C2-P2: Uma abordagem baseada em Chatbots para navegação de coleções de padrões

RAFAEL TOFOLI SEREICIKAS, Universidade Federal de São Carlos
ANATHAN TELLES PEREIRA, OPUS Software
VITOR PACHECO, Universidade Federal de São Carlos
LUCIANA A.M. ZAINA, Universidade Federal de São Carlos
EDUARDO GUERRA, Free University of Bolzen-Bolzano

Os chatbots são programas de computador capazes de estabelecer uma conversa com os usuários, e com eles o desenvolvimento de sistemas de atendimento a consultas tem sido alvo de vários trabalhos de pesquisa. Neste artigo apresentamos C2-P2, um chatbot capaz de entender as necessidades de um usuário e recomendar um padrão de projeto que atenda-as. O C2-P2 possui uma árvore de decisões que é percorrida durante a troca de mensagens com o usuário, e suas decisões são todas baseadas nas informações dadas pelo usuário em resposta a questionamentos feitos pelo ChatBot, com o intuito de entender qual a necessidade do projeto do usuário. Com a árvore de decisões completa, o chatbot é capaz de identificar um padrão, dentre todos que estão armazenados em seu repositório de padrões, que atenda as necessidades do usuário. A partir da identificação do padrão o chatbot traz informações de como aplicá-lo, quais são suas implicações e exemplos de uso.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: chatbot, interactive conversation, searching patterns

ACM Reference Format:
Rafael Tofoli Sereicikas, Anathan Telles Pereira, Vitor Pacheco, Luciana A.M. Zaina, and Eduardo Guerra. 2022. C2-P2: Uma abordagem baseada em Chatbots para navegação de coleções de padrões. In *SugarLoaf PLoP '22*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUÇÃO

Nos últimos anos, os chatbots têm desempenhado um papel de destaque como interfaces humano-computador. Os chatbots são programas de computador capazes de estabelecer uma conversa com os usuários, por meio de mensagens escritas ou comandos vocais. Esses programas proporcionam conversas de uma forma simples para os usuários; eles podem gerenciar automaticamente grandes volumes de usuários, e muitas vezes substituindo operadores humanos, que nem sempre estão disponíveis. Chatbots são uma ferramenta que pode ser encontrada em muitas plataformas famosas de conversação como WhatsApp, Facebook Messenger, and WeChat, além de que as assistentes digitais, que cada dia vem sendo mais utilizadas, como Amazon Alexa, Apple Siri, and Google Assistant [8], possuem um sistema de Chatbot integrado nelas.

Outro tópico abordado neste trabalho são os Repositórios de Padrões. Repositórios de Padrões são utilizados com o propósito de armazenar padrões de projeto e torná-los de fácil acesso aos desenvolvedores. Estudos encontrados na literatura feitos durante o Plop2015 [4] identificaram grandes desafios da comunidade de padrões quanto ao uso destes repositórios. Dentre esses desafios a busca pelos padrões que se encaixam com a necessidade do projeto apenas utilizando palavras chaves e os recursos atuais de pesquisa.

Trabalhos envolvendo a relação entre ChatBots e Padrões podem ser encontrados na literatura, em sua maior parte são trabalhos voltados para o desenvolvimento de ChatBots utilizando a aplicação de padrões de conversação com usuários[4]. Neste artigo é explorada uma relação entre os tópicos citados que não foi encontrada previamente na literatura, o uso de um ChatBot como um sistema de recomendação de padrões, assim facilitando a busca de padrões, identificando os que se adequem às necessidades do usuário descritas durante a troca de mensagens.

Esse trabalho apresenta um sistema de ChatBot, nomeado C2-P2 (*Conversational Chatbot for Patterns Practitioners*), que foi desenvolvido para funcionar como um sistema de recomendações de padrões. O C2-P2, a partir de troca de mensagens, identifica quais as necessidades do projeto apresentado pelo usuário e recomenda o uso de um padrão de projeto que possa ser aplicado na situação apresentada. O C2-P2 funciona com uma árvore de decisão, que é representada na prática por questionamentos que o ChatBot faz ao usuário. A partir das respostas dos questionamentos, o ChatBot armazena dados que são utilizadas para percorrer a árvore de decisão.

2 FUNDAMENTOS E TRABALHOS RELACIONADOS

Os chatbots, também conhecidos como agentes de conversação, são programas capazes de simular e reproduzir uma conversa inteligente com humanos. Eles são ferramentas que devem fornecer respostas às mensagens dos usuários, e com eles o desenvolvimento de sistemas de atendimento a consultas tem sido alvo de vários trabalhos de pesquisa.[5] Na prática, a interação do usuário com o chatbot é dividida em frases curtas e simples, assumindo que cada frase está semanticamente relacionada com a outra.[2]

Como existem vários chatbots já desenvolvidos, podemos distingui-los em duas categorias: os baseados em recuperação e os de modelo generativo. Os chatbots baseados em recuperação vêm com um conjunto de respostas escritas para minimizar erros gramaticais, melhorar a coerência. Os chatbots baseados em recuperação são mais adequados para sistemas de domínio fechado, que são criados para resolver especificamente problemas recorrentes simples [6].

