Ambiente para Aprendizagem de Programação com apoio Dialogado por Assistentes Inteligentes

Renato de M. Santos[1], Crediné Silva de Menezes [1,2]

[1]Programa de Pós-Graduação em Informática - Universidade Federal do Espírito Santo (UFES) - Caixa Postal l01.9011 - CEP 29.075-910 – Vitória – ES – Brasil

[2]Faculdade de Educação - Universidade Federal do Rio Grande do Sul (UFRGS) – Caixa Postal 90046-900 - Av. Paulo Gama, Farroupilha – Porto Alegre – RS - Brasil

{renatosantos.ti, credine}@gmail.com

Abstract. *This article introduces AmPaRe - a Python programming web environment that uses intelligents agents responsible for supporting students through conversation in their reported difficulties and in the mistakes made during the proposed exercises. We also seek to release the teacher from the basic and recurrent attendances and also gives conditions for the teacher to decide which service demands can be transferred to the environment according to the context of the activity and the predicted difficulties.*

Resumo. *Este artigo apresenta o AmPaRe – um ambiente web para a aprendizagem de programação em linguagem Python que emprega assistentes inteligentes responsáveis por apoiar, via conversação, os alunos iniciantes em programação em suas dificuldades informadas e nos erros cometidos durante realização dos exercícios propostos. Também buscamos liberar o professor dos atendimentos básicos e recorrentes além de dar as condições para que o docente decida quais demandas de atendimento poderão ser transferidas para o ambiente de acordo com contexto da atividade e das dificuldades previstas.*

1. Introdução

O ensino de programação possui uma forte demanda de interação a fim de atender, acompanhar, mediar e avaliar individualmente os alunos e suas atividades (Raabe e Silva, 2005). Adicionalmente, Aureliano, Tedesco e Giraffa (2016) afirmam que é extremamente difícil para um professor atender todas as demandas dos alunos ocorridas durante a realização das atividades e esse problema agrava-se mais quando o aluno se encontra fora dos limites de sala de aula.

No contexto da educação, uma interface com língua natural motiva a participação dos aprendizes e desempenha um papel importante no aprendizado (Santos, De Menezes e Cury, 2018). No entanto, um *feedback* individual pode consumir muito tempo do professor, com o risco de que outros estudantes possam não se beneficiar atendimento do docente no devido tempo.

As perguntas dos alunos oportunizam momentos de reflexão nos quais eles se sentem mentalmente envolvidos e desacomodados (De Camargo et al., 2011), neste instante inicia-se um processo de inquietação, desencadeando a tomada de consciência

DOI: 10.5753/cbie.sbie.2019.1511

sobre os seus conhecimentos anteriores, de modo a fazer comparações e relações, no sentido de fomentar a organização da pergunta (De Camargo et al., 2011) e, por conseguinte, esse conjunto de ações já se configura como um momento de aprendizagem, pois ao apresentar a sua pergunta, o sujeito já está reestruturando o seu pensamento.

No âmbito das tecnologias educacionais é percebida uma variedade de pesquisas e ferramentas que visam propiciar o diálogo com aluno ao longo de seus estudos. Os CAI (*Computer Aided Instruction*) e os STIs (*Sistemas Tutores Inteligentes*) frequentemente buscam implementar o diálogo com o aluno e vem progressivamente alçando sucesso (Gerdes et al., 2017).

O objetivo desse artigo é evidenciar como o uso do suporte computacional, concebido para monitorar, analisar e a atender as dúvidas dos alunos, pode minimizar o esforço e o tempo dedicado pelo professor nestas atividades ao tempo que pode propiciar ao aluno um atendimento no imediato momento de suas dificuldades.

Na fase 1 desse trabalho foi realizado o registro de ocorrências e das dificuldades de aluno e professores durante um semestre acadêmico e na fase 2 o ambiente proposto foi colocado em uso durante um curso de introdução à linguagem de programação.

2. Dificuldades no processo de ensino-aprendizagem de programação

Embora a importância do ensino de programação, as dificuldades dos estudantes aprenderem esta disciplina é notória (Gomes et al., 2015) e apresenta desafios (Júnior; Fechine; Costa, 2009). Para Raabe e Silva (2005) as dificuldades de aprendizagem de programação: os problemas de natureza didática, os problemas de natureza cognitiva e os problemas de natureza afetiva.

