Ferramentas de apoio ao aprendizadode programação na FAFICA

Márcia Valéria Rocha de Souza[1], César França[2], Walquíria Lins[2]

[1]FAFICA – Faculdade de Filosofia Ciências e Letras de Caruaru
R. Azevedo Coutinho – Petrópolis – Caruaru – PE - CEP 55030-240

[2]C.E.S.A.R. – Centro de Estudos em Sistemas Avançados do Recife
R. Bione, 200, Cais do Apolo – Bairro do Recife – Recife – PE – 50.030-390

marciavr.souza@gmail.com, {franca, wcbl}@cesar.org.br

Abstract.*In order to contribute to the learning process of programming students, we have initially conducted an evaluation of learning programmingtools that could be used in the classroom. Then, we conducted a survey study with students from programming classes at FAFICA, and exposedthe learning problems in terms of programming concepts. Our findings suggest the appropriate tools according to the needs of students.*

Resumo.*Com a finalidade de contribuir para o processo de aprendizagem dos alunos do curso de programação, realizou-se inicialmente uma pesquisa de ferramentas que poderiam ser usadas na sala de aula. Um segundo trabalho, após aplicação de questionário na instituição com alunos do primeiro ao quarto período,expôsos maiores problemasde aprendizagem enfrentados pelos alunos.Sugere-se a utilização de ferramentas para adequar o ensino às necessidades de aprendizagem dos alunos*

1. Introdução

O presente artigo tem como objetivo apontar dificuldades na aprendizagem de programação dos cursos de Análise e Desenvolvimento de Sistemas, e propor soluções com o auxílio de ferramentas educacionais. Espera-se que este estudo possa contribuir para professores e coordenadores adequarem a metodologia de ensino às necessidades de aprendizagem dos alunos desta instituição.

Este artigo é resultado de um trabalho de iniciação científica, e está organizado da seguinte forma: a seção 2 apresenta a contextualização referente à todo o trabalho realizado pelos pesquisadores, desde a classificação das ferramentas até a comprovação das dificuldades dos alunos desta instituição; a seção 3 trata-se da forma como foram coletados os dados para a pesquisa; na seção 4 e 5 são apresentados os resultados das coletas dos dados e a discussão destes. Por fim, na seção 6 estão as conclusões deste trabalho.

2. Estudo das Ferramentas de auxílio à aprendizagem de programação

Pesquisas apontam a falta de motivação e uma dificuldade de aprendizagem por parte dos alunos iniciantes do curso de computação. Esta dificuldade deve-se a uma forte carga de conceitos abstratos que permeiam todo o conhecimento em programação. O desinteresse pode acontecer por dificuldades de interpretação, ou devido aos alunos se sentirem demasiados ansiosos para começar a codificar, sem antes compreenderem os dados do problema, dificultando a fase seguinte, a construção do algoritmo, que seria caracterizado pelo déficit na capacidade de resolução de problemas (Gomes, Henriques

e Mendes [2008]). A forma como o aprendizado de programação é desenvolvido implica na competência do programador profissional, por isso as disciplinas que se propõe a ensinar lógica de programação devem dispor de ferramentas confiáveis e que proporcionem uma aprendizagem efetiva [Souza e França 2013]. Procedemos, então, a uma pesquisa na internet com 9(nove) ferramentas gratuitas que pudessem contribuir para o ensino em sala de aula.

Tais ferramentas foram mapeadas de acordo com asseguintes características básicas: plataforma, idioma, licença, preço e forma de interação (visual ou textual) adotada pela ferramenta. Foram divididas segundo conceitos básicos de programação abordados pela ferramenta, incluindo: compilação, alocação de memória, alocação vetorial, tipos de dados abstratos, operações aritméticas e lógicas, estruturas de controle de fluxo, funções, recursão e depuração representados na Tabela1.

