuração ou instalação no computador do aluno. Um primeiro curso utilizando o sistema Saci, de introdução à programação de computadores usando Javascript, foi disponibilizado na página da Olimpíada Brasileira de Informática em meados de março.*

1. Introdução

Uma multitude de projetos têm sido desenvolvidos para tornar linguagens e ambientes de programação acessíveis para uma audiência mais ampla, especialmente jovens estudantes. Keheller apresenta um extenso survey de tais projetos, de 1960 até 2005 [Kelleher and Pausch 2005]. Também no Brasil o interesse se manifesta, com inúmeros artigos nos dois principais eventos da comunidade brasileira de ensino em computação, o Simpósio Brasileiro de Informática na Educação (SBIE) e o Workshop sobre Educação em Computação (WEI).

Trabalho realizado com o apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo 2014/26560-5.

A Olimpíada Brasileira de Informática (OBI) é uma competição promovida anualmente pela Sociedade Brasileira de Computação, em que alunos dos ensinos fundamental e médio devem resolver exercícios de programação (modalidade Programação, para alunos com algum conhecimento em programação) ou exercícios de lógica (modalidade Iniciação, para alunos sem conhecimento em programação). Atualmente, a grande maioria dos participantes é da modalidade Iniciação, e a OBI tem interesse em que esses alunos sejam incentivados a aprender programação para poderem participar da modalidade Programação.

Uma das abordagens possíveis para expor uma grande quantidade de alunos à programação de computadores é a criação de cursos programação on-line, gratuitos, que possibilitem o ensino tanto de alunos iniciantes como de alunos interessados em tópicos avançados. Os cursos deveriam ser de preferência no estilo de auto-aprendizagem guiada. Para isso foi feita uma pesquisa para verificar a existência de ambientes disponíveis que atendessem aos requisitos da OBI.

1.1. Soluções encontradas na literatura e razões para um novo ambiente de aprendizagem

Embora alguns dos projetos encontrados na literatura sejam extremamente exitosos, ainda podemos notar algumas deficiências, que levaram à busca por uma nova abordagem.

Talvez a primeira e mais importante barreira para estudantes brasileiros usarem as ferramentas existentes é que elas foram desenvolvidas para um público que tem inglês como sua primeira língua. Algumas ferramentas provêm meios de internacionalização: vídeos podem ser legendados, alguns materiais podem ser traduzidos. Mas para a maioria dos ambientes a experiência exige algum conhecimento da língua inglesa, e não apenas referente às palavras-chave da linguagem de programação.

Scratch [Maloney et al. 2008, Resnick et al. 2009] e Code.org [CodeOrg 2015] são duas ferramentas que têm feito um bom esforço de tradução para o português. Ambas usam uma linguagem de programação baseada em blocos, similares a LogoBlocks [Begel 1996], no qual formas gráficas rotuladas representam comandos em um dialeto de Logo [Papert 1980]. Estudantes criam programas arrastando esses blocos gráficos a partir de uma palheta ao lado da tela para uma área de trabalho principal, onde os blocos podem ser colocados adjacentes a outros blocos para construir a sequência de comandos desejada. Os ambientes de programação de Scratch e Code.org são muito similares entre si, e muito atrativos visualmente. Tópicos são introduzidos por pequenos vídeos, e exercícios são baseados em jogos. Tanto Scratch como Code.org são ótimas ferramentas para a introdução de programação para programadores muito jovens, mas consideramos que, após os estágios iniciais, a abordagem de programação por blocos se torna uma amarra que impede a introdução de algoritmos mais complexos.

Portugol Studio é um ambiente de programação para iniciantes em programação [Noschang et al. 2014]. Ele permite o desenvolvimento, execução e depuração de programas escritos em Portugol, que é uma linguagem em formato de pseudo-código com comandos em português. Portugol Studio resolve bem o problema da língua é muito bom para iniciantes em programação, especialmente por utilizar uma linguagem de pseudo-código bem próxima do português. No entanto, pelas limitações do pseudo-código, o ambiente não permite a introdução de tópicos mais complexos em algoritmos. O fato de

ser um aplicativo para desktop também dificulta a criação de pacotes de recursos integrados à ferramenta, para serem baixados e utilizados exclusivamente pela Internet.