Já a pesquisa recente de Moreira et al. (2018) aponta que as dificuldades ligadas à programação identificadas há décadas ainda persistem e continuam preocupantes. Suas conclusões indicam que a maior dificuldade desses alunos está no desenvolvimento da lógica de programação (42.72%), no entendimento da sintaxe (34.54%), na falta de tempo na dedicação na disciplina (26.36%) e na dificuldade de interpretação dos problemas propostos (15.45%).

2.1 Dificuldades dos alunos

O trabalho realizado durante a primeira fase do estudo de caso permitiu observar e registrar as seguintes ocorrências de dificuldades dos alunos:

- Os alunos não recebiam a ajuda adequada em tempo hábil, por conseguinte eles desistiam da atividade ou buscavam ajuda direta entre os colegas que então os ajudavam sem o devido cuidado pedagógico, fornecendo-lhes respostas diretas para solução do problema, incluindo a cópia código-fonte;
- Os erros informados na tentativa de execução do programa, além de estarem numa língua estrangeira, eram formulados numa linguagem forma e técnica, logo não contribuíam efetivamente para o aprendizado do aluno;
- O aluno não assimilava ou refletia sobre a causa do programa não funcionar, mesmo quando obtendo alguma mensagem indicando o erro;

- As dúvidas sobre o enunciado ou sobre como iniciar a resolução das atividades eram constantes;
- As informações obtidas nas apostilas e livros não eram adequados por não oferecem um conteúdo contextualizado com a dificuldade e por dificultar uma pesquisa rápida.

2.1 Dificuldades dos professores

Durante estudo de caso foram identificadas as seguintes dificuldades na atuação do professor:

- Dificuldade em atender a tempo o aluno, no momento de sua dúvida;
- Dificuldade de identificar a tempo os conceitos malformados que evidentemente atrapalhavam progresso do aluno;
- Dificuldade em fornecer uma resposta de erro menos informativa e mais pedagógica quando o aluno desenvolvia código-fonte com problemas;
- Dificuldade de responder ao aluno quando este realizava a atividade em casa ou fora do momento da aula e apresentava dificuldades nestes momentos;
- Dificuldade em monitorar e corrigir a ajuda equivocada fornecida por outros alunos, que embora bem-intencionados, pulavam etapas importantes ao fornecer uma resposta muito direta ou a completa solução do problema.

3. Trabalhos Correlatos

O levantamento dos trabalhos correlatos consistiu de uma pesquisa bibliográfica nas publicações realizadas no período de 2008 a 2018, assim foram consideradas as bases brasileiras WCBIE (Anais dos Workshops do Congresso Brasileiro de Informática na Educação), RBIE (Revista Brasileira de Informação na Educação), SBIE (Simpósio Brasileiro de Informação na Educação), WIE (Workshop de Informática na Escola) e também as publicações internacionais disponíveis no repositório IEEE. Assim, foram selecionados 45 artigos para uma análise mais detalhada do quais 7 artigos foram considerados diretamente correlatos e, portanto, serão apresentados.

A fim de possibilitar a análise, foi desenvolvido o Quadro 1 que reúne as principais características identificadas em contraste com a proposta apresentada.

Quadro 1 - Comparação entre trabalhos correlatos.

	Action Remember	Analogus	Ask-Elle	CodeMage	LabPy	Tree Walkers	Proposta
Técnica para reconhecimento da dificuldade	RBC	RBC AIML	AST	Análise de Erros	N-gramas	AST	Sim
Extensível	Não	Não	Não	Não	Não	Sim	Sim
Possui editor de código integrado	Não	Sim	Sim	Sim	Sim	Sim	Sim
Ambiente online	Sim	Sim	Sim	Sim	Sim	Não	Sim

Flexibiliza configurar ajuda por exercícios	Não	Não	Sim	Não	Não	Sim	Sim
Contexto utilizado no apoio	Formulários	Atividade	Código-fonte	Erros gerados	Código-fonte; atividade	Código-fonte	Sim
Aprende com os de atendimentos anteriores	Sim	Não	Não	Não	Sim	Não	Sim
Responde as dúvidas registradas	Sim	Sim	Não	Não	Sim	Não	Sim
Explica as causas de erros	Não	Não	Sim	Sim	Sim	Sim	Sim
Guia à resolução a cada passo	Não	Não	Sim	Não	Não	Não	Não
Aluno interage com o professor ou colegas	Sim	Não	Não	Não	Sim	Não	Sim
Exibe avaliação da atividade	Não	Não	Não	Não	Sim	Não	Sim
Trata afetividade	Não	Sim	Não	Não	Sim	Não	Não
O professor recebe e atua dificuldades não resolvidas pelo ambiente	Sim	Não	Não	Não	Sim	Não	Sim
Possui avaliação automática da solução	Não	Não	Não	Não	Sim	Não	Não
Possui testes automatizados	Não	Não	Não	Sim	Não	Sim	Sim

