Ambientes de Desenvolvimento Integrado no Apoio ao Ensino da Linguagem de Programação Haskell

Davi Felipe Russi
Curso de Ciência da Computação – Universidade Federal de Santa Maria – davirussi@inf.ufsm.br

Andrea Schwertner Charão
Laboratório de Sistemas de Computação – Universidade Federal de Santa Maria – andrea@inf.ufsm.br

Resumo. *A linguagem Haskell é comumente utilizada em universidades para ensino do paradigma de programação funcional. Neste artigo, defende-se que o uso de um ambiente de desenvolvimento integrado (Integrated Development Environment - IDE) é vantajoso para apoiar o ensino e o aprendizado desta linguagem. Nesta linha, apresenta-se primeiramente uma análise comparativa de IDEs disponíveis para Haskell e, em seguida, relata-se uma experiência realizada utilizando-se o IDE Eclipse em sala de aula. Os resultados contribuem para a adoção deste tipo de ferramenta no ensino de Haskell.*

Palavras-chave: Haskell, ensino, linguagem de programação, ambiente integrado de desenvolvimento

Integrated Development Environments as a Supporting Tool for Teaching the Haskell Programming Language

Abstract. *The Haskell language is commonly used in universities for teaching functional programming. In this article, we argue that the use of an Integrated Development Environment (IDE) is useful for supporting the teaching and learning of the language. In this direction, we present a comparative analysis of Haskell's IDEs and we report an experiment using the Eclipse IDE in the classroom. The results contribute to the adoption of such a tool in the teaching of Haskell.*

Keywords: Haskell, teaching, programming language, integrated development environment

1. Introdução

A linguagem Haskell é uma linguagem de programação de propósito geral que segue o paradigma funcional de programação (Hugues, 1989). Sua principal estrutura de controle é a função, sendo que a linguagem possui suporte a funções recursivas, casamento de padrões, listas por compreensão, guardas, dentre outros recursos (Jones, 2002). Esta linguagem é relativamente recente, desenvolvida desde 1990, e vem sendo cada vez mais utilizada principalmente em universidades, no ensino e pesquisa sobre o paradigma funcional.

Como se trata de uma linguagem bastante diferente daquelas ensinadas no início dos cursos da área de computação (geralmente C ou Java), é natural que muitos alunos

enfrentem dificuldades no aprendizado de Haskell. Além disso, comparativamente com outras linguagens mais populares, Haskell dispõe de um menor conjunto de ferramentas de desenvolvimento, de forma que o primeiro contato com a linguagem é geralmente através de um interpretador de comandos usado em paralelo com algum editor de programas. Para alunos habituados com ambientes de desenvolvimento integrado (do inglês, *Integrated Development Environment* - IDE), este modo de trabalho tende a gerar dificuldades pela falta de integração entre editor e interpretador.

No ensino introdutório de linguagens de programação, o uso de IDEs é geralmente bem aceito por professores e alunos (Pears et al., 2007). De fato, alguns problemas que mais ocorrem quando alguém está iniciando uma linguagem, tais como erros de sintaxe ou mesmo de lógica, podem ser minimizados com a ajuda de IDEs que ofereçam um bom editor integrado ao interpretador/compilador da linguagem e a uma ferramenta de depuração.

O presente trabalho segue nesta linha, abordando o uso de IDEs no ensino da linguagem Haskell, para universitários que já tiveram contato com outra linguagem de programação. Um dos objetivos do trabalho é realizar uma análise comparativa de IDEs para Haskell, reunindo dados que possam ser úteis para quem desejar adotar este tipo de ferramenta. Outro objetivo é realizar um estudo de caso com um desses IDEs, utilizando-o em sala de aula, a fim de coletar impressões sobre esta abordagem.

O artigo está organizado da seguinte maneira: a seção 2 apresenta trabalhos relacionados, discutindo o uso de IDEs no ensino e suas aplicações à linguagem Haskell. Em seguida, a seção 3 apresenta a análise dos IDEs escolhidos, detalhando critérios de análise e resultados obtidos. A seção 4 apresenta o estudo de caso realizado e, por fim, a seção 5 apresenta as considerações finais sobre o trabalho.

