SECO-RCR: A Tool to Manage Requirements Change in Software Ecosystems

Eduardo dos Santos Gonçalves
edusantos@edu.unirio.br
UNIRIO
Rio de Janeiro, Brazil

Paulo Malcher
malcher@edu.unirio.br
UNIRIO & UFRA
Rio de Janeiro, Brazil

Laura O. Moraes
laura@uniriotec.br
UNIRIO
Rio de Janeiro, Brazil

Davi Viana
davi.viana@ufma.br
UFMA
São Luís, Brazil

Rodrigo Pereira dos Santos
rps@uniriotec.br
UNIRIO
Rio de Janeiro, Brazil

ABSTRACT

Software ecosystems (SECO) are becoming a predominant mode of modern software development and can be defined from different perspectives. From a project perspective, SECO are groups of projects that are developed and co-evolved in the same environment. SECO introduced complexity into requirements management, which is a key process of requirements engineering (RE) and an effective way to ensure software quality. There is difficulty in managing requirements change in open and dynamic environments such as SECO due to the existence of multiple stakeholders that can form distinct crowds in the ecosystem. In SECO, a crowd is a large and heterogeneous user base that provides requirements through multiple communication channels. Hence, meeting the expectations of distinct crowds in SECO requires new RE approaches. This paper presents a tool called Software Ecosystems - Requirements Change Request (SECO-RCR) that applies software repository mining, natural language processing techniques, and machine learning to support requirements change management in SECO. SECO-RCR allows requirements managers to define different RCR and consult different crowds to assist them in their decision making.

Ferramenta: https://zenodo.org/records/11432156
Apresentação (Youtube): https://youtu.be/2OgCRvrA1Ec
Apresentação (Zenodo): https://zenodo.org/records/11429519

KEYWORDS

Requirements Change, Requirements Management, Software Ecosystems, GitHub

1 INTRODUÇÃO

O conceito de ecossistemas de software (ECOS) impactou o mundo dos negócios e da pesquisa de plataformas, afetando significativamente a sociedade e a indústria de software [11]. Empresas têm formado ECOS para obter vantagem competitiva, desenvolvendo serviços em conjunto para os clientes [24]. Para Damian et al. [7], ECOS estão se tornando um modo predominante no desenvolvimento de software. Ghimire et al. [10] afirmam que eles são partes integrante do desenvolvimento de software moderno, com diversos atores colaborando para desenvolver, manter e evoluir um sistema de software complexo.

ECOS podem ser definidos como grupos de projetos que são desenvolvidos e coevoluem no mesmo ambiente [18]. ECOS compreendem diversos componentes de software, plataformas e múltiplos atores que interagem entre si [11]. Damian et al. [7] citam que existem três conjuntos diferentes, mas às vezes sobrepostos, de *stakeholders* em ECOS: os usuários dos produtos da organização central, os desenvolvedores externos e os usuários finais. Esses múltiplos *stakeholders* em um ECOS formam multidões distintas [12]. Para Johnson et al. [12], uma multidão em ECOS é uma grande base de usuários heterogênea fornecendo requisitos em vários canais de comunicação com múltiplos *stakeholders*.

ECOS se tornaram um campo de pesquisa ativo em engenharia de software (ES) [6, 21] e introduzem complexidade na gerência de requisitos [3, 7, 11, 14, 16]. Axelsson e Skoglund [3] afirmam que o conjunto dinâmico de atores nos ECOS introduz desafios para a compreensão, gerência e manutenção do equilíbrio entre os diversos requisitos dos *stakeholders*. A complexidade e natureza mutável de ECOS resultam em vários novos requisitos baseados nas tendências do ecossistema que dificultam a gerência de requisitos [14]. Segundo Damian et al. [7], a gerência de requisitos em ECOS traz desafios às parcerias entre a organização central e os desenvolvedores externos.

A gerência de mudanças de requisitos é uma atividade importante para a gerência de requisitos, mas permanece pouco explorada em ECOS [27]. Mudanças introduzem conflitos em ECOS e podem ser inesperadas e perturbadoras [5]. Gerenciar mudanças de requisitos em ambientes abertos e dinâmicos que vão além dos limites de uma única organização, como ECOS, é desafiador [7, 16]. Bogart et al. [5] afirmam que as abordagens tradicionais de controle centralizado de mudanças ou de gerência de mudanças (e.g., painéis de controle de mudanças e *roadmapping*) rompem com a natureza dinâmica e distribuída dos ECOS. Além disso, as ferramentas para análise do impacto da mudança enfrentam desafios devido à escala e abertura dos ECOS. Portanto, atender às expectativas de múltiplos usuários em ECOS requer novas abordagens que envolvam essa multidão de *stakeholders* nas atividades da engenharia de requisitos (ER) [1].

Com o objetivo de auxiliar a gerência de mudanças em ECOS, este artigo apresenta a ferramenta Software Ecosystems - Requirements Change Request (SECO-RCR). SECO-RCR utiliza mineração de repositórios, técnicas de processamento de linguagem natural e

aprendizado de máquina para apoiar gerência de mudanças de requisitos em ECOS. A ferramenta permite que o gestor de requisitos defina requisições de mudança de requisitos (do inglês, *requirements change requests* - RCR) e consulte as multidões distintas de um ECOS para auxiliá-lo nas tomadas de decisão.