Os *feedbacks* apresentados nos trabalhos correlatos possuíam um contexto pré-formatado, normalmente eram utilizados o código-fonte ou os erros ocorridos, sendo ignorado, portanto, a conversação ou a enunciação do aluno ao expressar a sua dúvida. O histórico de atendimento do professor não foi explorado no atendimento de casos recorrentes, exceto no *Action Recommender*, entretanto, neste ambiente o código do aluno e os erros ocorridos foram ignorados ao se analisar a dificuldade do aprendiz.

Nossa proposta se diferencia por prevê a extensibilidade do ambiente ao permitir que agentes inteligentes sejam acoplados ao ambiente, estendendo a área de suporte prevista inicialmente. Outro diferencial refere-se à capacidade do ambiente aprender com o suporte oferecido pelo professor, podendo ao longo do tempo possuir os mesmos estilos de mediação do docente à medida que evolui a sua capacidade de atendimento.

4. Metodologia de desenvolvimento

Primeiramente, com bases em experiências anteriores de regência de sala de aula, foi proposto uma primeira versão do ambiente. No semestre de 2018/2 o ambiente proposto foi disponibilizado para uso a partir do que suas funcionalidades foram refinadas na medida em que se identificava as necessidades de ajustes. Neste período as necessidades do professor e de alunos foram observadas e registradas para evolução da proposta do ambiente. No segundo momento da pesquisa durante o semestre 2019/1 uma prova de conceito foi criada e então foi submetida para uso durante um curso de reforço para alunos iniciantes programação de um curso técnico em informática. Durante o período do curso foi realizada a coleta de dados a partir dos registros internos do sistema, da avaliação do aluno para cada atendimento recebido e através de questionários

qualitativos realizados semanalmente. Os dados foram agrupados e analisados a fim de apontar os resultados obtidos.

5. Ambiente proposto

O ambiente proposto foi denominado de *AmPaRe* (***Am***biente de ***P***rogramação com Suporte aos ***A***tendimentos de Demandas ***Re***correntes) e visa oferecer ao professor os recursos para que o docente delegue uma parte de seu atendimento a quem se denominou aqui de assistentes conversacionais (ACs) e assistentes avaliadores de códigos (ACOs), ao tempo que propicia a condições para um atendimento imediato ao aluno no momento de suas dificuldades.

Os assistentes são constituídos a partir do emprego de agentes inteligentes (Russell e Norvig, 2013) criados para lidar com o conhecimento prévio do professor e com as demandas de suporte decorridas durante a realização das atividades propostas. Neste contexto, se buscará manter sempre preservado a liberdade para que o professor decida quais atendimentos serão delegados aos assistentes, enquanto que é salvaguardado a autonomia para que os assistentes interajam e atuem segundo os seus mecanismos internos de inferência ao lidar com o ambiente.

Conforme ilustrado na Figura 1, no **Passo 1** o professor cria uma instância a partir dos assistentes disponíveis no ambiente e os especializa para o contexto de uma atividade.

Cada assistente possui em seus mecanismos internos as condições para operar sobre o um ambiente, todavia, o professor poderá especializá-lo com base em sua experiência e forma própria de mediação. No **passo 2** os alunos acessam as atividades disponíveis para solucioná-las.

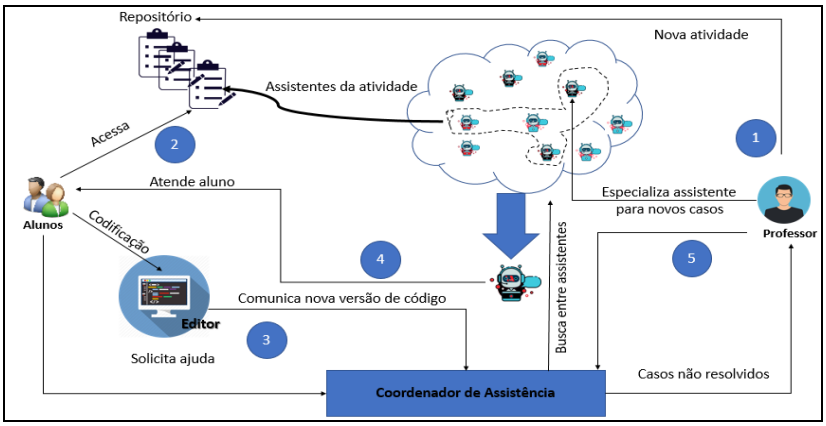

Figura 1 - Visão geral do ambiente.

