Towards a Conversational Agent to Support the Software Testing Education

Leo Natan Paschoal
University of São Paulo
São Carlos - SP – Brazil
paschoalln@usp.br

Lucas Fernandes Turci
University of São Paulo
São Carlos - SP – Brazil
lucas.turci@usp.br

Tayana Uchôa Conte
Federal University of Amazonas
Manaus – AM – Brazil
tayana@icomp.ufam.edu.br

Simone R. S. Souza
University of São Paulo
São Carlos – SP – Brazil
srocio@icmc.usp.br

ABSTRACT

The training of professionals in the field of software testing is increasing its relevance in the past few years and, therefore, efforts in appropriate methodologies for the learning-teaching process in this context have been proposed and appreciated. The emergence of pedagogical models, such as flipped classroom and team-based learning, which demand from the students a previous study of the theory before the lecture, creates a concern: how to support the before class learning? Because of the hybrid nature of these pedagogical models, which means they mix elements from traditional and distance education, it is possible that the support mechanisms used in distance learning platforms, such as conversational agents, can be applied for this matter. At the same time in which the academic work tries carefully to provide a proper software testing formation, there are also many contributions being established regarding the training and non-formal learning. Improvement and personal training courses about criteria, tools, and software testing good practices are being created by teaching institutes and offered in Massive Open Online Courses platforms (MOOCs). However, in this type of course, in the absence of a teacher, the student might be in a situation where there is nobody available to answer their questions about the topic. In this paper, we propose the use of conversational agents in solving the problems and challenges which encompass the learning through MOOCs and hybrid models. A conversational agent, called TOB-SST is proposed to support software testing education. A viability study was conducted to understand the quality of the given answers by TOB-SST and the possibility of it serving as a learning support tool. The results indicate that it is promising to employ a conversational agent to guide student study.

CCS CONCEPTS

• **Software and its engineering** → *Software verification and validation*; • **Social and professional topics** → Software engineering education;

SBES 2019, September 23–27, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8/19/09...$15.00.
https://doi.org/10.1145/3350768.3352456

KEYWORDS

Chatbot, Computer Science Education, Software Testing.

ACM Reference Format:
Leo Natan Paschoal, Lucas Fernandes Turci, Tayana Uchôa Conte, and Simone R. S. Souza. 2019. Towards a Conversational Agent to Support the Software Testing Education. In *Proceedings of XXXIII Brazilian Symposium on Software Engineering (SBES 2019), September 23–27, 2019, Salvador, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3350768.3352456

1 INTRODUÇÃO

Teste de software é uma atividade importante para o desenvolvimento de software, uma vez que é planejada e conduzida com o intuito de revelar os defeitos que perfazem o sistema sob teste [20]. Essa atividade, quando bem conduzida, contribui com a melhoria da qualidade do software.

Apesar de existirem indícios que a indústria vem adotando a atividade como uma prática durante o desenvolvimento de soluções de software [27], estudos anteriores revelam que uma das principais limitações da prática de teste na indústria está relacionada com a falta de avaliação da qualidade do teste conduzido e da cobertura do teste [8]. É emergente a necessidade de que a indústria aplique estratégias sistemáticas para a condução da atividade de teste[7, 27]. Para que isso ocorra, é fundamental que profissionais qualificados atuem na área, pois a literatura identifica uma carência de profissionais com o conhecimento adequado em teste de software[15].

A problemática associada ao ensino de teste de software vem produzindo desdobramentos e já foi reconhecida como um dos principais desafios da área [7]. Nesse sentido, pesquisadores e professores vêm investindo esforços em estabelecer contribuições ao ensino de teste de software. Dentre as contribuições, ao longo da última década foram surgindo jogos educacionais, módulos educacionais, ambientes virtuais colaborativos para treinamento [33] e, mais recentemente, os ambientes gamificados [10]. Outra frente de trabalhos está preocupada com o ensino de teste de software na educação não formal. Nesse contexto, há um forte investimento no desenvolvimento de *Massive Open Online Courses* (MOOCs) [9].

Além do estabelecimento de mecanismos de apoio, atualmente, existem algumas discussões sobre o uso de modelos pedagógicos alternativos aos métodos tradicionais de ensino, como *flipped classroom* [22, 25], revisão por pares e aprendizagem baseada em problemas [33]. O uso de modelos pedagógicos alternativos ao modelo tradicional de ensino estão alinhados a argumentação de Soska *et*

al. [31], que descrevem que o meio acadêmico precisa ajustar os métodos de ensino para atender com maior satisfação à demanda da indústria de software.

Apesar de existir uma conscientização da comunidade de computação pelo estabelecimento de mecanismos que buscam apoiar o ensino e o treinamento de teste de software, bem como pelo uso de modelos pedagógicos alternativos ao tradicional, as características desses recursos e modelos podem provocar algumas limitações ao aluno de teste de software durante o processo de aprendizagem.

Em MOOCs, por exemplo, há uma discussão na literatura sobre as baixas taxas de conclusão estarem relacionadas com a insatisfação dos estudantes com a falta de interação com o instrutor. Conforme Mittal *et al.* [18], os tutores não conseguem atender a demanda dos alunos, porque precisam monitorar todas as postagem e responder os alunos de maneira autônoma. De fato, a falta de suporte ao aluno é um problema dos MOOCs que vem sendo abordado em diferentes estudos da área [3, 17]. MOOCs sobre teste de software não estão livres de passar por situações similares.

Em modelos como *flipped classroom* e *team-based learning*, que demandam dos alunos um estudo do conteúdo preliminar a aula, há estudos que descrevem que os alunos relatam que tem dificuldade ao estudar *a priori* a aula. Em uma experiência anterior de aplicação de *flipped classroom* no ensino de teste de software, Paschoal e Souza [25] constataram que os participantes reclamaram que tiveram dificuldades ao se prepararem para a aula porque ficaram com dúvidas e não tinham como solucioná-las. Essa limitação do modelo pode influenciar no aprendizado, uma vez que podem existir cenários de uso do *flipped classroom* em que os alunos desistem de estudar o conteúdo por falta de apoio.

Apesar das limitações levantadas, pesquisas estão sendo conduzidas em busca de soluções para minimizá-las. No contexto de MOOCs, Aguirre *et al.* [3] descrevem que os agentes conversacionais podem oferecer apoio à tutoria dos alunos. Nesse seguimento, Mikic-Fonte *et al.* [17] demonstram alguns esforços para estabelecer um agente conversacional para apoiar alunos de MOOCs sobre arquitetura de computadores.

Em uma das aplicações de *flipped classroom* no ensino de teste de software, Paschoal e Souza [24] descrevem que um agente conversacional com conhecimento sobre critérios e técnicas de teste poderia ser investigado e utilizado.

No estudo de Valle *et al.* [33], ao mapearem os recursos educacionais e abordagens utilizadas no ensino de teste de software, os autores não identificaram agentes conversacionais. Recentemente, Lauvås e Arcuri [14] conduziram um estudo secundário, complementar ao estudo de Valle *et al.* [33], e também não identificaram estudos primário sobre agentes conversacionais para essa temática.

Este artigo contribui neste cenário, investigando a proposição de um agente conversacional (chamado TOB-STT) para apoiar o ensino de teste de software. O agente tem o intuito de interagir com os estudantes e solucionar suas dúvidas. Para o desenvolvimento do TOB-STT foram estudadas abordagens e *frameworks* que apoiam o desenvolvimento desses sistemas. Um estudo sobre a viabilidade do TOB-STT foi realizado. Os resultados indicam a capacidade do agente em solucionar as dúvidas dos alunos, dando atenção ao *feedback* emitido por alunos que interagiram com ele e a qualidade das respostas geradas pelo agente às perguntas feitas.