Tabela 1–Conceitos trabalhados nas ferramentas avaliadas (vide Souza e França [2013])

	Compilação	Alocação de memória (variáveis)	Alocação vetorial (arrays)	Tipos de dados abstratos	Operações Aritméticas	Operações Lógicas	Estr. de Controle de fluxo	Funções	Recursão	Depuração passo-a-passo
CodeMonster	*Não*	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	*Não*
FutCode	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	*Não*
Guido VanRobot	*Não*	*Não*	*Não*	*Não*	*Não*	**SIM**	**SIM**	**SIM**	*Não*	**SIM**
KidsRuby	*Não*	**SIM**	*Não*	*Não*	**SIM**	**SIM**	**SIM**	*Não*	*Não*	*Não*
RobotProg	*Não*	**SIM**	*Não*	*Não*	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**
TBC-AED	*Não*	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	*Não*	**SIM**
Visual Alg	*Não*	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**
Web Portugol	*Não*	**SIM**	**SIM**	*Não*	**SIM**	**SIM**	**SIM**	*Não*	*Não*	**SIM**

De acordo com Gomes, Henriques e Mendes [2008] o sistema de aprendizado poderá ser estruturado em três fases: (1) resolução de problemas de diversos domínios não utilizando algoritmos ou programação, em seguida (2) uma amostra da utilidade da programação com aplicação dos conhecimentos adquiridos na fase anterior e, finalmente, (3) passando para a construção dos algoritmos e a aplicação em problemas reais. Considerando as ferramentas pesquisadas e a divisão dos conceitos citados, pode-se ter uma noção quanto ao nível de indicação.

As ferramentas foram catalogadas em três níveis: iniciantes, intermediário e avançado. Nível iniciante trabalha os conceitos básicos e simples utilizando uma forma de interação visual enfatizando a construção de algoritmos. As ferramentas de nível intermediário priorizam os conceitos de programação como alocação de memória em variáveis e vetores, tipo abstrato de dados e funções. Trabalham com linguagens de programação reais esclarecendo a utilidade da programação. Já as de nível avançado são aquelas que abordam conceitos como compilação e recursão e ainda permitem ao programador iniciante o exercício amplo da solução de problemas em domínios reais.A finalidade deste agrupamento é uma possível aplicação em sala de aula, salientando que ao oferecer um ambiente lúdico para o aluno, estimula o interesse pela disciplina fazendo com que ele aprenda de forma gradual. É necessário, portanto, identificar as principais dificuldades dos alunos.

Tabela 2–Competências trabalhadas (vide Gomes, Henriques e Mendes [2008] citado por Souza e França [2013])

	(1) resolução de problemas de domínios diversos	(2) utilidade da programação	(3) construção dos algoritmos
CodeMonster	*Não*	**SIM**	**SIM**
FutCode	**SIM**	**SIM**	**SIM**
Guido VanRobot	*Não*	*Não*	**SIM**
KidsRuby	*Não*	**SIM**	**SIM**
RobotProg	*Não*	**SIM**	**SIM**
TBC-AED	*Não*	**SIM**	**SIM**
Visual Alg	*Não*	*Não*	**SIM**
Web Portugol	*Não*	*Não*	**SIM**

3. Estudo das dificuldades dos alunos do curso de Analise e Desenvolvimento de Sistemas – FAFICA/Caruaru-PE

Uma vez mapeados os conceitos e competências trabalhadas pelas ferramentas disponíveis, estes projeto de iniciação científica focou em realizar um estudo das dificuldades dos alunos matriculados do primeiro ao quarto período do curso de Analise e Desenvolvimento de Sistemas da Faculdade de Filosofia, Ciências e Letras de Caruaru. Ao fim do primeiro semestre acadêmico de 2013, aplicou-se um questionário presencial constituído em três partes: (i) dados demográficos, (ii) estudo de programação e (iii) auto avaliação.