Cada atividade poderá ser configurada com um conjunto de assistentes, cada qual especializado num tipo de dificuldade prevista; no **passo 3** o aluno codifica a solução do problema no editor disponível no ambiente. Ao se deparar com uma dificuldade o aluno poderá manisfestá-la através do ambiente de conversação.

O código-fonte do aluno também é enviado a cada nova versão desenvolvida; e no **passo 4** um assistente é definido dentre os vários disponíveis para o atendimento. Esse assistente será responsável por desenvolver uma conversação com o aluno no

contexto da dificuldade identificada; No **passo 5,** caso a dúvida do aluno persista ou não seja encontrado um assistente adequado para o atendimento, o professor receberá a dificuldade informada, dessa forma o docente poderá realizar um atendimento direto ao aluno e opcionalmente poderá especilizar ou criar uma nova instância de assistente para o atendimento de novos casos semelhantes.

Os assistentes conversacionais (**Erro! Fonte de referência não encontrada.**) possui comportamento previsto para atendimento as dificuldades informada via texto podendo ser especializados dialogarem sobre conceitos abordados nas aulas teóricas, sobre dificuldade de interpretação do problema apresentado ou dúvidas relacionada aos recursos da linguagem de programação, nisto incluem o fornecimento de exemplos de sintaxe e semântica da liguagem.

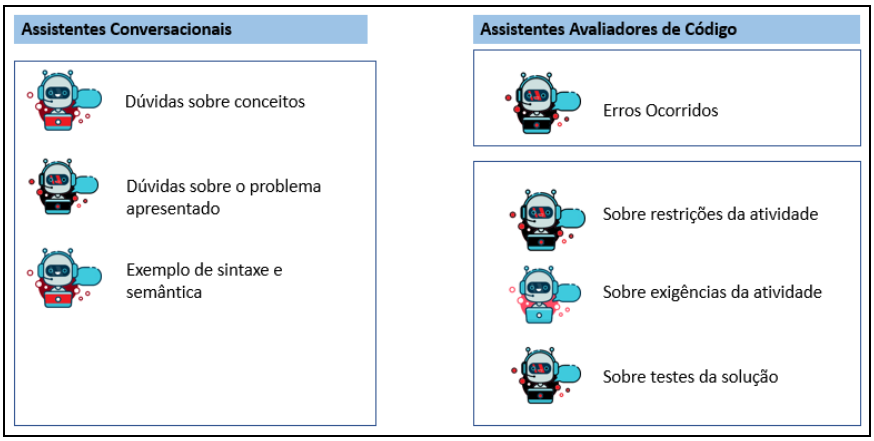

Figura 2 - Proposta de família de assistentes.

Já os assistentes avaliadores de código (Figura 2) são dedicados a análise do código-fonte do aluno e poderão ser especializados para o tratamento de erros identificados programa desenvolvido, para a conversação sobre as restrições violadas no código-fonte, por exemplo, o impedimento de uso de dada função ou de algum recurso externo não liberado pelo professor para tal atividade. Esses assistentes também poderão verificar se aluno cumpriu certas exigências no código-fonte ou mesmo testar certas condições específicas do código a fim de validá-lo ou de comunicar possíveis ajustes.

5.1 Implementação

Buscou-se implementação de um ambiente inteiramente funcional via Web a fim de permitir que tanto professor quanto alunos acessem o ambiente *AmPaRe* de qualquer lugar e a qualquer tempo. Tal iniciativa propicia uma extensão do laboratório, além do mais viabiliza a economia de tempo gasto com a preparação do ambiente de codificação necessário à resolução das atividades

Os assistentes conversacionais processam o texto informado pelo aluno ao expressar sua dificuldade e então cada assistente disponível na atividade avalia sua a condição para atendimento. Se a demanda for condizente com a sua especialidade o assistente busca em sua base de conhecimento uma resposta adequada ao contexto.