Este artigo está estruturado da seguinte forma. A Seção 2 apresenta uma visão geral sobre agentes conversacionais. A Seção 3 apresenta detalhes técnicos associados ao desenvolvimento do TOB-STT. Um estudo sobre a viabilidade do TOB-STT como mecanismo de apoio ao ensino é apresentado na Seção 4. Por fim, na Seção 5 são apresentadas as considerações, limitações deste estudo e perspectivas de trabalhos futuros.

2 AGENTES CONVERSACIONAIS

Agentes conversacionais são sistemas de software que conseguem simular uma interação em alguma linguagem natural com um parceiro humano ou com um parceiro de software [6]. Para tanto, fazem uso de algoritmos que são capazes de processar e reconhecer a linguagem utilizada por seres humanos, transmitida por meio de voz ou texto, e gerar um retorno em língua natural para o que foi transmitido.

Aplicações de agentes conversacionais foram estabelecidas para diferentes contextos, desempenhando uma ampla gama de funções. Existem agentes conversacionais sendo estabelecidos para o comércio, para o entretenimento, para o lazer e há uma série de aplicações de agentes conversacionais para apoiar os processos de ensino e aprendizagem [13].

Em particular, no âmbito educacional os agentes conversacionais podem ser utilizados como dispositivos autônomos ou como componentes integráveis para enriquecer e complementar recursos educacionais [26]. Nessa perspectiva, podem ser incorporados em aplicações que incluem baixo suporte ao aluno [6], como em mundos virtuais – atuando como orientador de navegação [12], e em MOOCs – comportando-se como tutor virtual [3].

Além disso, conforme Kerry *et al.* [13], os agentes conversacionais podem desempenhar um papel valioso, uma vez que podem ser estabelecidos com uma variedade de propósitos, incluindo a tutoria, prática de linguagem (para cursos de ensino de idiomas), companhia durante o aprendizado, até mesmo como mecanismo para encorajar o aluno a refletir sobre o conteúdo.

Estudos anteriores relatam que os agentes conversacionais podem contribuir de muitas formas no contexto educacional, apoiando diferentes populações (*e.g.*, estudantes de idiomas [11], universitários [17]), modalidades de ensino e até mesmo nos modelos de ensino híbrido [13], os quais mesclam elementos da modalidade de ensino presencial e a distância, como *flipped classroom* [24].

Em razão da característica associada à comunicação em linguagem natural, esses sistemas de software são estabelecidos para apoiar o ensino em diferentes áreas, como arquitetura de computadores [17], redes de computadores [16] e engenharia de requisitos [21]. Nessas temáticas, os agentes conversacionais são recomendados pelos professores para apoiar estudantes na resolução de dúvidas sobre o conteúdo, realização de atividades práticas e preparo para avaliações.

O estabelecimento de agentes conversacionais é direcionado para temáticas específicas. Uma explicação para isso está relacionada com o fato de que o desenvolvimento dos mesmos exige esforços significativos de engenharia [28]. Esses esforços estão concentrados especialmente na definição das bases de conhecimento do agente [28]. Em virtude disso, as soluções existentes ainda não abrangeram algumas áreas, como o ensino de teste de software [24].

3 TOB-STT

Esta seção tem a intenção de apresentar os esforços que foram realizados durante o estabelecimento do agente conversacional TOB-STT (*a chatBOT to support Software Testing Teaching*). Para tanto, um processo para apoiar o gerenciamento do projeto (Figura 1) será utilizado.

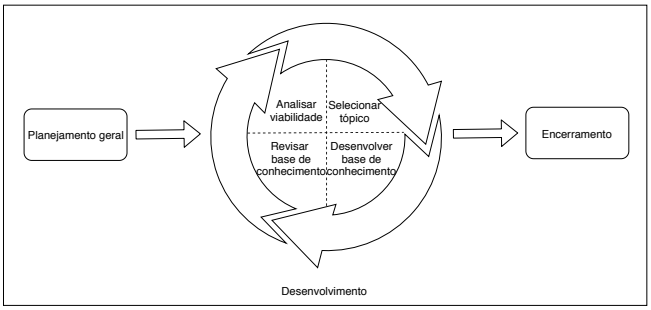

Figura 1: Processo adotado no desenvolvimento do TOB-STT

O processo para o gerenciamento do agente conversacional é inspirado no processo incremental e serve como uma abstração para nortear os objetivos e as atividades que foram desenvolvidas. Desse modo, a primeira fase consiste em planejar o desenvolvimento do agente. A segunda fase envolve um conjunto de atividade: (i) um tópico associado a temática deve ser selecionado; (ii) bases de conhecimento são definidas para o tópico; (iii) um especialista da área inspeciona e revisa as bases de conhecimento; (iv) o agente conversacional é avaliado considerando as bases de conhecimento. Finalmente, após o desenvolvimento de todas as bases de conhecimento, o projeto é finalizado (terceira fase).

O processo de gerenciamento permite descrever e organizar o desenvolvimento do agente. Além disso, a utilização do processo é inspirado em estudos anteriores que foram influenciados por processos para gerenciar o desenvolvimento de agentes conversacionais, tais como desenvolvimento evolucional e desenvolvimento iterativo e incremental. As próximas subseções apresentam as decisões tomadas e ações realizadas em cada etapa do processo.

3.1 Planejamento geral

O estabelecimento do agente conversacional envolveu, inicialmente, o estudo sobre abordagens e *frameworks* que apoiam o desenvolvimento desses sistemas. Nesse sentido, foram localizados estudos secundários na tentativa de identificar uma solução mais adequada para o projeto.

No *survey* de Abdul-Kader e Woods [1], os autores descrevem as principais estratégias de desenvolvimento de agentes conversacionais e ilustram os principais componentes de agentes que são baseados nessas estratégias. A partir do *survey* os autores constataram que existe uma tendência em utilizar uma técnica conhecida como Casamento de Padrões (*Pattern Matching*) e uma linguagem para implementar as bases de conhecimento denominada AIML (*Artificial Intelligence Modelling Language*).

Em um *survey* recente conduzido na pesquisa de Montenegro *et. al* [19], os agentes conversacionais estabelecidos para apoiar médicos e usuários foram, em sua maioria, desenvolvidos com a técnica de Casamento de Padrões.

Similarmente, Paschoal e Souza [24] descrevem que conduziram um mapeamento sistemático sobre agentes conversacionais construídos para apoiar o ensino-aprendizagem e constataram que a técnica de Casamento de Padrões destacou-se dentre as identificadas. Além disso, os autores observaram que a linguagem AIML é utilizada em conjunto com Casamento de Padrões para apoiar a implementação das bases de conhecimento dos agentes conversacionais.

Diante dos resultados descritos nos trabalhos de Abdul-Kader e Woods [1], Montenegro *et. al* [19] e Paschoal e Souza [24], decidiu-se que a técnica de Casamento de Padrões seria escolhida para a construção do agente conversacional.

Os agentes conversacionais baseados em Casamento de Padrões e AIML, em sua maioria, são textuais, isto é, processam mensagens em formato de texto e, com base na mensagem emitida pelo usuário, retornam outra mensagem de texto [34].

Os componentes que caracterizam e/ou constituem os agentes conversacionais desenvolvidos com Casamento de Padrões e AIML são descritos no trabalho de Abdul-Kader e Woods [1], são apresentados na Figura 2.