2. Trabalhos Relacionados

O ensino de linguagens de programação tem vários desafios, por exigir dos alunos uma alta capacidade de abstração e de atenção a detalhes. Além disso, trata-se de uma área que tradicionalmente requer o uso de software de apoio, tais como editor, interpretador, compilador e depurador. Em ambientes integrados de desenvolvimento, estes recursos estão reunidos de forma a facilitar as atividades do programador.

O uso de IDEs no ensino de linguagens de programação é um assunto já bastante explorado. De forma geral, entende-se que este tipo de ferramenta possa trazer vantagens para o aluno que está iniciando o aprendizado de uma nova linguagem de programação (Pears et al., 2007; Caspersen, 2007). Neste sentido, existem muitos ambientes especialmente concebidos para o ensino de alguma linguagem. Este é o caso, por exemplo, dos ambientes BlueJ (Kölling et al., 2003), Greenfoot (Henriksen & Kölling, 2004) e DrJava (Allen, Cartwright & Stoler, 2002) para a linguagem Java. Além disso, IDEs concebidas para uso profissional também podem ser empregadas para fins educacionais, embora não sejam particularmente indicadas para alunos novatos em programação.

No que diz respeito à linguagem Haskell, encontrou-se apenas um trabalho visando à construção de um IDE com propósitos educacionais, chamado HEAT (Ashton et al., 2005). Embora este IDE tenha evoluído desde sua concepção, seus desenvolvedores relatam pequenos problemas no seu funcionamento. No presente trabalho, decidiu-se seguir outra via, abordando IDEs para Haskell que não tenham sido especialmente concebidos para o ensino. Nesta linha, foram encontradas diversas ferramentas, porém não encontrou-se nenhum relato de uso de tais IDEs no ensino de

Haskell, gerando dúvidas sobre qual ferramenta melhor se prestaria tanto para ensino como para uso profissional. Este trabalho, portanto, dedica-se a preencher esta lacuna.

3. IDEs para a Linguagem Haskell

O levantamento de IDEs para Haskell foi realizado por meio de buscas em *sites* Web especializados nesta linguagem. Foram escolhidos 6 (seis) ambientes que tinham maior frequência de citações e que aparentavam estabilidade no desenvolvimento: Leksah, Eclipse (*plugin* EclipseFP), Winhugs, Yi, Vim (*plugin* Haskell) e Emacs (*plugin* Haskell).

Como critérios de análise, foram consideradas diversas características julgadas desejáveis sob o ponto de vista de um usuário que já tem experiência em programação e está iniciando na linguagem Haskell. Tais critérios são detalhados na seção 3.1. A metodologia de análise foi a seguinte: primeiramente, foram consultadas diversas fontes de documentação sobre cada IDE (site oficial, guia de usuário, fórum de discussões e fórum de rastreamento de *bugs*), buscando identificar se as ferramentas satisfaziam ou não as características desejáveis; em seguida, as ferramentas foram instaladas e testadas, a fim de validar e aprofundar as observações iniciais feitas a partir da documentação; por fim, as observações documentadas foram sintetizadas em forma de tabela para futura referência.

3.1. Critérios de Análise

Os seguintes critérios de análise foram definidos em conjunto com um aluno que havia aprendido a linguagem Haskell recentemente, cursando uma disciplina do terceiro semestre de um curso de graduação em Ciência da Computação:

- Instalação – a instalação do IDE deve ser fácil e rápida, para que o usuário possa começar a utilizá-la o mais breve possível.
- Plataformas – é interessante que o IDE funcione nas plataformas de hardware/software mais comuns, incluindo Windows, Linux e MacOS.
- Licença de uso – IDEs distribuídas sob licenças de software livre são preferíveis, por serem mais acessíveis.
- Manual do usuário – é desejável que o manual seja o mais completo possível, para que o usuário possa sanar as dúvidas iniciais sem ter que pesquisar em fóruns.
- Comunidade de usuários – uma comunidade numerosa é um ponto positivo, pois facilita a resolução de problemas que possam surgir.
- Comunidade de desenvolvedores – geralmente, IDEs desenvolvidas por grupos trazem atualizações mais rapidamente que os desenvolvidos por uma só pessoa.
- Interface – a interface deve ser intuitiva, limpa e amigável.
- Compiladores / interpretadores – é importante que o IDE funcione com os compiladores e interpretadores mais populares para a linguagem Haskell (Hugs e/ou GHC).
- Depurador – é desejável que esteja integrado para facilitar a localização de problemas de lógica.
- Editor – um editor poderoso com várias funcionalidades pode ajudar muito, por exemplo com numeração de linhas para localização de erros reportados,

múltiplas abas para acelerar a visualização de vários arquivos, linha atual destacada para facilitar a localização dentro do arquivo, dentre outros recursos considerados abaixo.