Embora o uso de ChatBots seja muito pequeno comparado a aplicações web e aplicações mobile, sua presença vem sendo cada vez mais comum. A razão é que esse sistema de troca de mensagens oferece uma experiência única ao usuário além de serem de fácil instalação.[3]

Padrões são soluções repetíveis para problemas recorrentes. Repositório de padrões, que contém padrões, são repositórios que buscam tornar o sistema mais apto a mudanças além de auxiliar o designer/desenvolvedor no fluxo de trabalho e na definição de fluxos de trabalho com mais rapidez e precisão, uma vez que os padrões estarão à disposição do desenvolvedor.[7]

Estudos feitos durante o Plop2015 [4] identificaram os maiores desafios da comunidade de padrões com repositórios de padrões. Primeiro, é difícil procurar padrões usando os mecanismos de busca da internet porque eles não podem facilmente distinguir os padrões simplesmente usando a pesquisa de palavras-chave. Segundo, os recursos de padrões de projeto são muito fragmentados, complicando a busca pelos mesmos. Padrões podem ser publicados em vários processos, revistas e livros, arquivados em diferentes repositórios de padrão ou hospedados em inúmeros sites. Finalmente, muitos repositórios de padrões de projeto não são mais mantidos porque o financiamento acabou, seus desenvolvedores priorizaram outros projetos, ou não havia pouco ou nenhum contribuição da comunidade de padrões. Apesar de

existirem estudos sobre o uso de padrões para a criação de ChatBots, atualmente não foram encontrados estudos sobre o uso de ChatBots, baseados em sistemas de recomendações e sendo utilizados como repositório de padrões, como é proposto neste artigo.

3 MODELO PROPOSTO

O modelo estrutural proposto para conceber o C2-P2 se baseia na divisão de suas principais funções em quatro módulos interconectados. O principal objetivo do C2-P2 é auxiliar desenvolvedores a buscar padrões de projeto que se encaixam às suas necessidades, incluindo os detalhes sobre os padrões e seus exemplos de uso. O C2-P2 faz o uso das características interativas providas pelos chatbots para melhorar a "conversa" entre o desenvolvedor e os padrões. Os quatro módulos que fazem parte do C2-P2 são (ver Figura 1):

Módulo Introdução: constitui-se como a porta de entrada para a conversa do desenvolvedor com o Chatbot. Sua finalidade é dar as boas vindas aos desenvolvedores e contextualiza-los sobre os recursos disponíveis no C2-P2 e como esses recursos pode auxiliá-los a encontrar o padrão de projeto ideal para o seu projeto. Ao final da interação de introdução ao C2-P2, a conversa é redirecionada para o **módulo Listagem** ou **módulo de Fluxo**, dependendo das opções escolhidas pelo usuário.

Módulo Fluxo: é o módulo responsável por auxiliar o desenvolvedor a encontrar o padrão de projeto que se encaixa as suas necessidades. Este módulo pode ser considerado o mais importante de todo o C2-P2. Seguindo a estrutura de uma árvore de decisão, são apresentadas uma série de perguntas sobre o projeto atual do desenvolvedor, cada uma com duas opções de resposta. No final de um fluxo é então apresentado o padrão que se melhor encaixa as necessidades do desenvolvedor de acordo com as respostas dadas pelo desenvolvedor ao C2-P2.

Módulo Listagem: é um módulo que funciona como alternativa para o **módulo de fluxo**. Caso o desenvolvedor já possua um conhecimento prévio sobre um padrão de projeto e deseje aprender mais sobre ele, a listagem permite que o desenvolvedor acesse o padrão de projeto diretamente sem ter que passar por uma série de perguntas. A listagem fica disponível na introdução em uma lista oferecida pelo C2-P2. A partir da escolha de um padrão de projeto da listagem o desenvolvedor e será redirecionado imediatamente para o **módulo de detalhes** do padrão selecionado.

Módulo Detalhes: este módulo representa a última fase da conversa do desenvolvedor com o C2-P2. Ele recebe como entrada o padrão de projeto que foi escolhido no **módulo de listagem**, ou designado no **módulo de fluxo**. O **módulo de detalhes** é responsável por apresentar todas as características e definições sobre um padrão de projeto. O desenvolvedor poderá então selecionar os tópicos que quiser obter mais detalhes até decidir finalizar a conversa com o C2-P2.