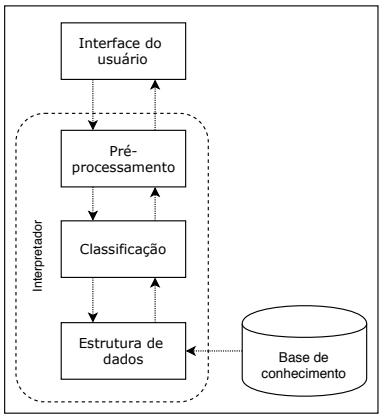

Figura 2: Componentes de um agente conversacional [1].

De modo geral, os agentes conversacionais são constituídos por uma interface do usuário, um interpretador e uma base de conhecimento. A interface é o recurso utilizado pelo usuário para interagir com o agente. Ela permite ao usuário escrever e enviar as mensagens, assim como, fazer a leitura das respostas reproduzidas pelo agente. O interpretador, por sua vez, é composto por mecanismos que realizam um conjunto de atividades (pré-processamento, classificação e busca em estrutura de dados) cujo propósito é processar a linguagem. Essas atividades são descritas a seguir:

- **Pré-processamento:** desempenha o papel de interface entre o classificador e a interface do usuário, controlando as entradas e saídas. É responsável por transferir os dados do usuário para o classificador, bem como controlar os dados que são retornados pelo classificador para a interface do usuário.
- **Classificação:** consiste na aplicação de um filtro e normalização da entrada. As abreviaturas, siglas e símbolos que

foram digitadas pelo usuário são substituídas pelas palavras ou locuções originais. Na sequência, a entrada é particionada em um número x de sentenças, em que x é o número de sentenças que correspondem a entrada. Para identificar o número de sentenças, as pontuações do texto são respeitadas (*e.g.*, ponto final). Além disso, todo tipo de pontuação existente é removida. Por último, todos os caracteres que constituem as sentenças são convertidos em letras maiúsculas.

- **Estrutura de Dados:** consiste em extrair os caracteres do classificador e compará-los com dados organizados em formato de árvore n-ária. Esses dados são pares conversacionais (*i.e.*, perguntas e respostas) que foram previamente escritos em AIML e transformados em ramos e nós da árvore.

O último constituinte é a base de conhecimento. Ela representa o conhecimento finito do agente conversacional sobre um determinado domínio, representando as entradas que o agente conseguirá compreender. A base de conhecimento pode ser escrita em diferentes linguagens. Em razão do projeto utilizar a técnica de Casamento de Padrões, optou-se por utilizar a linguagem AIML para implementá-las. A partir da escrita das bases de conhecimento, quando essas são associadas a um interpretador, os pares conversacionais são transformados em uma estrutura em formato de árvore.

Após o entendimento sobre o funcionamento dos componentes de agentes baseados em Casamento com Padrões, o próximo passo foi delimitar o conhecimento do agente para a implementação das bases de conhecimento. Nessa perspectiva, foram investigadas as iniciativas que mapearam os assuntos sobre teste de software que estão sendo ensinados e também aquelas que buscaram observar os currículos de referência (SBC, ACM e IEEE) que recomendam os assuntos que precisam serem ensinados [23, 32]. Diante disso, foi possível pré-delimitar o conhecimento que o agente deverá possuir nesta primeira etapa do seu desenvolvimento (Tabela 1).

Tabela 1: Domínio de conhecimento definido para o agente TOB-STT

Divisão de tópicos	Conteúdo
Fundamentos de teste	Terminologias relacionadas ao teste (engano, defeito, erro e falha), processos de testes, níveis de teste, atividades estáticas e dinâmicas.
Técnicas de teste	Teste funcional, teste estrutural e teste baseado em defeitos.
Critérios de teste	Particionamento em classes de equivalência, análise do valor limite, teste baseado em complexidade (teste do caminho base), teste baseado em fluxo de controle (todos-nós, todas-arestas, todos-caminhos: simples, completo, livre de laço), teste baseado em fluxo de dados (todas-definições, todos-usos) e teste de mutação.

Com base na delimitação dos conteúdos, os quais também são contemplados por cursos de treinamento de plataformas MOOCs tais como Coursera[1], edX[2] e Udacity[3], foi necessário determinar as tecnologias utilizadas no desenvolvimento do agente conversacional. Em relação a isso, alguns princípios foram considerados:

- Uso de tecnologias de software livre, buscando produzir um agente conversacional *open source*. Com isso busca-se estimular a utilização desse recurso educacional, permitindo que outros pesquisadores e professores possam estudá-lo e adaptá-lo de acordo com as suas necessidades.
- Uso de tecnologias *web*, de modo a facilitar a produção de um agente conversacional facilmente integrável com ambientes de aprendizagem (*e.g.*, ambientes virtuais de aprendizagem, repositórios de materiais, mundos virtuais, dentre outros).

Com base nessas características, foram estudados os interpretadores desenvolvidos para agentes conversacionais que suportam a linguagem AIML (*e.g.*, RebeccaAIML, Program-D, ChatterBean, Program-R, Program-Q, Program-O, PandoraBots, dentre outros). Em uma decisão de projeto, ficou definido que seria utilizado o Program-O [4], uma vez que é um interpretador livre, mantido por uma comunidade ativa no GitHub e é implementado na linguagem PHP (Hypertext Preprocessor), a mesma utilizada nos principais ambientes virtuais de aprendizagem, como o Moodle. Esse interpretador também possibilita gerenciar as bases de conhecimento do agente conversacional, registrar o histórico de interações e oferece uma interface inicial para bate-papo.

Após os estudos realizados e as decisões tomadas durante a primeira fase do projeto, a etapa de desenvolvimento foi executada.

3.2 Desenvolvimento

A etapa de desenvolvimento envolve as quatro atividades, mencionadas no início da Seção 3. Ao final da execução dessas atividades, um artefato é produzido. Em especial, o artefato produzido em cada ciclo de execução consiste em um conjunto de bases de conhecimento. Nesse sentido, considerando os assuntos de teste de software delimitados na Tabela 1, o primeiro ciclo teve a finalidade de estabelecer bases de conhecimento sobre fundamentos de teste de software.

Para desenvolver as bases de conhecimento, definiram-se diferentes padrões de pares conversacionais. Estabeleceu-se que os padrões respeitariam perguntas associadas a conceitos, demonstrações e exemplos. A categoria de conceitos englobaria perguntas de definição (por exemplo: *"What is a fault?"*) e comparação (*"What is the difference between fault and error?"*). A categoria demonstrações, por sua vez, envolveria questões sobre como utilizar um critério de teste para definir o conjunto de casos de teste (por exemplo: *"How do I apply equivalence partitioning criterion?"*). A última categoria engloba conhecimentos associados a exemplos de aplicações de critérios (*"Can you give me a example of equivalence partitioning criterion?"*).

Como o primeiro ciclo de desenvolvimento está associado ao conteúdo teórico, pares conversacionais foram estabelecidos somente para a categoria de conceitos. Um exemplo de um par conversacional dessa categoria é apresentado no Código 1. Conforme é possível observar, a pergunta é definida no comando <pattern> enquanto que a resposta é escrita dentro do comando <template>.

Os pares conversacionais que constituem as bases de conhecimento foram definidos tendo como base o vocabulário da OntoTest – uma ontologia sobre teste de software [4]. Em razão das bases de conhecimento serem implementadas em AIML, foi possível utilizar

[1] http://bit.ly/2MA652N
[2] http://bit.ly/2KoPVvz
[3] http://bit.ly/2XkoDdh

[4] https://github.com/Program-O/Program-O

estruturas da linguagem que permitem criar diálogos complexos, com tratamento de informações em quatro níveis linguísticos: morfossintático, semântico, discursivo e pragmático.