O mecanismo de inferência dos assistentes conversacionais foi desenvolvido com base no algoritmo *Naive Bayes* (Russell e Norvig, 2013). Sua implementação visou

apoio na identificação das intenções relacionadas a um texto e tomada de decisão do assistente conforme a intenção relacionada.

Conforme ilustra a Figura 3a, no processamento do texto é realizada a separação dos termos contidos na frase, a remoção de *Stop Words* e por último o processo de *Stemmer* do texto. Para a realização do *Stop Words* e o do processo de *Stemmer* sobre o texto foram utilizadas as classes contidas no pacotes *nltk.stem.RSLPStemmer* e *nltk.corpus.stopwords*, respectivamente. Ambas oriundas da biblioteca NLTK (*Natural Language Toolkit*) disponível para *Python*.

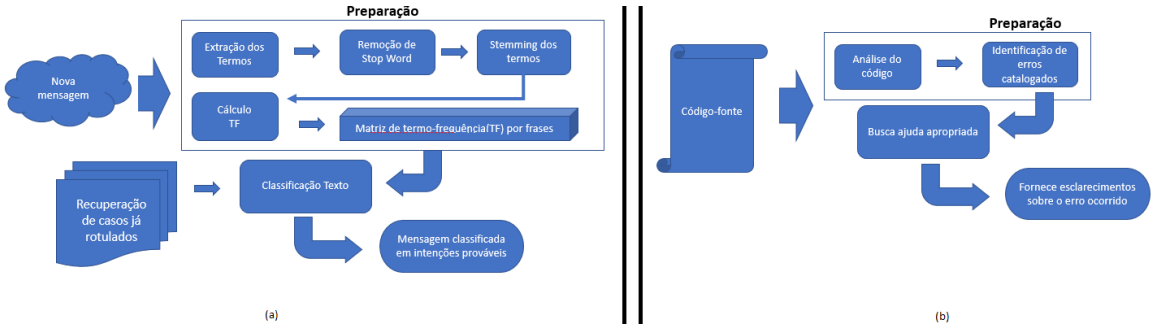

Figura 3 - Processo de identificação de dificuldades durante a conversação ou no código-fonte.

Com relação aos assistentes avaliadores de código, foi implementado um assistente responsável pela identificação de erros através da análise de código-fonte. Conforme ilustra a Figura 3b, o mecanismo de inferência dos assistentes de erro é realizado através da análise do código-fonte, então é checado um catálogo de erros previstos, uma vez identificado o erro o assistente recorre às mediações previamente definidas pelo professor para aquele caso em específico. A ferramenta *Pylint* foi utilizada na construção do mecanismo interno do assistente de erros enquanto que o professor é o responsável viabilizar as ações de suporte para cada contexto de atividade.

5.2 Ambiente de autoria e de conversação sobre as dificuldades ocorridas

Quando um aluno abre uma atividade para resolvê-la é disponibilizado um ambiente de codificação, o ambiente de conversação e a área de execução do código desenvolvido.

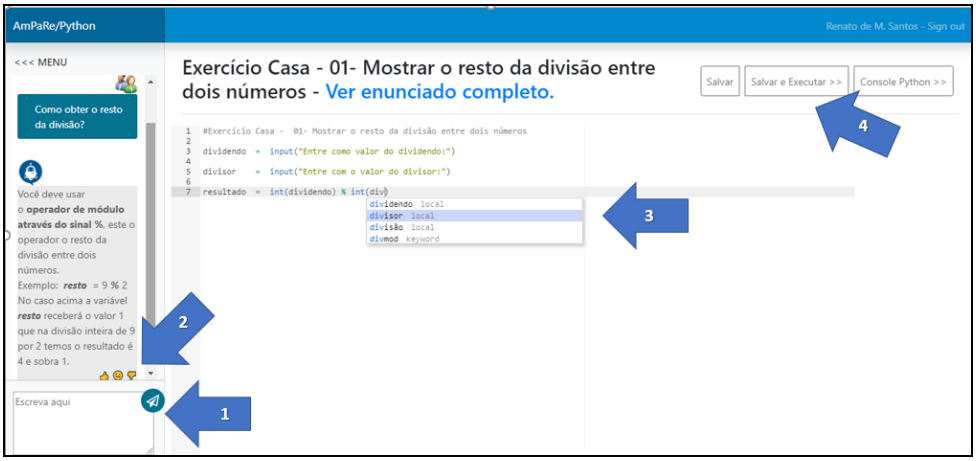

Figura 4 - Ambiente de resolução de atividades.