Na primeira parte coletou-se informações referentes aos dados populacionais dos alunos, entre elas, a distribuição destes por município. A instituição possui estudantes de cidades circunvizinhas que viajam diariamente para frequentar as aulas. Visto que o deslocamento de uma cidade a outra por repetidos dias gera o cansaço, este também pode ser um fator de desmotivação e, consequentemente, de déficitno aprendizado.

A segunda parte visa avaliar o interesse do aluno em relação aos estudos e o conhecimento das principais dificuldades no aprendizado em programação. A terceira parte do questionário propõe uma auto avaliação do aluno sobre alguns conceitos iniciais em linguagem de programação como compilação, alocação de variáveis e vetores, tipo abstrato de dados, operações de aritmética e lógica, estrutura de controle de fluxo, funções e métodos, ponteiros e referências, recursão e depuração. Os dados foram coletados entre os dias 06 e 10 do mês de maio no ano de 2013. Dos cento e noventa e oito alunos matriculados, por motivos de faltas, só foram contabilizados 137 questionários respondidos, correspondendo a aproximadamente 70% dos alunos matriculados. Foram aplicados, respectivamente, para duas turmas do 1° e do 3° Período e uma turma do 2° e 4° Período nas disciplinas Programação I, Programação II, Programação Orientada a Objetos II e Programação Web, na devida ordem, nos 1°, 2°, 3° e 4° períodos.

4. Resultados

Um dos primeiros pontos observados com o resultado dos questionários foi a evasão. Com o passar dos períodos, o número de alunos é reduzido como é observado na tabela 3.Gomes, Henriques e Mendes [2008] comentam que o desinteresse pode acontecer por dificuldades de interpretação ou devido os alunos se sentirem demasiados ansiosos para começar a codificar, sem antes compreender os dados do problema. Isto

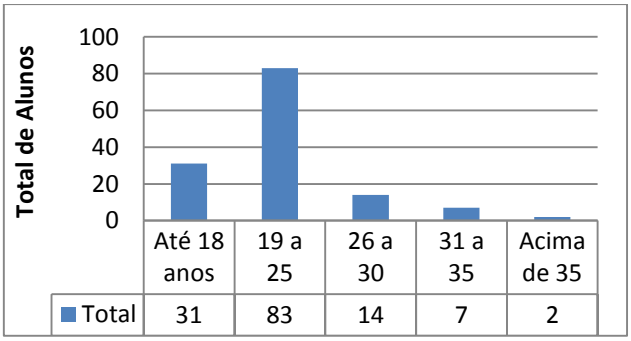
traz dificuldades para a construção do algoritmo que seria caracterizado pelo déficit de alunos na capacidade de resolução de problemas. Barros, Delgado e Machion [2004], comentam que a dificuldade em empregar o raciocínio lógico também gera um ambiente desmotivante, causando uma grande evasão e também grande reprovação na disciplina.

Tabela3–Número de alunos por período

Período	Número de Alunos
1°	62
2°	19
3°	28
4°	28
Total Geral	**137**

Fatores como o baixo nível de abstração, a falta de competências de resolução de problemas, a inadequação dos métodos pedagógicos aos estilos de aprendizagem dos alunos são apontados como fundamentais para essa desmotivação e evasão escolar (JENKINS, 2002). Além disso, as linguagens de programação possuem sintaxes adequadas para profissionais, mas não para aprendizes inexperientes. Contabilizamos 60,6% de alunos - entre 19 a 25 anos- que não trabalham podendo ser considerados pelo autor como aprendizes. O número de estudantes atuantes no mercado profissional são apenas 7 (sete), sendo quatro estudantes acima dos 26 anos.Dois alunos para cada período, entre os 2°, 3° e 4° períodos, sendo apenas um no 1° período. Entre as dificuldades citadas pelos experientes estão o método de ensino e o tempo para dedicar aos estudos. Um aluno registrou dificuldades no aprendizado com relação à insuficiência de material para estudo e compreensão de linguagens de baixo nível. Foi ressaltado que a biblioteca da instituição possui um acervo bem vasto para o estudo em linguagens de programação e é eminente a amplitude de materiais relativos ao tema encontrados na internet que se tornou um grande aliado a pesquisa. Por esse motivo, dificuldades registradas pelos alunos relativas à falta de material para estudo não serão relevantes.