Código 1: Exemplo de um par conversacional implementado em AIML sobre a definição de defeito.

```
<aiml>
<category>
<pattern> whats a fault? </pattern>
<template>
Fault is an incorrect step, process, or data definition
    in computer program. Fault is the formal name of a
    bug.
</template>
</category>
</aiml>
```

Para o tratamento morfológico, foi necessário produzir uma base de substituições de palavras. Essa base de conhecimento é capaz de corrigir erros de digitação e padronizar termos a partir de variações que não são necessariamente um erro. Um exemplo de substituição definido é: <substitute find="software test" replace="software testing"/>. Por meio desse padrão, quando o aluno fizer uma pergunta utilizando as palavras compostas "software test", elas serão substituídas por "software testing".

Outra estrutura que foi usada para oferecer tratamento morfossintático foi o molde de réplica. Essa estrutura permite que o TOB-STT simule diferentes formas de resposta para uma mesma pergunta. Nesse sentido, os comandos <random> e <li> da linguagem AIML foram utilizados. No Código 2 é apresentado um fragmento que foi extraído da base de conhecimento sobre fundamentos de teste de software. No exemplo existem duas definições para a pergunta "what's a fault?". Somente uma delas será sorteada aleatoriamente.

Código 2: Exemplo de par conversacional com molde de réplica.

```
<aiml>
<category>
<pattern> whats a fault? </pattern>
<template>
<random>
<li>  Fault is an incorrect step, process, or data
    definition in computer program. Fault is the formal
    name of a bug. </li>
<li> The fault is the adjudged or hypothesized cause of
    an error. </li>
</random>
</template>
</category>
</aiml>
```

Outro exemplo de tratamento linguístico em nível morfossintático se refere ao tratamento de expressões desconhecidas. Esse tipo de análise é importante para o agente conversacional conseguir oferecer uma resposta para um assunto que não foi previamente definido na base de conhecimento. Além disso, isso permite ao agente demonstrar ao aluno uma noção sobre seus limites e capacidades. Um exemplo é apresentado no Código 3.

O tratamento semântico também foi utilizado durante o desenvolvimento da base de conhecimento sobre fundamentos de teste de software. Nessa perspectiva, as bases de conhecimento do agente foram organizadas tendo como princípio os tópicos definidos na coluna "conteúdo" da Tabela 1, utilizando o comando <topic>. Com os pares conversacionais aninhados aos tópicos, é possível diferenciar o tema da conversação.

Código 3: Exemplo de tratamento de expressões desconhecidas.

```
<aiml>
<category>
<pattern> * </pattern>
<template>
Sorry, I still can't␣answer␣that.␣Could␣you␣ask␣me␣
    differently.
</template>
</category>
</aiml>
```

Em uma conversa entre humanos, quando uma pessoa está questionando a outra sobre um determinado assunto, ela pode utilizar o mesmo padrão de entrada para se referir a outro assunto. Nesse sentido, o tratamento semântico torna-se extremamente relevante. No Código 4, uma citação pertinente de uso é apresentada. Nessa situação, quando o tópico da conversa é teste funcional, por exemplo, o aluno poderá perguntar ao TOB-STT quais são os critérios dessa técnica. A mesma pergunta pode ser feita quando o tópico da conversa for teste estrutural.

Código 4: – Exemplo de definição de tópicos de conversação

```
<aiml>
<topic name="␣BlackBox">
<category>
<pattern> What are the criteria of the test technique? </
    pattern>
<template>
equivalence partitioning, boundary-value analysis, cause-
    effect graphing and error guessing
</template>
</category>
</topic>
<topic name="␣WhiteBox">
<category>
<pattern> What are the criteria of the test technique? </
    pattern>
<template>
statement coverage, decision coverage, condition coverage
</template>
</category>
</topic>
</aiml>
```

No contexto das bases de conhecimento do TOB-STT, também foi utilizada uma estrutura da linguagem AIML para oferecer tratamento linguístico no nível discursivo. Essa estrutura é similar a estrutura utilizada na substituição de palavras. Nesse sentido, foi definida uma base de substituição de pronomes, por meio dos comandos <person> e <substitute> (Código 5).

Código 5: – Exemplo de comandos utilizados no tratamento de informação em nível discursivo

```
<aiml>
<person>
<substitute find= she is replace= he is />
</person>
</aiml>
```

O último nível de tratamento linguístico contemplado pelas bases de conhecimento do agente é o nível pragmático. Ele diz respeito ao uso de variáveis que armazenam temporariamente informações sobre o usuário que está interagindo com o TOB-STT. As variáveis são definidas pelo comando <set> e recuperadas por <get>, inseridos nos padrões das categorias da base de conhecimento. Salienta-se que o nível pragmático não foi adotado especificamente nas bases de conhecimento sobre o tópico selecionado, mas em bases de interação.

Durante o desenvolvimento do TOB-STT, foi necessário considerar que a conversação é uma atividade estruturada (*i.e.*, a conversa é dividida em três fases distintas: início da interação, desenvolvimento e fim da interação [5]). Nessa perspectiva, bases de interação precisavam ser planejadas e escritas. Visando concentrar os esforços no estabelecimento das bases de conhecimento associadas ao domínio, optou-se por reutilizar bases de conhecimento de interação do agente conversacional ALICE [34], um sistema de conversação que possui as bases de conhecimento sob licença livre.

A opção por reusar as bases de conhecimento de início de interação e fim de interação do agente ALICE foi inspirada no trabalho de Herpich *et al.* [12]. Em razão de se fazer reúso das bases de conhecimento de interação, o agente TOB-STT recebeu um tratamento linguístico pragmático. Um fragmento de uma base de conhecimento é exposta no Código 6, em que o aluno menciona o seu nome ao TOB-STT e o agente armazena o nome do aluno.

Código 6: – Exemplo de comandos utilizados no tratamento de informação em nível discursivo

```
<aiml>
<category>
<pattern>My name is *</pattern>
<template> Hello <set name = "username"> <star/>! </set>
</template>
</category>
<category>
<pattern>Bye </pattern>
<template>
Bye <get name = "username"/>. See you
</template>
</category>
</aiml>
```

Após a implementação das bases de conhecimento associadas ao tópico de fundamentos de teste de software, a próxima atividade da fase do gerenciamento consistiu na revisão das bases de conhecimento. Nessa atividade um especialista, com mais de 20 anos de experiência em ensino e pesquisa na área de teste de software, foi convidado a examinar o conteúdo das bases de conhecimento que foram implementadas na atividade anterior.

O especialista convidado não havia participado da implementação das bases e foi requisitado para conferir o conteúdo. Ele recebeu um treinamento sobre a linguagem de marcação AIML, para facilitar a verificação. Além disso, para facilitar a posterior correção das bases de conhecimento, ele foi notificado que deveria fazer comentários no código.

Buscou-se utilizar o especialista que não estava envolvido na codificação para reconhecer problemas e inconsistências nas bases de conhecimentos. Assim, ao final da revisão do especialista, seria possível obter alguns esclarecimentos, tais como (i) quais informações estão inconscientes e (ii) quais informações estão em falta, mas deveriam estar presentes.

Após a verificação do especialista, ajustes foram realizados nas bases de conhecimento, visando adequá-las, em conformidade com as recomendações.

A última atividade da fase de desenvolvimento consiste na análise de viabilidade (apresentada na Seção 4). Esse tipo de avaliação tem a finalidade de gerar conhecimento sobre a aplicação da tecnologia que foi desenvolvida. Para apoiar a análise de viabilidade, a metodologia experimental para introdução de tecnologias de software na indústria definida por Shull *et al.* [30] foi utilizada.