Em uma futura versão do C2-P2, também serão implementadas funcionalidades extras que pretendem melhorar ainda mais a iteratividade entre o desenvolvedor e o Chatbot. Primeiramente, uma função de *feedback*, que permitirá que desenvolvedores expressem suas experiências e opiniões em relação a um padrão dentro da própria conversa com o Chatbot. Seus comentários poderão ser visualizados como uma opção extra na listagem de características e definições presente dentro do **módulo de detalhes**. Dentro deste mesmo módulo, ao selecionar a opção de visualização de padrões de projeto relacionados, o usuário também terá a oportunidade de acessar imediatamente os detalhes dos Patterns que possuem alguma relação com o Pattern atual. Isso irá retirar a necessidade de uma nova navegação pelos **módulos de fluxo ou listagem**, e trará dinamismo à conversa do usuário com o C2-P2.

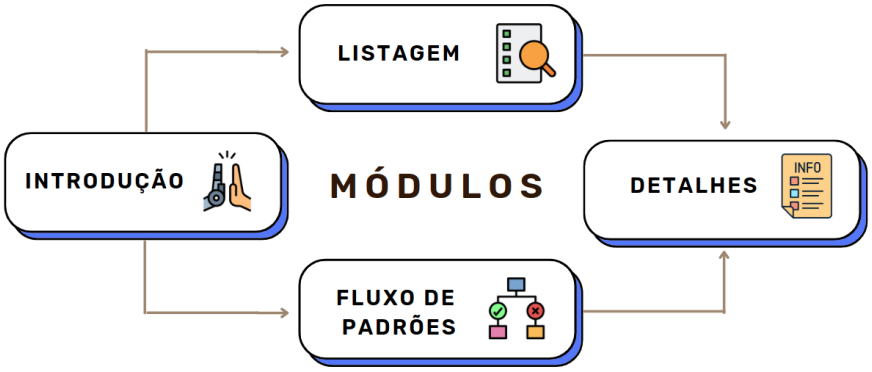

Fig. 1. Fluxograma da relação entre os módulos do C2-P2

O modelo do C2-P2 que está sendo proposto tem como vantagem de estimular uma conversa interativa junto aos desenvolvedores. E também retira a necessidade por parte do desenvolvedor dele ter que ter um conhecimento prévio sobre os detalhes do padrão de projeto. A árvore de decisão implementada no modelo permite que o desenvolvedor seja direcionado apenas aos patterns que estejam alinhados às suas necessidades, evitando que o desenvolvedor tenha que realizar uma navegação exploratória até encontrar o pattern que deseja. Desta forma, o C2-P2 faz a recomendação dos padrões de uma maneira mais orgânica e interativa.

Quando comparando o modelo proposto com outras alternativas exploradas pela comunidade, é possível observar que a abordagem de uso de Chatbots possui vantagens e desvantagens. Uma destas alternativas envolve criação de um repositório baseado em *wiki* com a necessidade de manutenção contínua pela própria comunidade [4].

Uma abordagem baseada em Chatbots possui desvantagem em relação a baseada em *wiki* no que se diz respeito ao volume de informações que podem ser adicionadas a um repositório em um determinado período de tempo. Pois, com o uso de Chatbots, apenas o desenvolvedor é capaz de inserir novos padrões de projeto. Enquanto com o uso de uma *wiki*, qualquer usuário verificado pode contribuir para o repositório.

Por outro lado, os Chatbots permitem a criação de repositórios que auxiliam o usuário a encontrar os padrões de projeto que melhor atendem as suas necessidades. Retirando a exigência de um conhecimento prévio por parte do usuário sobre os padrões de projeto disponíveis.

4 IMPLEMENTAÇÃO DA FERRAMENTA

O objetivo desta seção é apresentar uma implementação do modelo proposto para um conjunto de padrões específicos. Para a implementação da abordagem C2-P2 foi selecionado os padrões propostos por Anathan Telles Pereira and Zaina. [1] e a ferramenta Botpress[1] conforme descrito nas próximas seções.

4.1 Padrões Utilizados

Os padrões de projeto utilizados na implementação da abordagem C2-P2 foram propostos por [1] no artigo entitulado *Towards a Pattern Language to Embed UX Information in Agile Software Requirements*. O conjunto de padrões propostos no artigo tem o propósito de auxiliar times de desenvolvimento a organizar as informações de UX (User eXperience -

[1]https://botpress.com/

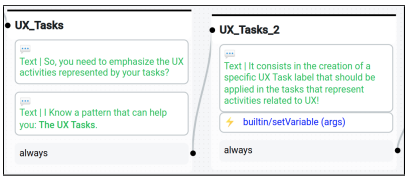

Fig. 2. Exemplo de um *Node* na ferramenta

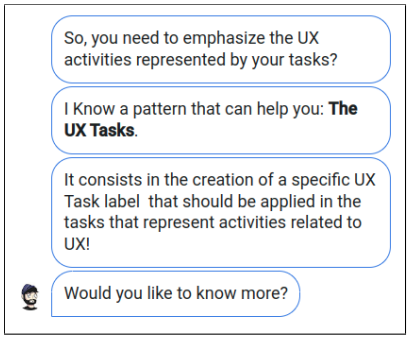

Fig. 3. Saída do *Node* em uma conversa

Experiência do Usuário) a partir de user stories.