Tabela 4 – Relação de alunos por idade



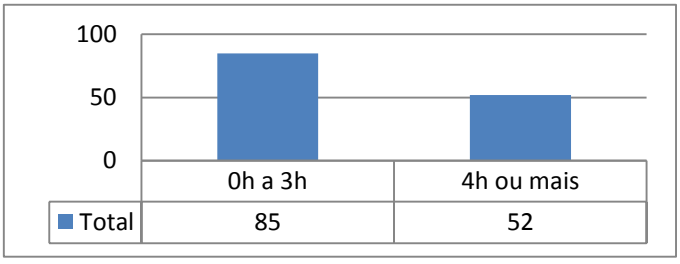
Mais de 80% dos alunos com até 18 anos estão no 1° período da instituição, e as principais dificuldades estão relacionadas com a lógica de programação seguido da metodologia de ensino. Segundo Gomes, Henriques e Mendes [2008], a um nível mais básico, o ensino das linguagens de programação tem como propósito conseguir que os alunos desenvolvam as suas capacidades, adquirindo os conhecimentos básicos necessários para conceber programas capazes de resolver problemas reais simples. Em compensação para que o estudante possa passar para um nível de entendimento avançado, faz-se necessário que ele assimile muito bem os conceitos básicos. Sendo assim, somente após o aprendizado dos conceitos de algoritmos e fundamentos de

lógica, o estudante pode travar contato com uma linguagem de programação concreta para experimentar os conceitos [Santos e Costa 2006]. Almeida, Castro e Castro [2006] afirmam que um dos principais componentes na aprendizagem de programação é a organização das habilidades para a resolução de problemas, elemento comum à construção de conhecimento em qualquer outro domínio, daí a necessidade de investigar como tais habilidades são construídas. Isto requer identificar e entender as dificuldades encontradas pelos estudantes de programação. Porém, em uma sala de aula, o nível de conhecimento e assimilação dos conteúdos passados é diferente de aluno para aluno o que sugere um ensino individualizado. No entanto, Falckembach e Araújo [2005], discutem que esse tipo de ensino, personalizado e individualizado, entra em choque com o pressuposto básico da educação tradicional que é a padronização.

A lógica e a metodologia aplicada estão entre os principais problemas citados pelos alunos da faixa etária dos 19 aos 25 anos, seguido do tempo para a dedicação aos estudos e o conhecimento na língua inglesa. Souza e França [2013] comentam que as dificuldades podem estar associadas tanto à forma de ensino quanto à ansiedade do aluno em querer ver os resultados práticos de seus códigos, como também, na compreensão e na aplicação de noções básicas para criação de algoritmos. Segundo Falckembach e Araújo [2005] a forma de ensino dentro da sala de aula é a mesma para todos os alunos, pois, é extremamente difícil para um docente levar em consideração o perfil, as metas, as necessidades, as expectativas, as preferências a primeira linguagem de contato.

Com o propósito de identificar a familiaridade dos alunos com a programação, antes da graduação, foi questionada com qual idade e qual a primeira linguagem de programaçãoque tiveram contato. 81 alunos tiveram como primeira linguagem o Java, seguido da linguagem C. Apenas um aluno respondeu que nunca teve contato com nenhuma linguagem de programação, e sete deles afirmam que não se dedicam à prática. Em relação aos alunos que já trabalham na área, e o tempo de dedicação, quatro praticam de 1 a 3h, dois praticam de 7 a 9h e apenas um dedica 10h ou mais de práticas em programação. Dos 137 alunos que responderam a pesquisa, 85 deles praticam programação entre zero a três horas por semana, e um grupo de 52 alunos praticam mais de 4 horas. De acordo com o ranking do site Tiobe [2013] a linguagem Java aparece em 2° lugar por dois anos consecutivos entre as mais procuradas. A linguagem Java é oferecida pela instituição nas disciplinas de Programação e Programação Orientadas a Objetos, portanto, existe uma preocupação em preparar os alunos para o mercado de trabalho com uma linguagem atual. Quanto ao número de horas por semana dedicadas ao estudo de programação, mesmo sabendo que depende da disponibilidade e interesse de cada aluno, é recomendável destinar um tempo para sua prática