Antes de conduzir o estudo de viabilidade foi necessário implementar uma interface para o agente conversacional. Como o Program-O oferece uma interface baseada HTML 5 com AJAX (*Asynchronous Javascript and XML*), foram feitas algumas adaptações na interface para adequá-la ao contexto do projeto. Essa interface permite que o usuário digite mensagens textuais e tem um campo para o usuário visualizar a resposta do agente e o histórico da interação. A Figura 3 ilustra a interface do TOB-STT.

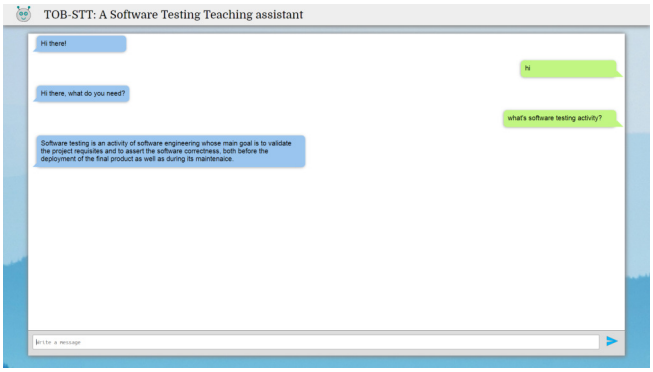

Figura 3: Interface do TOB-STT

3.3 Encerramento

A última fase do processo de gerenciamento é o encerramento. Essa fase é praticada quando todas as bases de conhecimento do agente TOB-STT estiverem implementadas. Quando isso ocorrer, espera-se fazer experiências que consideram o agente conversacional como mecanismo de apoio ao aprendizado de teste de software, em cursos MOOCs e em implementações de diferentes modelos pedagógicos (*e.g., flipped classroom, team-based learning*).

Além da condução de estudos, o agente conversacional será hospedado em uma página *web* para ser utilizado por professores e alunos que têm interesse em utilizar o sistema. O código-fonte completo será disponibilizado no Git-Hub, em razão do agente conversacional estar sendo desenvolvido com base na abordagem de software livre. Salienta-se que as bases desenvolvidas no primeiro ciclo já se encontram disponíveis no link: <http://bit.ly/2ZkWQdJ>.

4 ESTUDO DE VIABILIDADE

4.1 Planejamento

Buscando compreender se as bases de conhecimento sobre fundamentos de teste de software são viáveis para o propósito do agente conversacional (*i.e.*, solucionar as dúvidas dos alunos de teste de software), foi feito um estudo de viabilidade com estudantes de graduação em computação, dos cursos de bacharelado em Ciências de Computação e Engenharia de Computação, matriculados em uma disciplina de Teste e Inspeção de Software.

Este estudo de viabilidade foi realizado em Abril de 2019, no período em que o conteúdo associado ao tópico de fundamentos de teste de software já havia sido ministrado (há aproximadamente 45 dias). Neste sentido, os alunos possuíam um conhecimento preliminar sobre o conteúdo. Nesse ínterim, eles estavam aprendendo Inspeção de Software, em especial, técnicas de inspeção em código-fonte e em documento de requisitos. Em razão disso, buscando aproveitar o tema que os alunos estavam estudando, estabeleceu-se uma atividade que envolvia a inspeção de um documento.

A atividade envolveu conceitos e definições de terminologias relacionadas à atividade de teste, processos de testes, níveis de teste e atividades estáticas e dinâmicas. Ela foi planejada de modo que os alunos tivessem que interagir com o TOB-STT para esclarecer dúvidas. Buscando assegurar que os alunos utilizassem o agente conversacional, foi decidido que os alunos saberiam da atividade somente no momento da aula. Assim, a atividade envolveria o conhecimento retido pelos alunos a partir de aulas anteriores.

Para a atividade, um documento com uma argumentação sobre a importância do teste de software foi produzido. Esse documento englobou os conceitos e as definições dos assuntos que fazem parte dos fundamentos de teste. Defeitos foram introduzidos no documento. Os defeitos foram inseridos, uma vez que o propósito da atividade foi colocar os alunos no papel de inspetores.

Os alunos deveriam inspecionar o documento em duplas, de modo que pudessem discutir sobre os defeitos identificados no documento. Como os alunos estavam aprendendo o conteúdo sobre inspeção de software, eles ainda não haviam aprendido as técnicas de leitura de documento. Então, não foi recomendado ou exigido o uso de técnicas para leitura. A única recomendação realizada foi de que cada aluno adotasse sua própria maneira de ler o documento e, posteriormente realizasse uma discussão.

Ao final da atividade, os alunos deveriam entregar uma lista com os defeitos identificados. Ainda, após a entrega, os alunos deveriam emitir um *feedback* sobre suas percepções a partir do uso do TOB-STT. De maneira similar à atividade, o *feedback* deveria ser emitido pela dupla.

Um questionário foi preparado para os alunos emitirem suas percepções sobre o uso do TOB-STT. Esse questionário foi estabelecido tendo como base os instrumentos desenvolvidos e utilizados nas pesquisas de Paschoal *et al.* [21] e Herpich *et al.* [12]. O questionário é composto por nove assertivas, sendo que todas são baseadas em uma escala Likert de cinco pontos, cujas opções de respostas podem variar de "Discordo Totalmente" a "Concordo Totalmente". São elas:

(1) Ao interagir com o TOB-STT pela primeira vez, a experiência não foi animadora.
(2) As respostas fornecidas pelo TOB-STT foram sobre o tópico questionado.
(3) O TOB-STT não soube responder alguma pergunta realizada.
(4) Ao utilizar o TOB-STT, eu consegui obter o conhecimento pretendido.
(5) O TOB-STT não forneceu informações confiáveis em suas respostas.
(6) O TOB-STT contribuiu para a realização da tarefa.
(7) O TOB-STT demorou para fornecer as respostas.
(8) O TOB-STT possui uma interface fácil de usar.
(9) Eu fiquei insatisfeito com o TOB-STT.

Para complementar o questionário, baseando-se na pesquisa de AbuShawar e Atwell [2], uma análise sobre a qualidade das respostas produzidas pelo TOB-STT foi planejada. Para tanto, foi idealizado que todas as interações realizadas entre os alunos com o agente conversacional fossem registradas. Em razão do Program-O ter um mecanismo direcionado ao armazenamento de histórico de conversas, e esse ser o interpretador utilizado pelo TOB-STT, não foi necessário nenhum preparo adicional (*i.e.*, escrita de algoritmo para armazenamento de *log*).

Além da atividade, outros documentos foram elaborados. Dentre eles, uma apresentação sobre o TOB-STT, que contém detalhes e explicações sobre como utilizar o agente. Um conjunto de *slides* sobre inspeção de documentos de requisito foi preparado. O preparo desses documentos foi necessário porque eles seriam utilizados durante um treinamento que foi projetado para que os alunos recebessem algumas informações e conhecimentos antes da realização da atividade.

O treinamento projetado refere-se a uma aula de 60 minutos que busca apresentar uma visão geral sobre inspeção de software, mais especificamente, sobre estratégias para inspecionar documentos de requisitos. Nesse treinamento, os alunos teriam uma formação mais específica sobre como classificar os defeitos identificados no documento, considerando a classificação proposta no trabalho de Shull [29].

Além do material para o treinamento, foi necessário produzir um Termo de Consentimento Livre e Esclarecimento, que contém detalhes sobre o estudo de viabilidade (*i.e.*, objetivo, detalhes de execução) e tem a finalidade de esclarecer aos alunos todos os possíveis benefícios, riscos e procedimentos que serão realizados no estudo. A redação desse documento foi necessária, porque buscou-se fornecer aos participantes todas as informações pertinentes à pesquisa.