Os autores propõem 5 padrões de projeto, sendo que todos eles estão sendo apresentado na implementação da ferramenta seguindo o modelo do C2-P2. Embora esta implementação contenha cinco padrões, o modelo proposto pelo C2-P2 permite que outros padrões sejam adicionados de maneira flexível.

4.2 Desenvolvimento

O desenvolvimento da ferramenta foi realizado utilizando a ferramenta *Botpress* que provê uma interface visual para criação do chatbot que abstrai os detalhes de codificação em determinado nível. Em uma visão mais abrangente, pode-se considerar que a implementação do chatbot é realizada como um processo de um diagrama de fluxo que recebe o estado atual da conversa (por exemplo, apresentação, explicação de um Pattern) junto com a entrada do usuário, e direciona a conversa para um novo estado. O elemento fundamental que compõe os elementos do diagrama que é montado na ferramenta, é chamado de bloco de execução (ou *Node*). Ele possui o propósito de determinar as ações que o chatbot irá realizar durante a conversa com o usuário. Estas ações podem representar desde o envio de uma mensagem de texto até uma execução de código mais complexa. Ao fim de um destes bloco de execução, a conversa é direcionada para um novo bloco para então repetir o processo.

Um exemplo do funcionamento destes blocos pode ser encontrado na Figura 2-A. Nele, dois blocos se conectam com o propósito de introduzir o desenvolvedor ao padrão de projeto *UX Tasks* e perguntar se ele deseja saber mais sobre sobre esse Pattern. A Figura 3-B apresenta como esses blocos se traduzem em uma conversa real com o chatbot.

Há também uma variação do *Node* denominada de *Choice*, o qual também é usada extensivamente durante nossa implementação. O objetivo deste elemento é simplificar as interações de perguntas/respostas que ocorrem durante uma conversa do chatbot com o desenvolvedor. Isto é feito através do uso de respostas previamente definidas o qual o usuário é capaz de interagir pelo meio de botões. Cada resposta é capaz de gerar uma ramificação do fluxo da conversa, o que permite que o usuário tenha uma interação única de acordo com a sua necessidade.

Por último, um esquema de armazenamento de variáveis também é usado para auxiliar o desenvolvimento de funções mais complexas do chatbot. As variáveis possuem a característica de persistencia, ou seja, elas não são excluídas pelo Chatbot após o fim de uma sessão de interação entre o usuário e o chatbot. Essas variáveis podem ser acessadas futuramente sempre que necessário. Este recurso é útil para que o chatbot se "lembre" de ações passadas do usuário, o que gera possibilidades para criar uma experiência personalizada durante a conversa com o chatbot. Ao utilizar os elementos explicados anteriormente de forma conjunta, é possível, então, criar estruturas mais complexas com diversas

ramificações. Desta maneira, essas estruturas complexas são capazes de simular uma conversa como a ocorrida no mundo real.

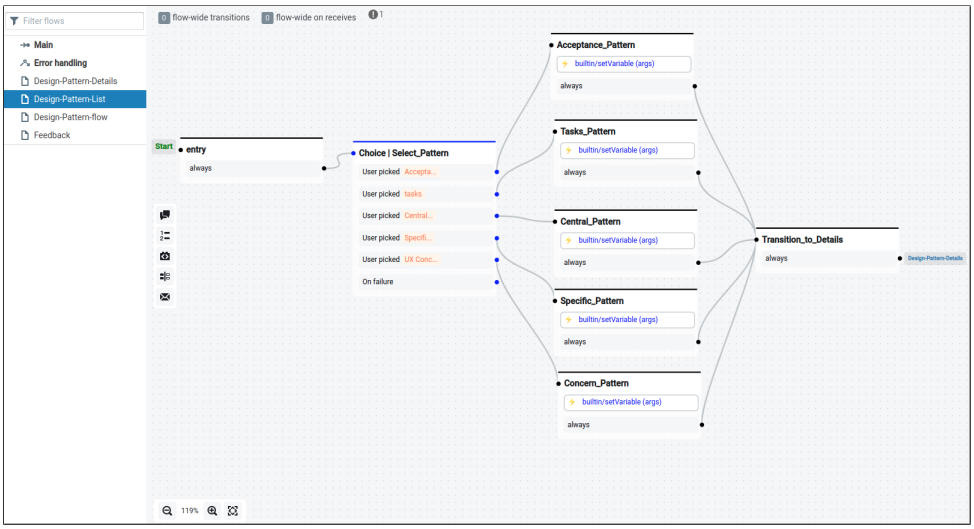

Fig. 4. Visualização de do subflow *List* dentro da plataforma Botpress

A Figura 4 ilustra a implementação do **módulo de listagem** dentro da ferramenta do botpress. Ela contempla os 3 elementos fundamentais (*Nodes*, *Choices* e Armazenamento de variáveis) aplicados na prática.