Tabela 5 – Horas semanais praticadas pelos alunos



O conhecimento dos alunos em outras linguagens de programação foi registrado no questionário. Dentre as linguagens mais citadas está o Java, por ser, também, a linguagem trabalhada em sala de aula, o HTML/CSS, o C/C++ e o Javascript. Das linguagens que receberam menos de dez votos estão o Ruby e o ActionScript, com 3 votos cada, o VBNet e o Cobol/Fortran com 2 votos cada, o Objective C e o Prolog com 1 voto cada.O o Perl e Lisp/Haskel não receberam nenhum voto. Portugol, ShellScript, Visualg e ADVPL, cada um com dois votos, e o Progress 4gl, Basic, Natural Clipper, Lua, Assembly e Grails com um voto cada, foram citadas, mesmo sem constar na relação oferecida para as suas escolhas.

Na terceira e última etapa do questionário, auto avaliação, os estudantes responderam sobre o grau de conhecimento em noções básicas de programação. As análises dos dados foram feitas por período, pois alguns dos conceitos começam a ser estudados a partir do segundo período. Entre os índices registrados de dificuldades dos alunos, estão os conceitos de alocação vetorial, tipo abstrato de dados, funções, métodos, ponteiros, referências, recursão e depuração. Estão apresentados, neste tópico, apenas os conceitos que foram registrados fora do domínio dos alunos.

Vetores é um tipo de array unidimensional. A alocação deste elemento consiste em uma série de elementos do mesmo tamanho e tipo. Seu acesso é feito pela posição do elemento. No primeiro período, 50% dos alunos que responderam o questionário, nunca ouviram falar em vetores. Como já foi comentado, alguns conceitos ainda não foram passados para os primeiros períodos, sendo este déficitjustificável. Entre os conceitos ainda não vistos no primeiro período estão: Tipo Abstrato de Dados, Funções e Métodos, Ponteiros, Referências e Recursão. O conceito de vetores é aplicado a partir do segundo período. Os níveis de conhecimento sobre este conceito, do segundo ao quarto período, estão acima dos 50% considerados satisfatórios.

A ideia de abstração de dados refere-se a uma modelagem de uma estrutura de dados de acordo com seu funcionamento. No segundo período, 68,42% dos estudantes conhecem o conceito, mas desconhece a forma de aplicação. Menos da metade dos alunos do terceiro período dominam o conceito. No quarto período houve um empate entre os que também sabem do conceito, mas não sabem da aplicação e dos que dominam o assunto.

Ponteiros, Referências, Recursão e Depuração são os conceitos de mais dificuldades entre os alunos. A depuração, apesar de ter sido discutida em sala de aula no primeiro período, 38,70% dos alunos nunca ouviram falar. No segundo período, a maioria, 47,36%, sabe do que se trata, porém não sabem usar. No terceiro período, houve um empate entre os que conhecem, mas não sabem da funcionalidade e os que dominam o conceito. O mais alarmante é o conceito de recursão, pois a maioria de todos os períodos nunca ouviu falar. Sobre conceito de ponteiros e referências, do segundo ao quarto período, a maioria respondeu que sabe do que se trata, mas não sabe usar.