Após o planejamento do estudo e preparação do material, a professora responsável pela disciplina foi convidada a revisar o material produzido, dado que não se tinha a intenção de oferecer um material não adequado aos alunos da disciplina. Na sequência, ajustes foram realizados e o estudo de viabilidade foi conduzido/executado.

4.2 Execução

A execução do estudo ocorreu conforme o planejamento. Nesse sentido, em uma das aulas da disciplina de Teste e Inspeção de Software, a professora explicou aos alunos que durante a aula, uma atividade com o uso de um agente conversacional seria realizada. Para tanto, os alunos necessitariam aprender a técnica de inspeção de documento de requisitos e compreender como classificar os defeitos.

A professora explicou que a atividade seria realizada após o treinamento sobre inspeção e argumentou que a participação de

todos os alunos era importante. A professora também explicou que a participação deveria ser feita de forma voluntária (*i.e.*, os estudantes poderiam optar por participar ou não do estudo) e que a atividade teria uma duração de aproximadamente 90 minutos.

A aula foi ministrada e os alunos puderam fazer questionamento e solucionar as dúvidas. Na sequência, foi feita uma demonstração sobre como fazer perguntas ao TOB-STT e como utilizar as respostas geradas pelo agente. Após a apresentação do agente, os alunos foram informados que deveriam formar duplas. Posteriormente, os alunos foram notificados que deveriam fazer a leitura do documento, discutir com o parceiro da dupla e interagir com o TOB-STT para solucionar as dúvidas sobre o conteúdo do documento. Com base na interação, defeitos encontrados deveriam ser registrados apropriadamente.

Os estudantes realizaram a atividade conforme solicitado, interagindo com o TOB-STT e entregando no final a lista de defeitos identificados. Na medida em que os alunos foram entregando o documento, eles foram informados que deveriam emitir um *feedback* sobre suas percepções do agente conversacional, respondendo o questionário logo após o término da atividade.

4.3 Resultados

Após a condução do estudo, os dados foram tabulados e analisados. Ao total, foram considerados os dados de 30 alunos, uma vez que somente essa quantidade concordou em participar do estudo e realizou todas as atividades previstas (*i.e.*, realizaram exercício e emitiram o *feedback* por meio do questionário).

Inicialmente, tendo a finalidade de observar o desempenho dos alunos na atividade proposta, uma análise a respeito da eficácia dos alunos em reconhecer os defeitos injetados na especificação foi realizada. Nessa perspectiva, os exercícios foram corrigidos tendo como base um oráculo definido pelos pesquisadores. Assim, a eficácia foi calculada tendo como base o número defeitos corretos em relação ao número de defeitos que era esperado pelo oráculo.

Os resultados obtidos a partir da correção do exercício e cálculo da eficácia estão representados na Figura 4. Com base no *box plot*, é possível observar que a maioria dos alunos conseguiu identificar entre 40% e 50% dos defeitos que foram injetados. Em média, a eficácia dos alunos no reconhecimento dos defeitos ficou em torno de 49,44%. A mediana, por sua vez, ficou em exatamente 50%. Esses resultados representam valores não esperados, uma vez que demonstram que os alunos não conseguiram identificar mais da metade dos defeitos presentes no documento. Diante disso, é possível levantar algumas hipóteses: (i) o agente conversacional não conseguiu ajudar os alunos a relembrar o conteúdo; (ii) o conhecimento retido pelos alunos era insuficiente para eles realizarem a atividade; (iii) a condução da atividade em dupla pode ter atrapalhado a realização do exercício (*i.e.*, os alunos da dupla não conseguiram entrar em um consenso sobre os defeitos).

Ainda em relação à Figura 4, alguns *outliers* foram identificados. Esses *outliers* representam os grupos de alunos que conseguiram identificar 90% dos defeitos inseridos no documento e o grupo de alunos que reconheceu apenas 20% dos defeitos.

Após a análise dos resultados plotados na Figura 4, foram verificadas as respostas emitidas pelos alunos no questionário, afim de compreender qual foi a percepção dos alunos sobre o uso do agente

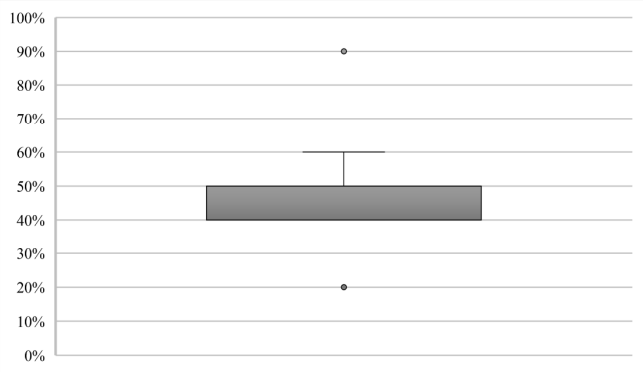

Figura 4: Eficácia dos alunos em identificar os defeitos

conversacional, mais especificamente, tentar averiguar se os alunos conseguiram a ajuda que precisavam com o agente conversacional. As respostas obtidas estão disponíveis na Tabela 2.

Tabela 2: Resultado das respostas dos alunos

	Discordo totalmente	Discordo parcialmente	Indeciso	Concordo parcialmente	Concordo totalmente
Assertiva 1	6,7%	33,3%	33,3%	20%	6,7%
Assertiva 2	0%	6,7%	33,3%	33,3%	26,7%
Assertiva 3	0%	6,7%	6,7%	26,7%	60%
Assertiva 4	6,7%	20%	26,7%	33,3%	13,3%
Assertiva 5	60%	26,7%	6,7%	0%	6,7%
Assertiva 6	0%	13,3%	33,3%	33,3%	20%
Assertiva 7	100%	0%	0%	0%	0%
Assertiva 8	0%	6,7%	6,7%	20%	66,7%
Assertiva 9	33,3%	46,7%	13,3%	6,7%	0%

Em relação à primeira assertiva, não foi possível constatar uma predominância nas respostas em qualquer um dos pontos da escala Likert. Se as respostas dos estudantes forem observadas por classe (discordância, indecisão e concordância), nota-se que há mais respostas no seguimento de discordância, o que pode simbolizar que os alunos acharam a experiência animadora.

A segunda assertiva dá indícios de que as respostas oferecidas pelo TOB-STT foram relacionadas aos tópicos que os alunos questionaram. Já em relação a terceira assertiva, os resultados expressam que o TOB-STT não soube responder algumas perguntas que foram realizadas pelos estudantes. Esses resultados podem explicar, por exemplo, a indecisão dos alunos ao responderem a primeira assertiva do questionário.

Os resultados associados à quarta assertiva são similares ao obtidos na primeira. Nesse sentido, apesar de uma porcentagem de alunos revelar que conseguiu obter o conhecimento pretendido a partir da utilização do TOB-STT, houveram expressivas respostas que transmitiram indecisão dos alunos ao assinalar as opções de resposta dessa assertiva.

Na quinta assertiva, notou-se que a maioria dos estudantes considerou confiável as informações emitidas pelo TOB-STT. De maneira similar, os resultados da sexta assertiva também indicam que mais de 50% dos estudantes sinalizaram que o TOB-STT contribuiu para a realização da atividade.

Durante a sétima assertiva, os alunos tiveram que indicar se o TOB-STT demorou para responder as perguntas que eles faziam. Conforme é possível observar na Tabela 2, todas as respostas indicam que os alunos ficaram satisfeitos com a agilidade do agente em responder às suas perguntas.