De maneira análoga, todos os módulos iniciam sua execução em um *Node* de entrada e finalizam em um *Node* de saída, no qual ele poderá ser direcionado ou para um novo módulo ou, então, finalizar a sessão. No caso em especifico do **módulo de listagem**, o *Node* entrada sempre será acessado através **módulo de introdução**, e a saída sempre irá fazer a transição para o **módulo de detalhes**.

O bloco de execução do tipo Choice é o proxímo passo logo após o *Node* de entrada. Ele é usado com o propósito de permitir que o usuário selecione manualmente qual padrão de projeto ele deseja visualizar. Em seguida, ele é direciona o fluxo para um dos 5 *Nodes* que podem ser visualizados na figura 4, sendo cada um relativo a uma das escolhas possíveis.

Esses *Nodes* utilizam a função de armazenamento de variáveis para registrar qual padrão de projeto foi selecionado pelo usuário. Isto será útil para que o próximo fluxo, o de detalhes, saiba qual é o padrão de projeto que ele deve apresentar informações. Uma vez que a informação é armazenada, o *Node* de saída direciona o fluxo da conversa para o **módulo de detalhes**.

Além do **módulo de listagem**, cada módulo também possui o seu próprio conjunto de especificidades na sua organização para que atuem de maneira correta e cumpram o seu propósito. A seguir estão detalhadas a implementação dos outros três módulos que compõem o bot C2-P2:

Main: o seu fluxo possui dois caminhos distintos que são acessados sob a condição de ser a primeira visita do usuário ou não. Caso seja, o usuário deve informar seu nome para que fique armazenado no esquema de variáveis do Bot. Independente de qual caminho o usuário percorrer, será apresentada uma escolha, representada por um *Node* do tipo *Choice*, que direcionará o usuário para o subflow *Design-Pattern-List* ou *Design-Pattern-Flow*

Design-Pattern-Flow: os blocos de execução do tipo *Choice* são responsáveis pela formação da árvore de decisão que direciona o usuário para o padrão de projeto que é adequado a suas necessidades. A árvore de decisão implementada para o C2-P2 pode ser encontrada na Figura 5. Nela é possível visualizar como os *Nodes* do tipo *Choice* ramificam o fluxo e formam os "galhos" da árvore. É também possível visualizar como esses *Nodes* sempre direcionam o usuário para um *Node* "folha" que representam o padrão de projeto que se adéqua as necessidades do usuário. Esse fluxo sempre seguirá um caminho unilateral, e não permite o percurso entre *Nodes* que não possuem um caminho entre eles explicitamente implementado no código.

Como nesta implementação considera-se apenas 5 padrões de projeto, apenas 4 perguntas foram suficientes para formar a estrutura e abranger todos os Patterns disponíveis.

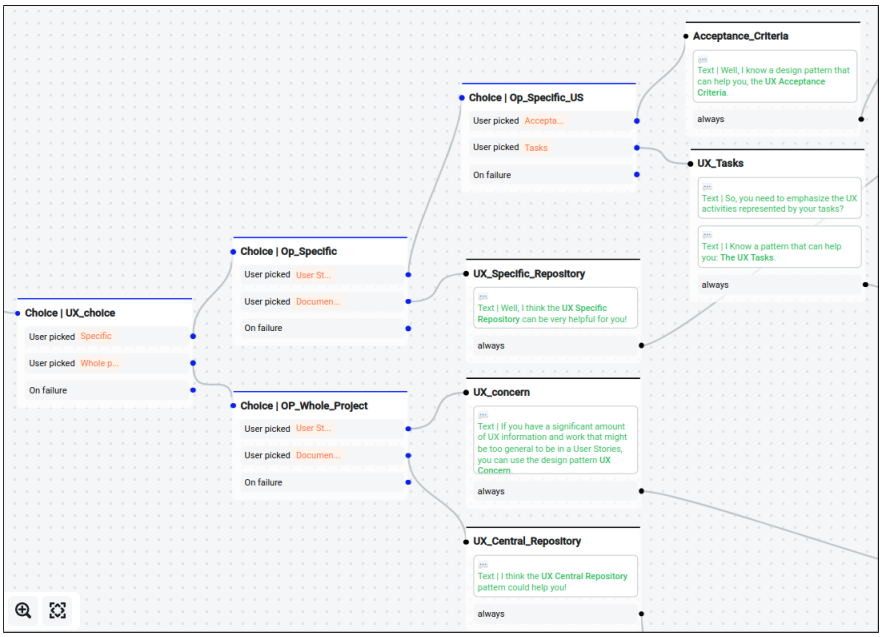

Fig. 5. Visualização da implementação de uma árvore de decisão dentro da plataforma Botpress

Design-Pattern-Details: esse sub-fluxo recebe como entrada a variável que foi preenchida pelo subflow *Design-Pattern-List* ou *Design-Pattern-Flow* e exibe uma lista. Esta lista apresenta as seguintes categorias relacionada ao pattern escolhido: Contexto, Aplicação, consequências de sua aplicação, padrões de projeto similares e Exemplo de uso. Após escolher uma das categorias, é exibido então a informação selecionada sobre o pattern.