- Checagem de sintaxe – é um recurso muito útil, pois avisa quando há algo errado no código antes que o usuário acione o compilador ou interpretador.
- Realce de sintaxe – é um recurso importante, pois mostra o código de forma que fica fácil de visualizar e localizar funções, variáveis, constantes, strings, operadores, entre outros.
- Auto-completar – este recurso auxilia na digitação do código, diminuindo os erros cometidos.
- Ajuda rápida – são dicas mostradas quando o mouse aponta para determinadas regiões do código.
- Endentador[1] – responsável pela formatação do código, ajudando na sua organização e facilitando sua leitura.

3.2. Análise dos IDEs

O IDE Leksah[2] tem uma instalação facilitada pelo instalador e é distribuído sob a licença de software livre GPL (*GNU General Public License*) 2.0. O desenvolvimento do software é dado por uma comunidade muito ativa, mas o projeto é coordenado por uma única pessoa. Leksah é multiplataforma, portanto suportado pelos sistemas operacionais Linux, Windows e MacOS. O ambiente conta com uma interface amigável e possui muitas funcionalidades, tais como depurador, checagem de sintaxe, auto-completar, realce de sintaxe, editor de texto com múltiplas abas, contador de linhas e linha atual destacada, dentre outras. Uma característica única deste IDE é seu sistema automatizado para correção de módulos faltantes, ilustrado na figura 1.

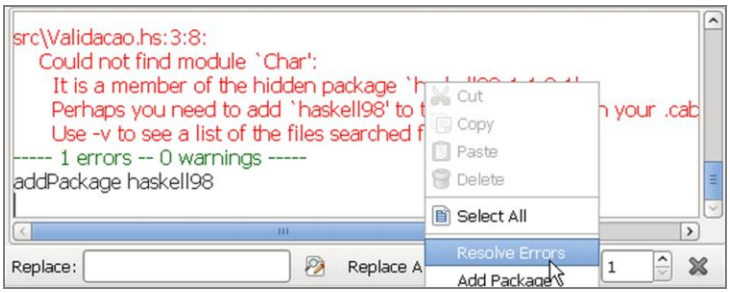

Figura 1 Leksah - Corretor de módulos

O ambiente Eclipse, com o *plugin* EclipseFP[3], tem uma instalação simples pelo instalador de *plugins* do Eclipse e é distribuído sob a licença de software livre EPL 1.0 (*Eclipse Public License*). O desenvolvimento deste *plugin* se dá por um grupo que é coordenado por uma única pessoa. EclipseFP é multiplataforma, uma vez que o Eclipse é suportado pelos sistemas operacionais Linux, Windows e MacOS, tanto em suas versões 32 bits quanto 64 bits. O ambiente conta com uma interface amigável, possui

1 Também conhecido popularmente pelo neologismo "indentador".

2 http://leksah.org

3 http://eclipsefp.sourceforge.net

muitas funcionalidades como depurador, checagem de sintaxe, auto-completar, visualizador de arquivos do projeto, realce de sintaxe, editor de texto com múltiplas abas, contador de linhas, linha atual destacada, dentre outras. Seu recurso de ajuda rápida mostra o tipo da função apontada pelo mouse, conforme ilustra a figura 2.