Na penúltima assertiva, constatou-se que mais de 80% dos alunos achou fácil usar a interface do TOB-STT. Por fim, a nona assertiva buscou compreender se os alunos ficaram satisfeito com o agente conversacional. Com base nos resultados, nota-se que a maioria dos estudantes indicou que ficou contente.

Após a análise dos resultados, foi possível observar que de modo geral, os resultados foram positivos. Observando as respostas obtidas na nona assertiva, por exemplo, é possível acreditar que as bases de conhecimento estabelecidas no primeiro ciclo de desenvolvimento podem ser consideradas satisfatórias, atendendo o propósito para o qual foram estabelecidas.

Procurando compreender a qualidade das respostas emitidas pelo TOB-STT, foi conduzida uma análise nos registros das interações feitas pelos participantes. A análise da qualidade das respostas foi realizada tendo como base o *framework* proposto na pesquisa de AbuShawar e Atwell [2]. Esse *framework* estipula que é necessário fazer uma classificação de cada resposta emitida pelo agente conversacional levando em consideração a pergunta feita pelo aluno. Assim, as respostas do agente foram classificadas em uma dentre quatro categorias:

- **Ideal:** a resposta está presente nas bases de conhecimento do agente conversacional e, por consequência, o agente respondeu a pergunta corretamente;
- **Ok:** a resposta não está presente nas bases de conhecimento do agente conversacional e, por isso, o agente informou ao usuário que não possui conhecimento sobre o assunto;
- **Problemática:** a resposta não está presente nas bases de conhecimento do agente conversacional e o agente apresentou uma resposta incorreta para a pergunta;
- **Erro:** a resposta está presente nas bases de conhecimento do agente conversacional, mas o agente não conseguiu localizar a resposta correta.

Os resultados da classificação são apresentados na Figura 5. Nota-se que 259 perguntas feitas pelos alunos foram respondidas corretamente pelo agente. Os alunos que interagiram com o TOB-STT fizeram 188 perguntas que não estavam previstas e por conta disso foram notificados que o agente não possuía conhecimento sobre o assunto. Por outro lado, notou-se que o TOB-STT respondeu 133 perguntas de forma inadequada. Dentre as 133 respostas reproduzidas pelo TOB-STT, 81 deveriam notificar o usuário que o agente não conseguiria responder a pergunta e o usuário deveria reescrevê-la de outra forma e 52 respostas estão relacionadas com perguntas que eram esperadas pelo agente, mas foram respondidas incorretamente pelo agente.

Com base nos resultado apresentados na Figura 5, foi possível observar que os alunos realizaram uma quantidade significativa de perguntas que não abrangiam o escopo do exercício e, consequentemente das bases de conhecimento definidas no primeiro ciclo de desenvolvimento. Percebeu-se que os alunos tendem a explorar o agente, fazendo perguntas que não tem relação com o conteúdo. Em

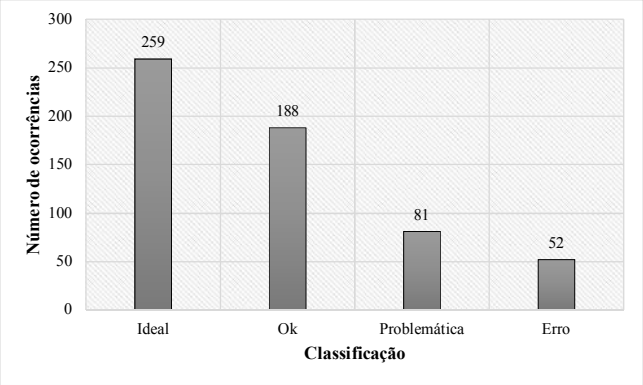

Figura 5: Proporção de cada tipo de resposta identificada

razão disso, o TOB-STT respondeu aos alunos que não tinha conhecimento sobre o questionamento do aluno. Esse tipo de resultado, é interessante ao passo que pode ser usado como um indicativo para explicar algumas das respostas expressas pelos alunos no questionário. Notou-se, conforme já mencionado, que alguns alunos revelaram que não ficaram satisfeitos com a experiência de uso do agente conversacional. Essa insatisfação pode ter relação com o fato do aluno não ter feito as perguntas associadas ao escopo.

Ainda, é possível que a análise da qualidade de respostas possa também dar um sinal sobre o porquê dos alunos não terem conseguído obter um bom desempenho na atividade de inspeção. Como os alunos realizaram perguntas fora do contexto, é possível que eles não tenham dado atenção ao exercício e, preferiram atentar-se ao uso do agente. Essa seria uma provável explicação para a baixa eficácia dos alunos em identificar os defeitos que estavam presentes no documento.

Além disso, foi possível observar que o TOB-STT não conseguiu responder algumas perguntas que deveria responder. Isso também pode ter prejudicado o desempenho dos alunos na atividade e refletido no descontentamento dos mesmos durante a emissão do *feedback* apresentada no questionário.

Após a condução do estudo de viabilidade, algumas pesquisas precisam ser realizadas. Em especial, tentar compreender o porquê do agente conversacional ter apresentado respostas incorretas aos alunos quando respostas corretas estavam presentas nas suas bases de conhecimento. Também é necessário averiguar por que o agente apresentou respostas aleatórias aos participantes quando deveria informar que ele ainda não tinha conhecimento sobre o assunto. Vale ressaltar que a comunidade de desenvolvimento e manutenção do Program-O está fazendo aperfeiçoamentos no interpretador e pode ajudar nessas questões.

Para contribuir com pesquisas futuras, um pacote de laboratório foi preparado: <http://bit.ly/32ZTZcn>. Nele estão disponíveis os formulários, materiais usados no treinamento e os dados que foram coletados. Além disso, as ameças à validade estão listadas, com as ações que foram tomadas para mitigá-las.

5 CONCLUSÕES

Nos últimos anos, a comunidade de computação tem incentivado o uso de diferentes modelos pedagógicos no ensino de teste de

software. Além disso, cursos de formação continuada estão sendo produzido em plataformas MOOCs. De modo geral, há uma concentração de esforços em oferecer subsídios à formação em teste de software. Apesar disso, os resultados desses esforços apresentam algumas limitações. Para tratá-las, os agentes conversacionais são descritos como alternativas para solucionar algumas dessas limitações.

A falta de um agente conversacional para o ensino de teste de software motivou a realização deste trabalho. Assim, este estudo apresentou esforços em direção ao estabelecimento do agente conversacional TOB-STT e resultados de um estudo inicial sobre a viabilidade do agente como um mecanismo para resolver as dúvidas dos alunos de teste de software. Com base nesse estudo, notou-se que melhorias precisam ser feitas nas bases de conhecimento, uma vez que foram identificados problemas durante a análise da qualidade das resposta. Nessa perspectiva, novos pares conversacionais precisam ser estabelecidos. Uma oportunidade de pesquisa é investir em redes neurais para permitir um aprendizado por parte do agente conversacional. Com isso, espera-se que o agente tenha uma base de conhecimento ampliada e maiores chances de retornar para o usuário uma resposta adequada ou muito próxima do que é esperada.

O estudo de viabilidade ainda ofereceu uma visão sobre as percepções dos alunos. Com base nelas e na análise das interações, constatou-se que é possível que haja uma tendência entre os alunos em despender a atenção no agente conversacional ao invés de focar na atividade, o que pode se tornar um problema. Pesquisas nesse sentido podem se tornar necessárias à medida em que dados mais concretos sobre o uso do agente conversacional forem obtidos.

wow... you introduced it!

AGRADECIMENTOS

Os autores gostariam de agradecer a CAPES - Código de Financiamento 001, a FAPESP (Processo 2018/16882-6, Processo 2017/10941-8 e Processo 2013/07375-0) e ao CNPq (Processo 311494/2017-0 e Processo 312922/2018-3) pelo apoio financeiro.