O sub-fluxo também possui um loop, que permite que o usuário escolha quantas categorias desejar até que a finalização da conversa, ou seja, a saída do loop, seja especificada pelo usuário. Consequentemente, a sessão com o chatbot será encerrada.

Considerando que a implementação de demonstração possui 5 padrões de projeto (com a possibilidade de adicionar mais com passar do tempo), sendo que cada um destes possuem 5 categorias diferentes de exibição, temos no total 25 categorias que devem ser desenvolvidas para o produto final. Para este caso, o uso de uma interface gráfica pode ser complicado no ponto de vista de um desenvolvedor, justamente por conta da grande quantidade de informações que precisam ser exibidas ao mesmo tempo.

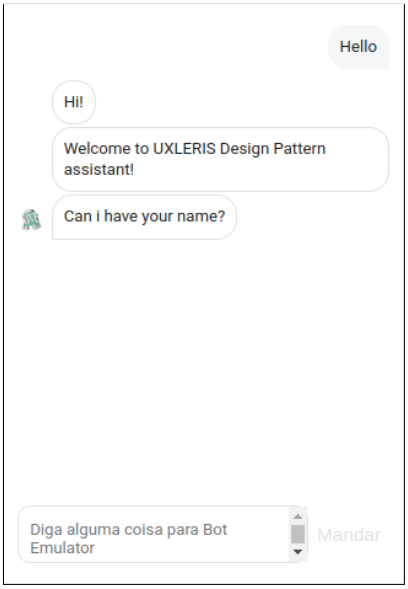

Fig. 6

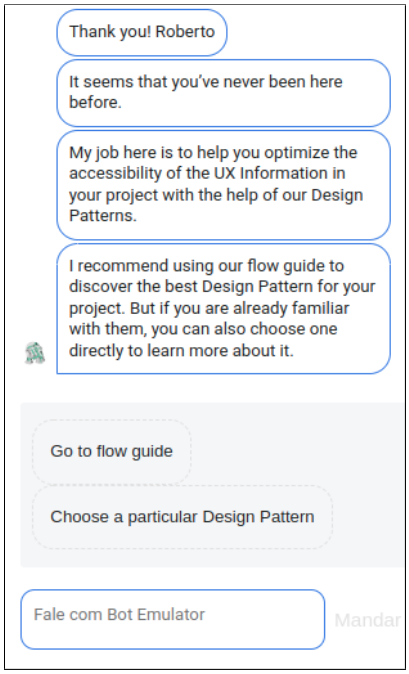

Fig. 7

Um código auxiliar escrito em *JavaScript* foi feito para contornar esse problema. Ele recebe uma chave que combina o padrão de projeto com a categoria desejada, e retorna para o Bot a informação desejada. Desta maneira, os dados ficam armazenados em uma lista localizada em um código separado da ferramenta, o que facilita a manipulação dos dados pelo desenvolvedor.

4.3 Exemplo de uso

Nesta seção será apresentada um exemplo de uso da ferramenta considerando um usuário fictício. Roberto trabalha para uma empresa de investimentos e está atuando como Scrum Master para um de seus projetos. No início do planejamento de uma das Sprints foi definida a seguinte User Story: "Como usuário, eu quero ver todos os meus investimentos em uma apenas tela". Junto com ela, existem especificações de UX que são essenciais para a sua realização. Roberto gostaria de encontrar alguma maneira de enfatizar essas especificações para facilitar o trabalho dos desenvolvedores e garantir que elas não sejam esquecidas.

A sessão de conversa com o chatbot começa no momento em que o usuário envia uma mensagem pela caixa de texto 6. Como é a primeira interação de Roberto, ele irá informar o seu nome para que fique armazenado chatbot e possa ser usado em futuras conversas (interações) 6.

Após as introduções feitas pelo chatbot, são oferecidas as opções de visitar o Fluxo de padrões ou a Listagem. Roberto não conhece em detalhes os Patterns oferecidos pelo chatbot, pórem ele possui um conhecimento do projeto que está atuando. Logo a opção do Fluxo de padrões é a melhor escolha 7.

Para ilustrar a conversa entre Roberto e o chatbot será apresentada uma série de perguntas sobre o seu projeto para que o seu Design Pattern adequado possa ser encontrado 8. A primeira pergunta é sobre a natureza da informação