Vários dos ambientes existentes foram projetados como aplicativos para desktop, muitas vezes porque, pela complexidade, necessitam de um maior poder computacional. Para que o aluno possa praticar em casa, no seu computador pessoal, é normalmente necessário instalar as ferramentas do ambiente (editor, compilador, IDEs, etc.), o que pode ser uma tarefa difícil. Mesmo considerando que não se precise pagar licença pelo software, a tarefa de baixar e instalar um aplicativo pode se revelar intimidadora ou mesmo impossível, ou porque o aluno não tem o domínio das ferramentas necessárias para baixar software em seu computador, ou porque se sinta intimidado pela tarefa, ou porque não tem acesso privilegiado para instalar novos softwares no computador de sua casa.

App Inventor é uma plataforma de programação para criação de aplicativos móvies para dispositivos baseados em Android [Abelson and Friedman 2010]. Como em Scratch, o estudante não precisa digitar código para desenvolver aplicativos em App Inventor. Os estudantes projetam visualmente a interface gráfica e usam blocos de componentes conectáveis para controlar o comportamento do aplicativo. O principal atrativo é que os aplicativos executam em telefones celulares tipo *smartphones*, e estudantes podem compartilhar seus aplicativos e vê-los executando em dispositivos reais [Gray et al. 2012]. App Inventor é muito atraente e normalmente consegue um bom engajamento dos estudantes, mas necessita que professores e estudantes tenham acesso a dispositivos móveis 'smart', e que se baixe e configure o ambiente (embora seja um esforço mínimo, pode se provar difícil e intimidante na prática, como mencionado acima).

Um Juiz *on-line* é um sistema que corrige submissões de programas de computador automaticamente, baseado em um gabarito pré-fornecido. Eles em geral incluem um repositório com centenas, ou mesmo alguns milhares de problemas coletados de competições, e são usados para receber, executar, verificar a correção de uma solução e retornar o resultado para o usuário. *URI OnLineJudge Academic* [Tonin et al. 2013] e *CodeChef for Schools* [CodeChef 2015] são dois exemplos de juízes que oferecem também facilidades para professores criarem disciplinas com exercícios dos repositório. A principal deficiência dos sistemas baseados em juizes on-line é a falta de integração com recursos adicionais para uma aula, como por exemplo vídeo e outros recursos de apresentação. MOJO [Chaves et al. 2013] é uma ferramenta que procura resolver esse problema, realizando a integração de um juiz on-line com o ambiente Moodle. No entanto, o aluno devem instalar um ambiente de desenvolvimento em seu computador para implementar suas soluções, o que novamente traz à tona o problema de baixar e configurar aplicativos.

Khan Academy, uma organização famosa por suas lições de vídeo em uma variedade de disciplinas (matemática, ciências, etc) iniciou em 2012 uma vertente em ciência da computação [KhanAcademy 2015]. Ela usa javascript como linguagem base, e as aulas, como em todas as outras disciplinas da Khan Academy, são acompanhadas por vídeos de curta duração. O ambiente de aprendizagem é muito bem construído, com uma profusão de elementos gráficos (desenhos e animação) e sons. Os estudantes escrevem código-fonte para construir os programas, mas o ambiente é amigável, detectando erros de sintaxe e sugerindo como corrigí-los. No entanto, a ferramenta é construída considerando que a entrada para o programa são *interações* como a posição ou movimento do mouse, o que torna mais difícil propor problemas de natureza mais algorítmica.