Tabela 6 – Porcentagem dos estudantes por período letivo que dominam os conceitos citados

Período	Nº Alunos	Compilação	Alocação de variáveis	Alocação vetorial	Tipos de dados abstratos	Operações Aritméticas	Operações Lógicas	Estr. de Controle de fluxo	Funções e Métodos	Ponteiros e Referências	Recursão	Depuração
1°	62	54,88%	45,16%	1,61%	6,45%	66,12%	30,64%	35,48%	9,67%	6,45%	1,61%	20,96%
2°	19	84,21%	94,73%	78,94%	15,78%	84,21%	47,36%	100%	52,63%	21,05%	5,26%	10,52%
3°	28	71,42%	82,14%	60,71%	46,42%	96,42%	67,85%	92,85%	64,28%	25,00%	10,71%	35,71%
4°	28	82,14%	78,57%	46,42%	39,28%	100%	78,57%	85,71%	71,42%	32,14%	21,42%	39,28%

5. Interpretação

De acordo com a última parte do questionário, destinado à avaliação da compreensão dos conceitos, numa escala de 1 a 5, entende-se que o domínio seria a completa compreensão do conceito, e os demais que existe um déficit a ser resolvido. Analisando estas dificuldades dos alunos representadas neste artigo, como a referenciada tabela 1, já podemos identificar quais das ferramentas, estudadas anteriormente, poderão ser aplicadas. O primeiro dos conceitos trabalhados segundo as competências é o de compilação, apesar de mais de 50% da turma do primeiro período dominar o conceito, uma parcela considerável de 28 alunos sente dificuldade entre a usabilidade e a utilidade na aplicação. Das ferramentas citadas na tabela 1, apenas o FutCode trabalha este conceito, porém como no FutCode também é aplicado conceitos ainda não estudados pelas turmas do primeiro período, convém analisar uma forma de desta ferramenta contribuir na aprendizagem.

Menos da metade dos alunos dominam o conceito de alocação de varáveis no primeiro período, 34 alunos, num total de 62, responderam que possuem algumas dificuldades neste conceito. Todas as ferramentas citadas na tabela 1, com exceção do Guido VanRobot, trabalham o conceito de alocação de variáveis, porém o TBC-AED trabalha conceitos mais avançados e não possuem uma versão para iniciantes. O CodeMonster e o KidsRuby utilizam as linguagens Javascript e Ruby, respectivamente, o que seria desaconselhável para alunos do primeiro período que ainda não tiveram nenhum contato com linguagens de programação, e estão sendo iniciados com o Java. As demais turmas apresentaram um nível satisfatório de compreensão do conceito de alocação de variáveis.

No primeiro período não indicaremos ferramentas para alocação vetorial, tipo abstrato de dados, funções e métodos, ponteiros e referências e recursão, apesar do resultado negativo obtido, estes conceitos só serão cobrados para os alunos no período seguinte. Nos terceiro e quarto período, onze e quinze alunos, respectivamente, dos 28 entrevistados em cada turma, responderam que sentem alguma dificuldade em alocação vetorial. Entre as ferramentas que podem ser aplicadas para este conceito segundo a Tabela 1, estão: CodeMonster, FutCode, TBC-AED, VisuAlg e Web Portugol. No conceito de tipo abstrato de dados, observa-se claramente que existe incompreensão do segundo ao quarto período, ou seja, mais da metade dos que responderam o questionário afirmaram que sentem algum bloqueio neste conceito. As ferramentas que podem ser indicadas para servir como apoio são: CodeMonster, FutCode, TBC-AED e VisuAlg.