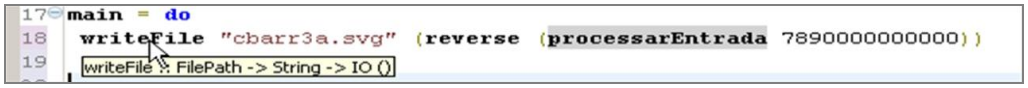

Figura 2 Eclipse - Ajuda rápida

O IDE Winhugs[4] também tem instalação facilitada pelo instalador e é distribuído sob a licença de software livre BSD (*Berkeley Software Distribution*). Seu desenvolvimento se dá por uma única pessoa. Winhugs é basicamente uma interface para o ambiente Hugs no sistema operacional Windows. O ambiente conta com uma interface simples e amigável (ver figura 3), mas possui poucas funcionalidades. Uma delas é a checagem de sintaxe, que em caso de erro indica o nome do arquivo, a linha e o tipo da ocorrência. Como editor de texto, utiliza o *notepad* do Windows.

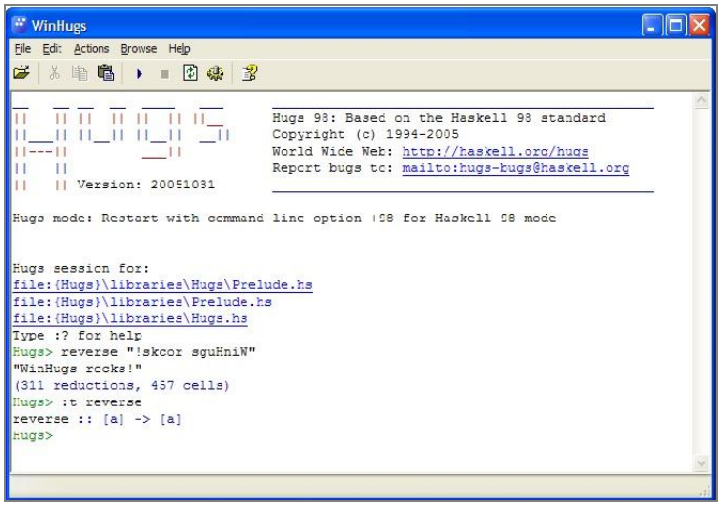

Figura 3- Winhugs Interface simples

O ambiente Yi[5] não passou pela fase de testes, pois não foi possível instalá-lo. No fórum de rastreamento de *bugs* (*bug tracker*) do Yi, foram encontradas várias mensagens sinalizando erros. Um erro de instalação foi solucionado por um usuário e a solução foi compartilhada no fórum, mas não funcionou como esperado. Devido a esta dificuldade, o IDE Yi não foi incluído na análise comparativa que consta neste artigo.

O IDE Vim[6] tem uma instalação facilitada no Linux (comando apt-get), mas seu *plugin* para Haskell revelou-se complicado para um iniciante. Para instalação do *plugin* é necessário obtê-lo usando a ferramenta darcs. Após obtê-lo e instalá-lo, é necessário editar um arquivo de configuração. O desenvolvimento do *plugin* para Haskell é um projeto coordenado por um único desenvolvedor. Vim é multiplataforma, portanto suportado pelos sistemas operacionais Linux, Windows e MacOS. No entanto, é uma

4 http://www.haskell.org/haskellwiki/WinHugs

5 http://www.haskell.org/haskellwiki/Yi

6 http://www.cs.kent.ac.uk/people/staff/cr3/toolbox/haskell/Vim/vim.html

ferramenta mais popular entre usuários de Linux, sendo que usuários de outros sistemas operacionais podem ter dificuldades com as configurações para o correto funcionamento do *plugin*. O ambiente conta com uma interface rudimentar, mas é possível a instalação do GVim para tornar a experiência mais amigável. Apresenta várias funcionalidades tais como checagem de sintaxe (ver figura 4), realce de sintaxe, editor com múltiplas abas, e vários atalhos para facilitar o desenvolvimento, como por exemplo o comando gf (*go to file,* que acelera a navegação entre os módulos).