Após analisar os ambientes disponíveis, e considerando as razões apontadas acima, decidimos contruir um novo ambiente de aprendizagem, que se adequasse mais diretamente aos requisitos dos cursos da OBI: (i) uso de uma linguagem real, que permita o desenvolvimento de tópicos mais avançados; (ii) possibilidade de configurar "aulas", compostas de material de apoio e de exercícios; (iii) correção automatizada das submissões dos alunos e retro-alimentação dos resultados; (iv) possibilidade de entrada e saída na forma de texto, que os alunos se acostumem ao modelo utilizado nas provas da OBI.

O restante deste artigo apresenta o aplicativo Web do ambiente de aprendizagem Saci, com foco em suas funcionalidades e alguns detalhes de implementação. Os outros componentes do Sistema Saci (subsistemas de gerência de usuários e de gerência de aulas) são sistemas de processamento de dados normais, e não necessitam de mais detalhes.

2. O ambiente de aprendizagem Saci

O ambiente de aprendizagem Saci foi implementado em Javascript e projetado como um aplicativo Web de página única. A tela do ambiente é dividida em três áreas: Área de Vídeo, Área de Conteúdo e Área de Trabalho, com o layout mostrado na Figura 1(a).

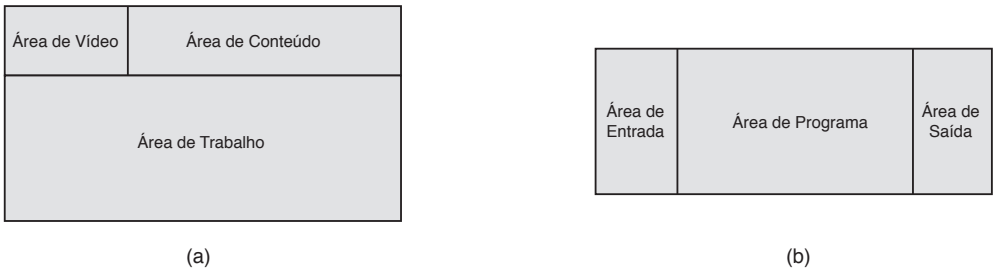


Figura 1. Layout do ambiente Saci.

A Área de Vídeo contém o vídeo da aula, com apresentações curtas sobre o tópico da aula (5 a 10 min). O vídeo é armazenado no YouTube, e o tocador embutido na Área de Vídeo é configurado com os controles normais da API do YouTube, podendo ser expandido para visualização em tela cheia. Se a aula não contém recurso de vídeo, o painel da Área de Vídeo não é mostrado.

A Área de Conteúdo contém "abas", que podem ser selecionadas para mostrar o seu conteúdo. A aba de Revisão contém textos e documentos associados ao conteúdo da aula, como slides da apresentação do vídeo, textos sobre o tópico, links para outras informações e trechos de programa. As abas de exercícios contêm um conjunto de exercícios com o objetivo de fixar o conhecimento do tópico da aula. Finalmente, a aba de Soluções contém exemplos comentados de soluções para cada um dos exercícios.

O componente principal do ambiente é a Área de Trabalho, que é similar à utilizada no Khan Academy CS, mas com a adição de uma Área de Entrada. O layout básico da Área de Trabalho é mostrado na Figura 1(b).

Ela é composta por:

- Painel de Entrada (*editável*), onde são colocados os valores a serem usados como entrada pelo programa

- Painel de programa, que contém um editor para edição do código-fonte do programa
- Painel de saída (*não editável*) onde é mostrada a saída do programa e outras mensagens do sistema.

O Editor usado no Painel de Programa é o CodeMirror [Haverbeke 2015]. Ele é configurado para mostrar a sintaxe da linguagem usada (Javascript ou Python) com cores e para detecção de erros de sintaxe. O módulo de mensagens de Javascript foi re-escrito com todas as mensagens de erro traduzidas para português.

O aplicativo de página única foi implementado utilizando jQWidgets [jQWidgets 2015], sob permissão. Um exemplo de página do ambiente SACI é mostrado na Figura 2, com as Áreas de Vídeo, Conteúdo e Área de Trabalho visíveis.