Na Figura 4 são apresentados, em 4 passos, as principais funcionalidades dispostas no ambiente de resolução da atividade proposta.

No **passo 1** o aluno solicita uma ajuda descrevendo a sua dificuldade na área de conversação. Essas dúvidas podem ser diversas, por exemplo dúvida sobre o enunciado, sobre o uso de um dado comando e etc. A resposta adequada dependerá da existência de um assistente condizente com a necessidade informada. Já no **passo 2** um assistente retorna à solicitação de ajuda e o aluno avalia o *feedback* recebido. No **passo 3** é apresentada a área de codificação com recurso de auto endentação, destaque na sintaxe, comandos e funções nativas da linguagem e o auto complemento de código.

No **passo 4** o aluno poderá visualizar o enunciado completo da atividade sem sair da página, salvar a versão atual de seu código ou executá-lo. Também nesta área é disponibilizado o *Console Python* online, onde o aluno poderá experimentar comandos e ideias sem comprometer seu código principal. Ainda no **passo 4**, caso seja identificado algum erro ao salvar ou executar o programa do aluno, um *Assistente Conversacional* é acionado na área de conversação.

6. O experimento

O experimento foi realizado a partir da promoção de um curso de introdução a programação de computadores através da linguagem de programação *Python*. Foram inscritos 41 alunos dos quais 17 foram selecionados. Os critérios para seleção foram: alunos que se auto avaliaram com muita dificuldade na disciplina e aqueles indicados por seus professores como alunos com baixíssimos rendimentos.

O curso proposto possuiu carga horária de 40 horas com duração de 4 semanas, sendo 20 horas distribuídas em aulas presenciais compostas de teoria e práticas em laboratório e 20 horas em atividades extraclasse. Os assuntos foram abordados no curso foi uma adaptação de (Berssanette e Francisco, 2018) para a linguagem *Python*.

6.1 Avaliação do ambiente

Semanalmente foram aplicados questionários a fim de avaliar qualitativamente a satisfação dos alunos quanto ao suporte oferecido pelo ambiente e do desenvolvimento do aluno durante as atividades. Paralelamente o sistema registrava em log as suas decisões sobre as demandas de suporte ocorridas e o seu comportamento ao apoiá-las.

Ao avaliar a aceitação do aluno quanto a mudança da metodologia de ensino através do uso do *AmPaRe* foi possível verificar uma gradativa aceitação do aluno. Na primeira semana 41.2% alunos concordavam plenamente com o uso da metodologia apoiada pelo ambiente, sendo que 47.7% concordavam parcialmente. Ao término da 4º semana, houve uma convergência para concordância plena (50%) ou parcial (50%).

Aos alunos se auto avaliarem sobre o seu desenvolvimento na realização das atividades, foi observado que na primeira semana 58.8% deles julgavam estar plenamente confiantes e motivados, subindo 66,7% a partir da 3º semana.

Ao avaliar a opinião dos alunos quanto aos atendimentos fornecidos pelo ambiente foi identificado um gradativo crescimento na satisfação dos alunos. Ao término do experimento 50% dos alunos julgavam que suas necessidades estavam

plenamente atendidas pelos assistentes no ambiente, enquanto que os demais 50% dos alunos informaram que precisaram da complementação do suporte pelo docente.

6.2. Avaliação quantitativa

Durante o período do curso foram disponibilizados 28 exercícios dos quais 19 foram desenvolvidos em laboratório e 9 foram extraclasse durante a semana. No período foram realizadas 106 solicitações de ajuda e 627 ocorrências de erros. Para todos os casos demandados o sistema atuou fornecendo o suporte para os erros ocorridos.

Ao avaliar o atendimento na perspectiva do *feedback* fornecido pelos alunos sobre o suporte recebido identifica-se que o sistema atendeu adequadamente em 60% dos casos. Quando essa análise foi realizada sob perspectiva do professor quanto ao comportamento esperado nos assistentes foi identificado que o sistema atendeu corretamente em 84% dos casos de pedido de suporte.