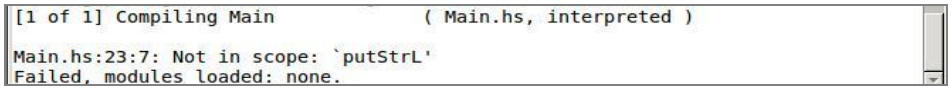

Figura 4 Vim - Checagem de sintaxe

O ambiente Emacs tem uma instalação facilitada no Linux (comando apt-get) e seu *plugin* Haskell[7] também pode ser instalado da mesma forma (apt-get install haskell-mode). O desenvolvimento do *plugin* é feito por um único desenvolvedor. Emacs é multiplataforma, portanto suportado pelos sistemas operacionais Linux e Windows. No Windows, podem ocorrer dificuldades com as configurações para o correto funcionamento do *plugin*. O ambiente conta com uma interface amigável. Algumas de suas funcionalidades são: endentador de código, checagem de sintaxe, realce de sintaxe (ver figura 5) e editor com múltiplas abas.

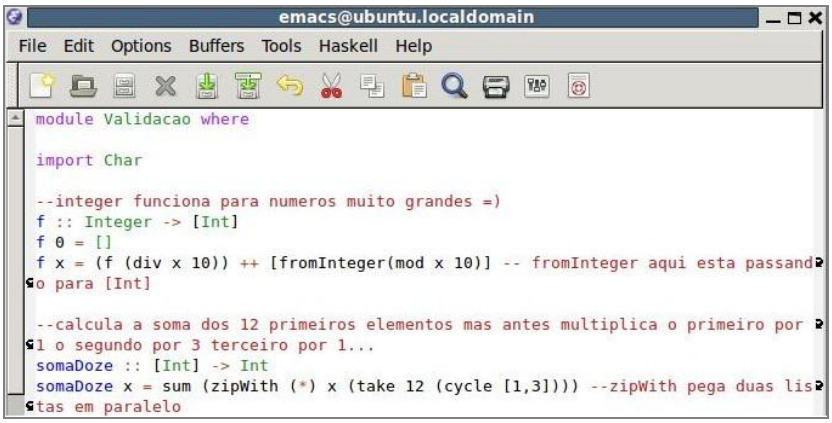

Figura 5 Emacs - Realce de sintaxe

3.3. Comparativo

Após análise dos ambientes, elaborou-se um comparativo resumindo objetivamente os principais recursos das ferramentas. Este comparativo é apresentado no quadro 1.

7 http://projects.haskell.org/haskellmode-emacs/

	Leksah	Eclipse	WinHugs	Vim	Emacs
Instalação	Instalador	Plugin (Install New Software)	Instalador	Plugin (Darcs + Vba + vimrc)	Plugin (apt-get)
Sistema Operacional	Windows Linux Mac	Windows Linux Mac	Windows	Windows Linux Mac	Linux Windows
Compiladores Interpretadores	GHC* GHCi	GHC GHCi Hugs98	Hugs98	GHC GHCi	GHCi Hugs98
Debugger	Sim	Sim	Não	Não	Não
Checagem de Sintaxe	Sim (cores e facilita a adição de pacotes para correção de erros)	Sim (avisa se contem erros a cada vez que o arquivo é salvo)	Sim (cores para os erros)	Sim (sem cores)	Sim (cores para os erros)
Realce de sintaxe	Sim	Sim	Não	Sim	Sim
Auto Completar	Sim	Sim (com mais funcionalidades)	Não	Não	Não
Ajuda Rápida	Não	Sim (mostra o cabeçalho da função)	Não	Não	Não
Múltiplas abas	Sim	Sim	Não	Sim (GVIM)	Sim

Figura 6 Tabela de Comparação entre as IDEs

Como já foi comentado, o IDE Yi foi retirado da comparação devido à impossibilidade de instalá-lo. O recurso de endentação, também considerado como critério de análise, não consta no quadro porque nenhuma ferramenta possui endentador automático. Quanto aos demais recursos, observou-se que os recursos de depuração e ajuda rápida são oferecidos por poucos IDEs, diferenciando-os significativamente dos demais.

Devido às suas características, o IDE Eclipse com o *plugin* EclipseFP foi considerado o mais recomendável para iniciantes. Alguns dos fatores que levaram a essa escolha:

- facilidade de instalação;
- problemas pós instalação estavam documentados e eram de fácil correção;
- interface limpa, intuitiva e amigável;
- multiplataforma abrangendo Windows, Linux e MacOS;
- Depurador disponível;
- Único a apresentar ajuda rápida;
- Auto completar mais poderoso (para funções pré definidas e funções de usuário);
- Checagem de sintaxe executada a cada vez que o arquivo é salvo.