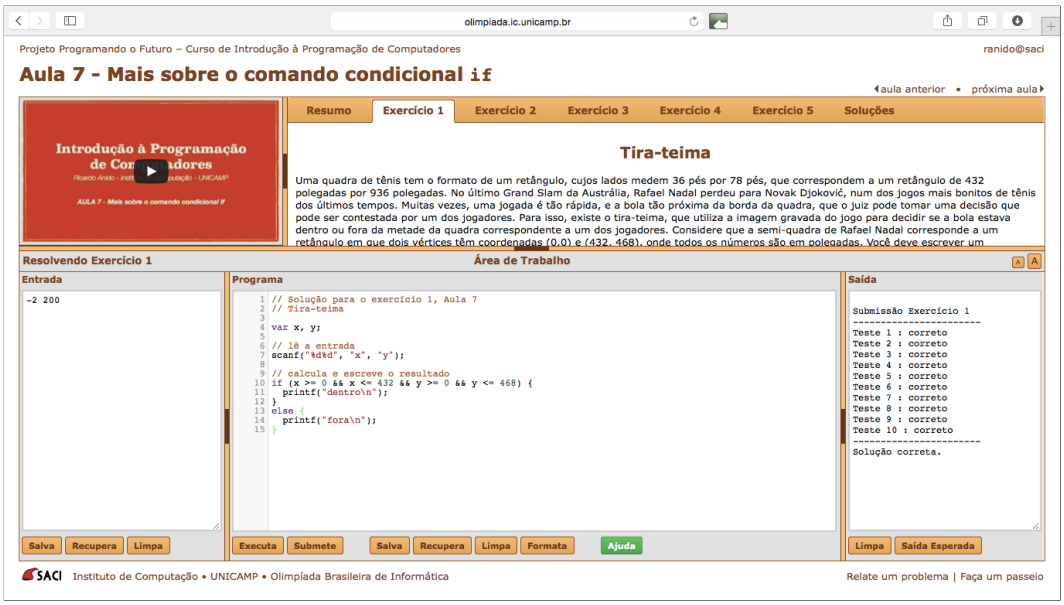
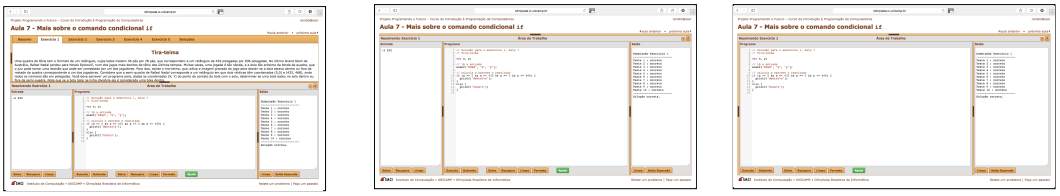


Figura 2. O ambiente Saci.

O usuário pode alterar a configuração de espaço do ambiente ou clicando na parte central das barras de separação (a parte mais escura) ou arrastando as barras de separação para a posição desejada. Clicando na parte central de uma barra de separação o usuário "esconde" uma das duas áreas que a barra separa, expandindo a outra área; clicando novamente na parte central da barra faz a barra voltar à posição anterior, mostrando novamente as duas áreas. A Figura 3 mostra três exemplos de configuração de espaço:

- (a) Área de Vídeo escondida, deixando visível apenas a área de Conteúdo e a Área de Trabalho.
- (b) Área de Conteúdo escondida (nesse caso esconde também a Área de Vídeo), deixando visível apenas a Área de Trabalho.
- (c) Painel de Entrada escondido, deixando visível apenas o Painel de Programa e o Painel de Saída.

Na parte inferior da Área de Trabalho aparecem vários botões que ativam as funcionalidades do ambiente:



(a) (b) (c)

Figura 3. Reconfigurando a divisão de espaço do ambiente.