6.3. Considerações finais sobre o experimento

Foi observado um crescimento gradual na satisfação dos alunos com as respostas recebidas no decorrer das semanas do experimento. Enquanto na primeira semana apenas 35% dos alunos informavam estar satisfeitos com os feedbacks recebidos, ao término do experimento foi verificado que 100% dos alunos ou estavam plenamente satisfeitos (50%) ou eram parcialmente satisfeitos (50%), sendo que os casos onde o sistema não atendia nenhuma das necessidades dos alunos somente foi significativo até a segunda semana, onde esse índice foi de 17%.

Já a análise quantitativa do comportamento geral dos assistentes demonstrou que eles agiram corretamente em 60% dos casos e 84% se considerado apenas o comportamento previsto pelo professor. Enquanto que a avaliação qualitativa apontou uma boa aceitação por parte dos alunos e promoveu o seu desenvolvimento.

7. Considerações finais

Durante o período do curso foram disponibilizados 28 exercícios dos quais 19 foram desenvolvidos em laboratório e 9 para resolução em casa e durante a semana. No período foram realizadas 106 solicitações de ajuda e 627 ocorrências de erros. Para todos os casos o sistema atuou fornecendo o suporte para dificuldades informada e as explicações das causas de ocorrências dos erros no código-fonte.

Nosso objetivo foi alçado ao demonstrar a aplicação e a utilidade do ambiente proposto no atendimento às dificuldades ocorridas durante as atividades de programação e permitiu liberar o professor dos atendimentos que o docente julgou triviais e repetitivos.

Os trabalhos futuros poderão investigar a criação de assistentes dedicados ao acompanhamento da depuração do código-fonte. Os assistentes poderão possuir a habilidade de interpretar e tratar emoções ou empregar técnicas para incorporação de diálogos longos, onde eles poderiam realizar questionamentos ao aluno ou propor o aprofundamento da conversa num dado assunto. Os assistentes poderiam empregar técnicas para incorporação de diálogos longos, onde de forma mais proativa, poderiam realizar questionamentos ao aluno ou propor o aprofundamento da conversa num dado

assunto; Redes Bayesianas (RB) poderão ser integradas na construção de hipóteses sobre as dificuldades dos alunos num certo contexto.

8. Referências

Alves, Fábio P.; Jaques, Patrícia. Um ambiente virtual com feedback personalizado para apoio a disciplinas de programação. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2014.

Aureliano, Viviane Cristina O.; tedesco, PC de AR; Giraffa, Lúcia Maria M. Desafios e oportunidades aos processos de ensino e de aprendizagem Congresso da Sociedade de programação para iniciantes. In: Brasileira de Computação. 2016.

Berssanette, João Henrique; Francisco, Antônio Carlos. Proposta de Abordagem Prática para o Ensino de Programação Baseada em Ausubel. In: Simpósio Brasileiro de Informática na Educação-SBIE). 2018.

De Camargo, Andrea Norema Bianchi et al. A pergunta na sala de aula: concepções e ações de professores de Ciências e Matemática. 2011.

Gerdes, Alex et al. Ask-Elle: An adaptable programming tutor for Haskell giving automated feedback. International Journal of Artificial Intelligence in Education, 2017.

Gomes, Marina et al. Um estudo sobre erros em programação-Reconhecendo as dificuldades de programadores iniciantes. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2015. p. 1398.

Júnior, Gps Santos; Fechine, Joseana Macêdo; Costa, E. d B. Analogus: Um Ambiente para Auxílio ao Ensino de Programação Orientado pelo Raciocínio por Analogia. XVII WEI, v. 28, 2009.

Moreira, Gabriel Luídy et al. Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. Anais do ECOP/UFERSA, v. 2, n. 1, 2018.

Raabe, André Luís Alice; Silva, JMC da. Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos. In: XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil. 2005.

Raabe, André et al. Avaliação do feedback gerado por um corretor automático de algoritmos. In: Brasileiro de Informática na Educação-SBIE). 2015. p. 358.

Russel, Stuart; Norvig, Peter. Inteligência Artificial. 3a Ed. Campus, 2013.

Santos, Renato; De Menezes, Crediné; Cury, Davidson. Uma Arquitetura de Tutor Inteligente que Provê Suporte ao Diálogo com o Aluno Iniciante em Linguagem de Programação. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2018. p. 768.

Sirotheau, Silverio et al. LabPy: Laboratório virtual de ensino em Python. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2018.

Whittall, S. J. et al. CodeMage: educational programming environment for beginners. In: 9th International Conference on Knowledge and Smart Technology (KST). 2017.