Com essa gama de características, esta ferramenta foi escolhida para o estudo de caso apresentado na próxima seção.

4. Estudo de Caso

O IDE Eclipse, com seu *plugin* para Haskell, foi escolhido para um estudo de caso. O objetivo do estudo foi reunir opiniões e obsevações sobre seu uso em uma disciplina de graduação. Optou-se por uma abordagem qualitativa, alinhada com o caráter exploratório do estudo. A disciplina escolhida foi a de Paradigmas de Programação na Universidade Federal de Santa Maria (UFSM). Trata-se de uma disciplina teórico-prática de 4 horas/semana, oferecida a alunos que já tenham cursado pelo menos dois semestres de disciplinas de programação de computadores. Este tipo de disciplina é comumente oferecido em outras universidades, visando ampliar o conhecimento dos alunos sobre diferentes paradigmas de programação. Na Universidade Federal de Santa Maria, o programa da disciplina aborda quatro diferentes

paradigmas, sendo um deles o paradigma funcional.

Para ensino do paradigma funcional, várias edições da disciplina empregaram a linguagem Haskell, utilizando como software de apoio apenas um interpretador (Hugs ou GHC) e um editor qualquer de programas. Com estas ferramentas, os alunos repetem várias vezes o seguinte ciclo: editar programa, salvá-lo, carregá-lo no interpretador, observar resultado e corrigir o programa. Observou-se que muitos alunos se atrapalham neste ciclo, por exemplo esquecendo de carregar o programa modificado (e assim terminando por executar o programa antigo). Este tipo de problema motivou a introdução do IDE Eclipse em uma edição da disciplina. Nos parágrafos seguintes, apresentam-se maiores detalhes sobre este estudo de caso (4.1) e seus resultados (4.2).

4.1. Descrição do Caso

No caso em questão, a turma era formada por cerca de 30 alunos, que tiveram 15 horas de aulas teóricas/práticas sobre o paradigma funcional e a linguagem Haskell. Em uma aula prática, todos os alunos utilizaram o interpretador GHC e um editor de programas à escolha. Nas demais aulas, totalizando cerca de 6 horas, indicou-se a ferramenta Eclipse como opção aos alunos. Esta ferramenta, com seu *plugin* EclipseFP, tinha sido instalada no laboratório, mas nas primeiras práticas notou-se que alguns computadores não estavam corretamente configurados. Assim, houve alunos interessados no IDE que não puderam utilizá-lo. Os problemas foram corrigidos nas práticas seguintes e, além disso, os alunos podiam usar seus próprios notebooks em aula, de modo que alguns optaram por instalar a ferramenta ao invés de usar os computadores do laboratório.

Durante as práticas, os alunos foram incentivados a emitir opiniões sobre a experiência. Essas opiniões eram verbalmente, no momento em que o professor atendia cada aluno (ou duplas de alunos) durante a prática, e registradas em anotações feitas pelo professor. Além disso, foi colocado um formulário on-line à disposição dos alunos, onde os mesmos podiam registrar suas impressões anonimamente. O formulário continha perguntas objetivas sobre a experiência dos alunos com IDEs em geral e, em particular, com o EclipseFP durante a disciplina. No formulário também havia espaços para os alunos registrarem comentários e sugestões.

4.2. Resultados

Durante esta breve experiência, foi possível observar vários aspectos positivos do uso do IDE. Como o ambiente integra todas as ferramentas necessárias às práticas com Haskell, os alunos pareceram realizar com mais facilidade o ciclo de edição e teste dos programas. De fato, muitos alunos comentaram que preferiram utilizar o IDE ao invés da combinação editor-interpretador, também devido à variedade de recursos disponíveis. Além disso, notou-se que muitos alunos acharam a experiência positiva devido à popularidade do Eclipse, que é uma ferramenta de desenvolvimento amplamente utilizada para linguagens como C, C++ ou Java. Estima-se que esse fato ajude a aproximar a linguagem Haskell das linguagens de programação mais populares, servindo como um fator motivacional para o aprendizado desta nova linguagem. Por fim, alguns alunos registraram que o uso do IDE os ajudou a evoluir na programação em Haskell, o que corrobora a hipótese inicial desta experiência.