Botões do Painel de Programa

- Executa: executa o programa, usando como entrada os valores presentes no Painel de Entrada.
- Submete: verifica se o programa está correto. Executa o programa com vários casos de testes escondidos, e compara com a saída esperada.
- Salva: salva o programa no servidor, para uso posterior (funcionalidade restrita a usuários registrados).
- Recupera: recupera um programa salvo anteriormente do servidor (funcionalidade restrita a usuários registrados).
- Limpa: remove todo o conteúdo do Painel de Programa.
- Formata: formata o trecho selecionado de programa.
- Ajuda: dá dicas sobre a resolução de um problema.

Botões do Painel de Entrada

- Salva: salva o conteúdo do Painel de Entrada no servidor, para uso posterior (funcionalidade restrita a usuários registrados).
- Recupera: recupera uma entrada salva anteriormente do servidor (funcionalidade restrita a usuários registrados).
- Limpa: remove todo o conteúdo do Painel de Entrada.

Botões do Painel de Saída

- Limpa: remove todo o conteúdo do Painel de Saída.
- Saída Esperada: mostra a saída esperada usando como entrada os valores presentes no Painel de Entrada (executa a solução padrão, fornecida pelo professor para cada exercício).

2.1. Entrada e Saída

Para realizar entrada e saída em seu programa, o usuário deve utilizar as funções `printf` e `scanf`, desenvolvidas especialmente para o ambiente Saci. As duas funções são similares às funções homônimas da linguagem C, tendo o formato geral igual às funções em C:

`printf("`*cadeia_formatadora*`",` *lista_de_expressões*`)`

`scanf("`*cadeia_formatadora*`",`*lista_de_variáveis*`)`

As cadeias formatadoras dos dois comandos podem usar os mesmos especificadores de formato que na linguagem C: `%d` para valores inteiros, `%f` para valores reais, `%s` para cadeias de caracteres, por exemplo. A principal diferença é que no comando `scanf` os nomes das variáveis da *lista_de_variáveis* devem ser colocados entre aspas, e as variáveis devem ter escopo global.

A implementação dos comandos `printf` e `scanf` usam cursores de posição internos, que são atualizados a cada invocação, de forma que chamadas sucessivas de `scanf` lêem valores em sequência no Painel de Entrada, e chamadas sucessivas de `printf` escrevem em sequência no Painel de Saída.

2.2. O botão Ajuda

Programação de computadores é reconhecidamente uma atividade difícil, especialmente nos estágios iniciais, por exigir uma abstração de raciocínio diferente da qual os alunos estão acostumados. Para amenizar essas dificuldades o ambiente Saci permite que sejam incorporadas *Dicas de Ajuda* aos exercícios. As dicas são acessadas através do botão Ajuda da Área de Trabalho. Considerando que todo o aplicativo deve executar no navegador, e que o código deve ser relativamente enxuto para que seu carregamento pela Internet não demore demasiadamente, optou-se por um mecanismo simples de ajuda, ao invés de abordagens mais complexas [Gomes et al. 2011].

Para ativar a funcionalidade, professor deve fornecer um arquivo de ajuda, composto de um arquivo texto contendo uma sequência de *expressões regulares* e de *parágrafos*. A expressão regular deve ter o formato especificado para o comando *RegExp* de Javascript. Um parágrafo é uma "dica" que vai aparecer em uma janela de notificação (que fica visível no canto inferior direito do ambiente). Expressões regulares e parágrafos são separados por duas quebras de linha consecutivas. Uma expressão regular sempre inicia com o caractere '@', que serve apenas como marcador e é descartado no processamento. Cada expressão regular deve ser seguida de ao menos um parágrafo.

Quando o botão *Ajuda* é pressionado, o sistema procura casar uma expressão regular com o texto corrente do programa do aluno. O sistema busca as expressões regulares na ordem em que foram declaradas no arquivo de ajuda. Comentários no programa são desconsiderados no casamento.