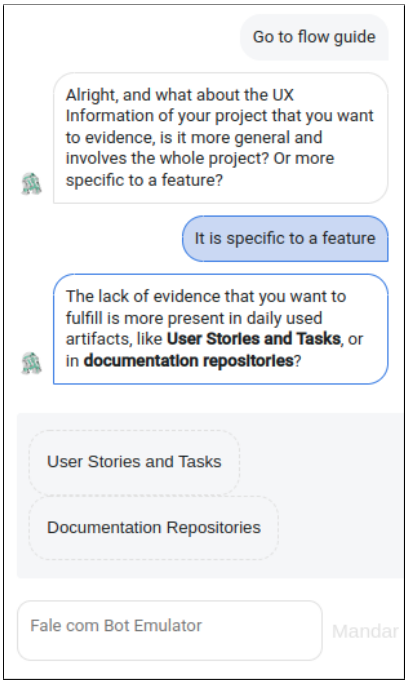

Fig. 8

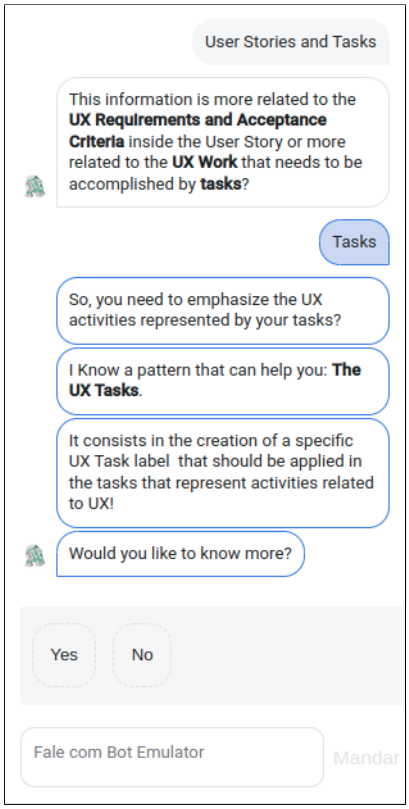

Fig. 9

que ele deseja evidenciar, Roberto informa que é específico para uma feature 8. Em seguida, indica que seu problema envolve User Stories e Tasks, mais especificamente o trabalho de UX que precisa ser realizado por uma Task. Assim finalizando o fluxo de decisão 9.

O Design Pattern selecionado para Roberto é o UX Tasks. É apresentado um breve resumo sobre no que se baseia este Design Pattern. O chatbot pergunta então se Roberto deseja saber mais sobre o pattern, e ele indica que sim 9. O chatbot então oferece um lista com todas as informações que podem ser relevantes para ele, desta vez não no formato de botões, mas com números que representam cada opção possível 10.

Roberto resolve selecionar a opção sobre Exemplo (opção 5), o chatbot disponibiliza um exemplo de uso do Design Pattern UX Tasks, o que poderá ser usado de inspiração para aplicação no projeto de Roberto 11 . Como Roberto deseja saber tudo sobre o pattern UX Tasks, ele interage com o chatbot para verificar todas as opções disponíveis. Ao finalizar sua conversa com o chatbot, Roberto passa a ser capaz de aplicar o pattern que o chatbot indicou a ele. Agora Roberto, tem um conhecimento sobre como funciona a conversa com o chatbot e poderá visitá-lo outras vezes no futuro caso suas demandas sejam outras.

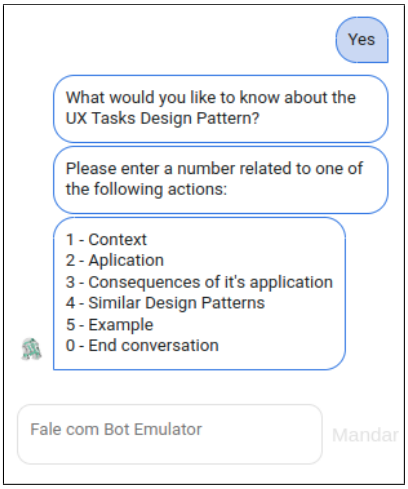

Fig. 10

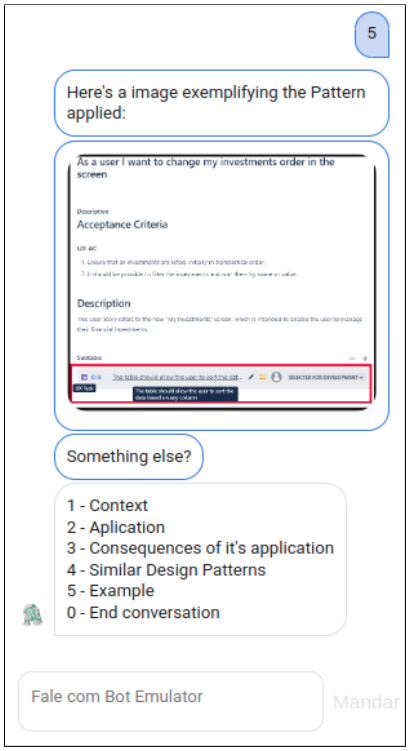

Fig. 11

4.4 Limitações

A proposta C2-P2 possui algumas limitações quanto a sua escalabilidade em termos de quantidade de informações e também sobre funcionalidades que permitem uma maior interatividade entre o Chatbot e o desenvolvedor. ———-