Sobre o conceito de operações aritméticas, há uma distribuição ínfima do segundo ao quarto período de alunos com bloqueio neste conceito. Já no primeiro período, 21 alunos dos 62 entrevistados possuem dificuldades. Em operações lógicas, mais da metade no primeiro e segundo períodos possuem dificuldades; no terceiro e

quarto períodos somam 15 alunos. Já no conceito de controle de fluxo, o primeiro período tem mais dificuldades, 40 alunos dos 62 entrevistados afirmam que sentem alguma dificuldade neste conceito. Nos terceiro e quarto períodos somam 06 alunos que sentem algum bloqueio. Todas as ferramentas citadas na Tabela 1 trabalham estes três conceitos, exceto o Guido VanRobot. Porém, para o primeiro período teríamos que excluir o TBC-AED, CodeMonster e o KidsRuby pelos motivos anteriormente citados.

Tabela 7 – Análise das Habilidades

Descrição das habilidades em programação				
Período	Tem dificuldade na construção de algoritmos simples	Constrói algoritmos mas não visualiza a utilidade	Programa bem e visualiza a utilidade mas tem limitações p/ resolver problemas reais	Programa bem e visualiza a utilidade e é capaz de resolver problemas reais
1° (62)	14,51%	25,80%	50,00%	8,06%
2° (19)	5,26%	26,31%	57,89%	10,52%
3° (28)	7,14%	25,00%	50,00%	17,85%
4° (28)	7,14%	32,14%	42,85%	17,85%

Os conceitos de funções, métodos, ponteiros, referências e recursão, são trabalhados inicialmente no segundo período, onde 9 dos 19 alunos apontaram dificuldades em funções e métodos. No terceiro e quarto períodos, respectivamente 10 e 8 sentem dificuldades. De acordo com a tabela 6 deste artigo, observa-se que os conceitos mais críticos são os de ponteiros, referências e recursão. Do segundo ao quarto período, mais da metade dos estudantes respondeu que sentem dificuldades nestes dois conceitos. O conceito de recursão chega a ser o mais grave: mais de 50% de todas as turmas responderam que nunca ouviu falar. Infelizmente, a nossa tabela, referenciada de um estudo anterior, não avalia as ferramentas com o conceito de ponteiros e referências. As ferramentas CodeMonster, FutCode, RobotProg e VisuAlg são as únicas que trabalham Funções, Métodos e Recursão.

A depuração é um processo de análise de código passo a passo e aplicável desde o primeiro período. Porém, mais de 50% em todas as turmas relataram algum tipo de dificuldade neste conceito. Das ferramentas que podem ser aplicadas estão o Guido VanRobot, RobotProg, TBC-AED, VisuAlg e Web Portugol.

Segundo a referenciada tabela 2, das competências trabalhadas, aponta-se o FutCode como uma ferramenta educacional que possibilita o aluno reconhecer alguns dos principais conceitos em programação. Na tabela 7, estão as análises das habilidades dos alunos, onde eles avaliaram suas competências em programação ficando notável a dificuldade dos alunos em resolver problemas reais. Sugerindo, então, o FutCode ou uma adaptação desta ferramenta para auxiliar na compreensão dos alunos.

6. Discussão

Com a categorização das ferramentas no primeiro trabalho, identificamos três níveis de complexidade. Para iniciantes, onde são enfatizados os conceitos básicos como operações aritméticas, lógicas e estrutura de controle, enfatizando a construção de algoritmos, podendo ser indicada para turmas iniciantes e associamos com o primeiro

período. Entre as ferramentas citadas para o nível iniciante estão o RobotProg e o Guido VanRobot. Asferramentas de nível intermediário foram associadas aosos conceitos de alocação de memória em variáveis e vetores, tipo abstrato de dados e funções. Com o propósito de trabalhar com linguagens de programação reais, passaria a transparecer para o aluno a utilidade da programação. Associamos ao segundo período com as ferramentas Code Monster e KidsRuby. Para o nível avançado, a proposta é fazer com que os alunos visualizem a solução de problemas em domínios reais. Correlacionamos com os terceiro e quarto períodos. O FutCode atende o requisito onde são trabalhados conceitos avançados, como compilação e recursão, este último com o pior índice registrado na instituição.