A primeira expressão regular que casar com o texto do programa é aplicada. O sistema então vai mostrar as dicas contidas nos parágrafos subsequentes, até a próxima expressão regular.

As dicas são mostradas em uma janela do tipo *notificação*, que flutua acima da página, no canto inferior direito. Apenas um parágrafo é mostrado na janela de notificação. O usuário deve clicar no botão "próximo" da janela de notificação para ver a próxima dica, que é então mostrada em uma outra janela de notificação, com a janela de notificação anterior sendo fechada.

Para exercícios muito simples, com poucas variáveis e comandos, é possível guiar o estudante desde o primeiro comando até o programa completo. Para exercícios mais complexos o uso de expressões regulares se torna impossível, pela explosão no número de estados possíveis. Nesses casos, o botão de Ajuda pode ser utilizado para dicas de

mais alto nível, sobre o algoritmo, como por exemplo sugerir a melhor forma de pensar na resolução do problema.

```
@(var\s+pontos\s*;)\s*(pontos\s*=\s*10\s*;)\s*(printf\s*\(\s*\"%d\\n\"\s*,\s*.*\))
Muito bem, está quase certo. Vejo que já declarou a variável, escreveu
o comando de atribuição e o comando printf com a
cadeia formatadora na forma correta. Mas a expressão do
comando printf parece não estar correta. Leia novamente o enunciado e
veja qual deve ser a expressão escrita pelo comando printf.

@var\s*\w+\s*;

Hmm, vejo que já declarou uma variável, mas o nome está diferente do
especificado. Leia novamente o enunciado e corrija o nome da variável.

@.*

Você deve inicialmente declarar uma variável de nome pontos.
```

Figura 4. Exemplo de arquivo de configuração de ajuda

Um exemplo de arquivo de ajuda é mostrado na Figura 4.

2.3. Instrumentação para análise de padrões de uso

O sistema está instrumentalizado para possibilitar a análise de alguns parâmetros de uso. Embora a principal instrumentação instalada seja a do Google Analytics, alguns parâmetros podem ser coletados adicionalmente (como a frequência de uso dos botões Executa, Submete e Ajuda). Parâmetros interessantes para a análise da efetividade da ferramenta e que podem ser obtidos com o Google Analytics são por exemplo quais páginas são mais acessadas e tempo médio que o usuário permanece em cada página.

3. Um primeiro curso no ambiente Saci

Um primeiro curso utilizando o ambiente Saci foi desenvolvido, e disponibilizado no meio do mês de março de 2015 com quatorze aulas. O curso é de introdução à programação de computadores usando a linguagem Javascript. O curso usa a abordagem de *resolução de problemas* – em que a introdução de um tópico é feita sempre usando um problema "real". Na parte prática, os estudantes devem ler o enunciado, entender a descição do problema, modelá-lo e encontrar uma solução algorítmica para resolvê-lo, exercitando o material apresentado na aula.

A abordagem de uso de uma história real para o enunciado de problemas é comumente usada em competições de programação, como a OBI e a Maratona de Programação da SBC, e nesse contexto tem sido proposta como um novo método de ensino de programação, conhecido como *Competitive Learning* [Verdú et al. 2011].

Em três meses, esse curso inicial atingiu mais de 2900 alunos registrados, e muitos outros usuários têm utilizado o sistema sem se registrarem. Apesar de ainda ser cedo para fazer uma avaliação tanto do curso como do ambiente desenvolvido, as primeiras impressões são bastante positivas. A Tabela 1 mostra algumas informações sobre o uso do ambiente, com o número de ações realizadas pelos usuários.