O caminho para se chegar até os padrões está ligada diretamente com as informações descritas de forma explícita no próprio código do chatbot. Isto pode ser considerada uma boa alternativa para ferramentas que possuem uma quantidade pequena ou média de patterns. Contudo, para um volume maior de padrões é recomendado buscar desenvolver soluções mais genéricas e de fácil expansão que permitam a escalabilidade do conjunto de padrões. Um detalhe importante é que o problema de escalabilidade não está relacionado necessariamente com o poder de processamento do Chatbot, e sim com a dificuldade que um desenvolvedor pode enfrentar para gerir o caminho para cada padrão dentro da árvore de decisão em uma interface gráfica. Uma sugestão seria o armazenamento externo das informações relativas a árvore de decisão em uma base de dados que pudesse ser chamada a qualquer momento pelo Chatbot (Com uma implementação semelhante a do **módulo de detalhes**). Desta maneira o desenvolvedor não se limitaria apenas as funcionalidades oferecidas pela sua ferramenta de escolha.

A Seção 4 apresentou apenas uma implementação da abordagem C2-P2, contudo não foi realizada uma avaliação com desenvolvedores reais para obter feedback sobre os recursos. Além disso, ainda existem funcionalidades que ainda estão para serem desenvolvidas, como uma opção do usuário avaliar por forma de texto o padrão de projeto que lhe foi atribuído. Assim como ler avaliações de outros usuários.

5 CONCLUSÃO

Neste artigo foi apresentado o C2-P2, um ChatBot capaz de conversar com o usuário, entender suas necessidades e recomendar o uso de um padrão de projeto que possa ajudá-lo. O Chatbot coleta várias informações durante a conversa, que são armazenadas em variáveis para que possam ser utilizadas em uma árvore de decisão para encontrar o padrão ideal. Este trabalho se diferencia de outros trabalhos que abordam os tópicos de ChatBots juntamente com linguagens de padrões, explorando o uso de um ChatBot como um sistema de repositório e consulta de padrões, que é uma prática não muito explorada até então.

O C2-P2 possui claras limitações apresentadas na seção 4.3, principalmente quanto a interatividade entre o ChatBot e o usuário. Em trabalhos futuros será trabalhado uma maneira de diminuir essa limitação de iteratividade permitindo que o usuário permitindo que o mesmo dê informações adicionais como exemplos de casos de uso de padrões. Além disso, será feito um trabalho para que o sistema reconheça uma maior quantidade de padrões de projeto.

REFERENCES

[1] Eduardo Guerra Anathan Telles Pereira, Abner Cleto Filho and Luciana A.M. Zaina. 2021. Towards a Pattern Language to Embed UX Information in Agile Software Requirements. *European Conference on Pattern Languages of Programs* (2021). https://doi.org/10.1145/1122445.1122456

[2] Mario Casillo, Fabio Clarizia, Giuseppe D'Aniello, Massimo De Santo, Marco Lombardi, and Domenico Santaniello. 2020. CHAT-Bot: A cultural heritage aware teller-bot for supporting touristic experiences. *Pattern Recognition Letters* 131 (2020), 234–243. https://doi.org/10.1016/j.patrec.2020.01.003

[3] Ahmed Fadhil. 2018. Domain Specific Design Patterns: Designing For Conversational User Interfaces. https://doi.org/10.48550/ARXIV.1802.09055

[4] Paul Salvador Inventado and Peter Scupelli. 2017. Towards a Community-Centric Pattern Repository. In *Proceedings of the 22nd European Conference on Pattern Languages of Programs* (Irsee, Germany) *(EuroPLoP '17)*. Association for Computing Machinery, New York, NY, USA, Article 36, 4 pages. https://doi.org/10.1145/3147704.3147743

[5] J V Kulikova, Yu A Orlova, V L Rozaliev, A V Zaboleeva-Zotova, and R L Lelikov. 2021. Development of a method for determining the state of a person using a chat bot. *Journal of Physics: Conference Series* 1801, 1 (feb 2021), 012008. https://doi.org/10.1088/1742-6596/1801/1/012008

[6] Milla T Mutiwokuziva, Melody W Chanda, Prudence Kadebu, Addlight Mukwazvure, and Tatenda T Gotora. 2017. A neural-network based chat bot. In *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*. 212–217. https://doi.org/10.1109/CESYS.2017.8321268

[7] John Ndeta, Stamatia Katriou, and Kerstin Siakas. 2015. An Approach To E-Workflow Systems With The Use Of Patterns. *International Journal of Entrepreneurial Knowledge* 3 (06 2015). https://doi.org/10.1515/ijek-2015-0007

[8] Arsénio Reis, Dennis Paulino, Hugo Paredes, Isabel Barroso, Maria João Monteiro, Vitor Rodrigues, and João Barroso. 2018. Using intelligent personal assistants to assist the elderlies An evaluation of Amazon Alexa, Google Assistant, Microsoft Cortana, and Apple Siri. In *2018 2nd International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW)*. 1–5. https://doi.org/10.1109/TISHW.2018.8559503