REFERÊNCIAS

[1] S. A Abdul-Kader e J. C. Woods. 2015. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6, 7, 72–80.

[2] B. AbuShawar e E. Atwell. 2016. Usefulness, localizability, humanness, and language-benefit: additional evaluation criteria for natural language dialogue systems. *International Journal of Speech Technology*, 19, 2, 373–383.

[3] C. C. Aguirre, C. D. Kloos, C. Alario-Hoyos e P. J. Muñoz-Merino. 2018. Supporting a mooc through a conversational agent. design of a first prototype. Em *International Symposium on Computers in Education*, 1–6.

[4] E. F. Barbosa, E. Y. Nakagawa e J. C. Maldonado. 2006. Towards the establishment of an ontology of software testing. Em *18th International Conference on Software Engineering & Knowledge Engineering*. San Francisco, USA.

[5] F. Barros e P. Tedesco. 2016. Agentes inteligentes conversacionais: conceitos básicos e desenvolvimento. Em *35ª Jornada de Atualização em Informática - CSBC*. Porto Alegre, Rio Grande do Sul, Brasil.

[6] L. Benotti, M. C. Martnez e F. Schapachnik. 2018. A tool for introducing computer science with automatic formative assessment. *IEEE Transactions on Learning Technologies*, 11, 2, 179–192.

[7] A. Bertolino. 2007. Software testing research: achievements, challenges, dreams. Em *Future of Software Engineering*, 85–103.

[8] A. C. Dias-Neto, S. Matalonga, M. Solari, G. Robiolo e G. H. Travassos. 2017. Toward the characterization of software testing practices in south america: looking at brazil and uruguay. *Software Quality Journal*, 25, 4, 1145–1183.

[9] A. G. O. Fassbinder, M. Fassbinder, E. F. Barbosa e G. D. Magoulas. 2017. Massive open online courses in software engineering education. Em *IEEE Frontiers in Education Conference*, 1–9.

[10] G. Fraser, A. Gambi, M. Kreis e J. M. Rojas. 2019. Gamifying a software testing course with code defenders. Em *50th ACM Technical Symposium on Computer Science Education*. ACM, 571–577.

[11] Z. Guo e T. Inoue. 2019. Using a conversational agent to facilitate non-native speaker's active participation in conversation. Em *Conference on Human Factors in Computing Systems*, LBW1216:1–LBW1216:6.

[12] F. Herpich, F. B. Nunes, G. B. Voss e R. D. Medina. 2016. Three-dimensional virtual environment and npc: a perspective about intelligent agents ubiquitous. Em IGI Global, Hershey, EUA, 510–536.

[13] A. Kerry, R. Ellis e S. Bull. 2009. Conversational agents in e-learning. Em *Applications and Innovations in Intelligent Systems XVI*. T. Allen, R. Ellis e M. Petridis, editores. Springer London, London, 169–182.

[14] P. Lauvås e A. Arcuri. 2018. Recent trends in software testing education: a systematic literature review. Em *The Norwegian Conference on Didactics in IT education*, 1–11.

[15] O. A. L. Lemos, F. F. Silveira, F C. Ferrari e A. Garcia. 2018. The impact of software testing education on code reliability: an empirical assessment. *Journal of Systems and Software*, 137, 1, 497–511.

[16] M. D. Leonhardt, L. Tarouco, R. M. Vicari, E. R. Santos e M. S. da Silva. 2007. Using chatbots for network management training through problem-based oriented education. Em *7th International Conference on Advanced Learning Technologies*. Niigata, Japan, 845–847.

[17] F. A. Mikic-Fonte, M. Llamas-Nistal e M. Caeiro-Rodríguez. 2018. Using a chatterbot as a faq assistant in a course about computers architecture. Em *IEEE Frontiers in Education Conference*, 1–4.

[18] A. Mittal, L. Vigentini, M. Djatmiko, G. Prusty, Y. Sharma e M. E. King. 2018. Mooc-o-bot: using cognitive technologies to extend knowledge support in moocs. Em *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, 69–76.

[19] J. L. Z. Montenegro, C. A. Costa e R. R. Righi. 2019. Survey of conversational agents in health. *Expert Systems with Applications*, 129, 56–67.

[20] G. J. Myers, C. Sandler e T. Badgett. 2011. *The Art of Software Testing*. Wiley Publishing, Hoboken, Nova Jersey, EUA.

[21] L. N. Paschoal, M. M. de Oliveira e P. M. M. Chicon. 2018. A chatterbot sensitive to student's context to help on software engineering education. Em *XLIV Conferencia Latinoamericana de Informática*, 1–10.

[22] L. N. Paschoal, L. R. Silva e S. R. S. Souza. 2017. Abordagem flipped classroom em comparação com o modelo tradicional de ensino: uma investigação empírica no âmbito de teste de software. Em *XXVIII Simpósio Brasileiro de Informática na Educação*, 476–485.

[23] L. N. Paschoal e S. R. S. Souza. 2018. A survey on software testing education in brazil. Em *17th Brazilian Symposium on Software Quality*. ACM, 334–343.

[24] L. N. Paschoal e S. R. S. Souza. 2018. Como ensinar teste de software com flipped classroom? Em *VIII Workshop on Thesis and Dissertations of CBSoft - IX Brazilian Conference on Software: Theory and Practice*, 46–52.

[25] L. N. Paschoal e S. R. S. Souza. 2018. Planejamento e aplicação de flipped classroom para o ensino de teste de software. *RENOTE - Revista Novas Tecnologias na Educação*, 16, 2, 1–10.

[26] L. Rodríguez-Gil, J. García-Zubia, P. Orduña, A. Villar-Martinez e D. López-De-Ipiña. 2019. New approach for conversational agent definition by non-programmers: a visual domain-specific language. *IEEE Access*, 7, 5262–5276.

[27] L. P. Scatalon, M. L. Fioravanti, J. M. Prates, R. E. Garcia e E. F. Barbosa. 2018. A survey on graduates' curriculum-based knowledge gaps in software testing. Em *IEEE Frontiers in Education Conference*, 1–8.

[28] S. Schlögl, G. Doherty e S. Luz. 2015. Wizard of oz experimentation for language technology applications: challenges and tools. *Interacting with Computers*, 27, 6, 592–615.

[29] F. J. Shull e V. R. Basili. 1998. *Developing techniques for using software documents: a series of empirical studies*. Tese de doutorado. research directed by Dept. of Computer Science.University of Maryland ...

[30] F. Shull, J. Carver e G. H. Travassos. 2001. An empirical methodology for introducing software processes. Em *8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. Vienna, Austria, 288–296.

[31] A. Soska, J. Mottok e C. Wolff. 2016. An experimental card game for software testing: development, design and evaluation of a physical card game to deepen the knowledge of students in academic software testing education. Em *7th IEEE Global Engineering Education Conference*, 576–584.

[32] P. H. D. Valle, E. F. Barbosa e J. C. Maldonado. 2015. Cs curricula of the most relevant universities in brazil and abroad: perspective of software testing education. Em *XVII International Symposium on Computers in Education*. Setubal, Portugal, 62–68.

[33] P. H. D. Valle, E. F. Barbosa e J. C. Maldonado. 2015. Um mapeamento sistemático sobre ensino de teste de software. Em *XXVI Simpósio Brasileiro de Informática na Educação*, 71–80.

[34] R. Epstein, G. Roberts e G. Beber, editores. 2009. *The anatomy of a.l.i.c.e. Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. Springer Netherlands, Dordrecht, 181–210.