Ação	*Quantidade*
Carregar página	13.723
Executar	35.087
Submeter	10.253
Ajuda	2.254

Tabela 1: Dados de uso do ambiente Saci

Mais recentemente foi introduzida também uma funcionalidade para permitir que os usuários possam avaliar cada atividade, atribuindo de uma a cinco estrelas. Como o sistema não obriga o usuário a realizar a avaliação, o número de avaliações é relativamente pequeno, mas ainda assim os resultados são animadores. A Tabela 2 mostra a distribuição das avaliações até o momento:

Avaliação	*1*	*2*	*3*	*4*	*5*
Quantidade	41	16	38	52	331

Tabela 2: Avaliação do Curso de introdução à programação de computadores

4. Trabalhos futuros

Entre os trabalhos futuros previstos estão a continuidade do curso de Javascript, para tópicos mais avançados, como algoritmos básicos em grafos e programação dinâmica, além de um outro curso introdutório utilizando Python.

Também pretendemos investigar uma maneira de medir mais objetivamente o resultado do aprendizado, e da utilização das ferramentas disponíveis (Ajuda, Saída Esperada) na construção da solução pelos alunos. Em outra dimensão, pretendemos também investigar técnicas motivacionais e abordagens pedagógicas para inclusão em futuras aulas.

Uma outra extensão planejada é a abertura do sistema para que qualquer professor possa criar cursos, aulas ou exercícios adicionais, no estilo do que já oferecem *URI OnLine Judge Academics* [Tonin et al. 2013] e *CodeChef for Schools* [CodeChef 2015]. Alguns poucos professores já estão testando a criação de aulas extras, mas o sistema ainda não está totalmente preparado para disponibilização para uso geral.

Referências Bibliográficas

Abelson, H. and Friedman, M. (2010). App Inventor – a view into learning about computers through building mobile applications. In *Proceedings of the 2010 ACM SIGCSE Symposium*.

Begel, A. (1996). *LogoBlocks: a graphical programming language for interacting with the world*. Electrical Engineering and Computer Science Department, MIT, Boston, MA. Retrieved from http://research.microsoft.com/en-us/um/people/abegel/mit/begel-aup.pdf.

Chaves, J. M. O., Castro, F. A., Rommel, W. L., A., L. M. V., and Ferreira, K. H. A. (2013). Mojo: Uma ferramenta de auxílio à elaboração, submissão e correção de

atividades em disciplinas de programação. In *Proceedings of the XXI WEI – Workshop sobre Educação em Informática*, pages 402–407.

CodeChef (2015). Codechef for schools. http://www.codechef.com/school.

CodeOrg (2015). Codeorg.com. http://code.org.

Gomes, C. C., S., L. D. H., Ribeiro, R. P., Almeida, E. S., and Brito, P. H. S. (2011). Uma proposta para auxiliar alunos e professores no ensino de programação: O ambiente aiip. In *Proceedings of the XIX WEI – Workshop sobre Educação em Informática*.

Gray, J., Abelson, H., Wolber, D., and Friend, M. (2012). Teaching cs principles with app inventor. In *Proceedings of the 50th ACM Annual Southeast Regional Conference*.

Haverbeke, M. (2015). Codemirror. http://codemirror.net.

jQWidgets (2015). jqwidgets. http://jqwidgets.com.

Kelleher, C. and Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2):83–137.

KhanAcademy (2015). Khan academy computer science. http://www.khanacademy.org/computing/computer-programming.

Maloney, J., Peppler, K., Kafai, B., Y., Resnick, M., , and Rusk, N. (2008). Programming by choice: urban youth learning programming with Scratch. *ACM SIGCSE Bulletin*, 40(1):367–371.

Noschang, L. F., Pelz, F., Jesus, E. A. J., and Raabe, A. (2014). Portugol studio: Uma ide para iniciantes em programação. In *Proceedings of the XXII WEI - Workshop sobre Educação em Computação*, pages 1287–1296.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11):60–67.

Tonin, N. A., Ferreira, C. E., and Bez, J. L. (2013). Uri online judge academic: A tool for professors. In *Proceedings of the International Conference on Advanced ICT*, volume 1, pages 763–766.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., Castro, J. P., and Queirós, R. (2011). A distributed system for learning programming on-line. *Computers & Education*, 58:1–10.