Como era esperado, alguns aspectos negativos também foram observados. O aspecto mais marcante foi que alguns alunos tiveram problemas ao tentar instalar a ferramenta em seus notebooks ou ao utilizar um computador do laboratório com a

ferramenta parcialmente instalada (algumas instalações ficaram sem as opções de realce de sintaxe e de ajuda rápida). Este fato acabou desmotivando alguns alunos, que preferiram utilizar apenas as ferramentas básicas (editor-interpretador). Esperava-se que alguns alunos também tivessem dificuldades com a interface do Eclipse que, embora amigável, não é especialmente simplificada para fins educacionais. No entanto, no caso em questão, isso não pareceu trazer dificuldades para os alunos que utilizaram a ferramenta.

5. Considerações Finais

Este trabalho abordou o uso de IDEs como tecnologia de apoio ao ensino da linguagem de programação Haskell. Ao contrário de outros trabalhos centrados em IDEs concebidas para fins educacionais, o foco deste trabalho é a aplicação de IDEs profissionais no processo de ensino-aprendizagem.

Como uma de suas contribuições, este trabalho traz um comparativo de características dos IDEs profissionais disponíveis para Haskell. Este comparativo serviu para eleger-se um IDE que pareceu mais apropriado para uso em sala de aula, mas pode também servir a programadores mais experientes que desejem escolher uma ferramenta rapidamente.

Outra contribuição deste trabalho está no relato de uso do IDE Eclipse, com seu *plugin* para Haskell, em uma disciplina sobre paradigmas de programação. Embora a experiência tenha sido breve, os resultados confirmaram a hipótese de que o IDE é vantajoso no processo de ensino-aprendizagem desta linguagem. Além disso, o uso de um IDE profissional serviu como fator de motivação para muitos alunos.

Como trabalhos futuros, pretende-se ampliar a experiência com o EclipseFP em outras edições da disciplina, evitando que os alunos tenham problemas de instalação e configuração do ambiente. Para isso, pretende-se reproduzir as condições que levaram a erros e, se for necessário, gerar um documento com orientações sobre como resolvê-los. Sabe-se também que a ferramenta EclipseFP está em contínuo desenvolvimento, por isso entende-se que é necessário acompanhar seu fórum de discussões e, quando pertinente, relatar erros que poderão ser corrigidos em outras versões.

Referências Bibliográficas

ALLEN, E.; CARTWRIGHT, R.; STOLER, B. DrJava: a lightweight pedagogic environment for Java. In: **SIGCSE '02: SIGCSE technical symposium on computer science education**, 33, 2002, Cincinnati, USA. Anais. ACM, 2002, p. 137-141.

ASHTON, D.; OLIVE, C.; TRAVERS, J.; WHEST, L. **HEAT – A Beginners' IDE for Haskell**. Technical Report. Canterbury: University of Kent, 2005.

CASPERSEN, M. E. **Educating Novices in The Skills of Programming**. Aarhus, Denmark: University of Aarhus, 2007. 323 p. Tese de Doutorado.

HENRIKSEN, P.; KÖLLING, M. Greenfoot: combining object visualisation with interaction. In: **OOPSLA '04: ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications**, 19, 2004, Vancouver, Canada. Anais. ACM, 2004, pp. 73-82.

HUGUES, J. Why Functional Programming Matters. **Computer Journal**. v.32, n.2, p.

98-207, 1989.

JONES, S. P. Haskell 98 Language and Libraries – The Revised Report. 2002. Disponível em: <http://www.haskell.org/onlinereport/>. Acesso em: Outubro/2011.

KÖLLING, M., QUIG, B., PATTERSON, A.; ROSENBERG, J., The BlueJ system and its pedagogy, **Journal of Computer Science Education**, v.13, n.4, p. 249-268, 2003.

PEARS, A.; SEIDMAN, S.; MALMI, L.; MANNILA, L.; ADAMS, E.; BENNEDSEN, J.; DEVLIN, M.; PATERSON, J. A survey of literature on the teaching of introductory programming. **SIGCSE Bull**. v.39, n.4, p. 204-223, 2007.