7. Considerações finais

De acordo com os dados apresentados, salientamos odéficit no aprendizado dos alunos da Faculdade de Filosofia Ciências e Letras de Caruaru no curso de Análise e Desenvolvimento de Sistemas. A proposta desde estudo é contribuir com a instituição em oferecer para os seus alunos um curso com um nível de qualidade de aprendizagem alta. A procura pela graduação em Análise e Desenvolvimento de Sistemas é grande, mas por se tratar de um curso complicado, muitos alunos acabam desistindo como pudemos observar nos números dos matriculados do primeiro ao quarto período. Vários agravantes foram citados, como por exemplo, um aluno chega ao quarto período sem saber dos conceitos básicos de programação como alocação de uma variável e o desconhecimento em relação à conceitos importantes como a recursão. Neste artigo, estão em evidência as principais dificuldades dos alunos e espera-se que com esses dados, coordenadores e professores possam tomar medidas para reduzir essa problemática.

A nossa contribuição é evidenciar a possibilidade de fazer uso de diversas ferramentas disponíveis para melhorar o ensino e a aprendizagem de programação. Ressaltando a queixa da maioria dos entrevistados: é necessário que os docentes possam acompanhar os alunos no processo aprendizagem, identificar as dificuldade e dispor de estratégicas de ensino simples, como as ferramentas apresentadas, que de forma lúdica acabam contribuindo em sala de aula. Um maior engajamento e envolvimento dos alunos, resultantes de uma aprendizagem mais significativa e contextualizada, traria como conseqüência, também, a diminuição da evasão escolar.

Referências

Almeida, N. F. A., Castro, T., Castro, A. N. (2006). “Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação”. http://www.lbd.dcc.ufmg.br/colecoes/sbie/2006/019.pdf, Junho 2013.

Almeida, E. S.; B.; Braga, J. D. H.; Silva, K.; Paes, R. B.; Almeida, A. (2002) “Ambap: Um Ambiente de Apoio ao Aprendizado de Programação”, http://goo.gl/KSNmG, Dezembro 2012.

Barros, L. N., Delgado, K. V., Machion, A. C. G. (2004). “An ITS for programming to explore practical reasoning”. http://goo.gl/z4f50, Junho 2013.

Deitel, H. M.; Deitel, P. J. (2010). “Java Como Programar”. Ed. Pearson Prentice Hall, Junho 2013.

Falckembach, G. A. M., Araújo, F. V. (2005). “Aprendizagem de algoritmos: dificuldades na resolução de problemas”.http://www.fabricioviero.com.br/artigos/a4_siie.pdf, Junho 2013.

Gomes, A., Henriques, J., Mendes, A. J. (2008). "Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores", http://goo.gl/DGLXh, Junho 2013.

Jenkins, T. (2002). "On the difficulty of learning to program", http://www.ics.heacademy.ac.uk/Events/conf2002/tjenkins.pdf, Junho 2013.

Mota, M. P., Brito, S. R., Moreira, M. P. and Favero, E. L. (2009) "Ambiente Integrado à Plataforma Moodle para Apoio ao Desenvolvimento das Habilidades Iniciais de Programação", http://goo.gl/dFHKS. Junho 2013.

Santos, R. P. ; Costa, H. A. X. (2006). "Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos Iniciantes em Computação e Informática." http://goo.gl/RcCjk, Junho 2013.

SBC - Sociedade Brasileira de Computação (2003). Currículo de Referência da Sociedade Brasileira de Computação para Cursos de Graduação em Computação e Informática". http://www.sbc.org.br. Junho 2013.

Souza, M. V. R., França, A. C. C. (2013). "Ferramentas de Auxílio ao Aprendizado de Programação: Um Estudo Comparativo". http://www.erbase2013.itatechjr.com.br/index.php/programacao/weibase. Junho 2013.

Tiobe. (2013). "TIOBE Programming Community Index for july 2013". Disponível em<http://www.tiobe.com/>. Acessado em Julho 2013.