Este artigo está estruturado em seis seções. Após a introdução, a Seção 2 detalha como ocorrem as mudanças de requisitos em ECOS. A Seção 3 apresenta a ferramenta, incluindo a sua arquitetura, componentes, funcionamento e tecnologias utilizadas. A Seção 4 aborda alguns trabalhos relacionados. Na Seção 5, é descrito um exemplo de uso realizado com a ferramenta. Por fim, a Seção 6 conclui o artigo com algumas considerações finais e aponta trabalhos futuros.

2 MUDANÇA DE REQUISITOS EM ECOS

ECOS surgiram como um paradigma para a compreensão da dinâmica e da heterogeneidade no desenvolvimento colaborativo de software [9]. De acordo com Bianco et al. [4], diversas empresas perceberam que a funcionalidade necessária para satisfazer as necessidades dos clientes era maior do que sua capacidade poderia suportar e, por isso, começaram a formar ECOS. Ghimire et al. [10] citam que empresas como HubSpot[1], Salesforce[2], Xero[3], Slack[4], Shopify[5] e Wix[6] prosperaram com sua integração, mercado, inovação e outras qualidades que tornam um ECOS próspero. Santos et al. [25] apresentaram alguns exemplos de ECOS como Eclipse Foundation[7] e Apache Foundation[8] (ECOS baseado em um conjunto de projetos mantidos por diferentes atores e comunidades), SAP[9] e Amazon[10] (ECOS baseado em código-fonte e outros artefatos protegidos por acordos de confidencialidade), e iOS[11] e Android[12] (ECOS baseado em contribuições proprietárias e de código aberto).

Em particular, existe uma lacuna significativa em relação à gerência de mudanças de requisitos em ECOS, que é a existência de múltiplos atores que contribuem com diferentes elementos para o sistema ou produto e podem ter objetivos diferentes ou até conflitantes [8]. Além disso, as mudanças em um produto podem causar efeitos cascata em muitos outros em ECOS [20]. Na prática, os profissionais enfrentam frequentemente desafios na gerência de mudanças de requisitos devido à abertura e dinamismo dos ECOS com a interdependência entre os múltiplos produtos e a integração entre eles e a plataforma tecnológica comum [14]. O "gestor de requisitos" é profissional que tem a responsabilidade de identificar, definir e priorizar as solicitação de mudanças de requisitos da multidão de *stakeholders* do ECOS. Dentro de muitas empresas de software, esta função pode ser chamada de "gerente de produto", "proprietário do produto" ou "gerente de projeto". No contexto de ECOS, também pode ser chamado de "gerente da plataforma".

Em ECOS, o destinatário ideal para uma determinada solicitação geralmente muda ao longo do tempo, sendo necessário repensar periodicamente a forma como os requisitos trafegam pelas organizações de software e quais papéis são responsáveis por buscar o alinhamento dos objetivos [14]. Além disso, como, quando e por quem as mudanças são realizadas em um ECOS estão sujeitos a negociação (muitas vezes implícita) entre vários atores do ecossistema. De acordo com Bogart et al. [5], o fardo da mudança pode ser suportado por diferentes atores do ECOS: uma organização central pode decidir como fazer uma mudança, pode investir esforços adicionais para facilitar a adoção da mudança ou pode decidir aceitar custos de oportunidade para não fazer nenhuma mudança. Os desenvolvedores externos podem monitorar regularmente as mudanças em suas dependências e tentar influenciar seu desenvolvimento ou podem retrabalhar seus próprios produtos. Os usuários finais podem encontrar defeitos se as alterações não forem feitas ou dificuldades de instalação se os produtos do repositório se tornarem incompatíveis. Portanto, a consulta a todos esses atores pode trazer benefícios para a gerência de mudanças de requisitos em ECOS.

3 VISÃO GERAL

SECO-RCR é uma ferramenta web gratuita e de código aberto sob a licença MIT[13]. A ferramenta visa auxiliar gestores de requisitos e desenvolvedores engajados em ECOS na gerência de mudanças de requisitos. SECO-RCR utiliza a mineração de dados em plataformas de interação entre a organização central, os desenvolvedores externos e os usuários.

3.1 Arquitetura da ferramenta

SECO-RCR foi dividida em módulos para maior manutenibilidade e separação de responsabilidades, visando aderir a conceitos-chaves da ES moderna, como coesão e acoplamento [26]. Para atender a este propósito, a ferramenta é arquitetada a partir de quatro módulos, conforme apresentado na Figura 1. O sistema gerenciador de banco de dados (SGBD) utilizado é o PostgreSQL[14].

3.1.1 Módulo Mineração. É responsável por fazer a coleta e tratamento inicial dos dados. Ele aloca as interações com ferramentas externas, como a mineração de dados em repositórios do GitHub e envio de e-mails. Atualmente, o módulo disponibiliza a mineração somente em repositórios abertos do GitHub, obtendo os dados de *issues* contidas na plataforma e, a partir da seleção do usuário sobre um filtro de palavras-chave ou não, realiza uma filtragem nos dados obtidos. Este módulo foi desenvolvido em Javascript utilizando o *framework* NodeJS.

3.1.2 Módulo Tópicos. É responsável por aplicar técnicas de aprendizado de máquina nos dados minerados. Ele realiza a modelagem de tópicos (utilizando Top2Vec [2]), visando o agrupamento das *issues* filtradas no módulo de *Mineração* em grupos de *issues* (tópicos) que possuam tendência a se relacionarem com um mesmo assunto. A premissa é que, se as *issues* utilizam as mesmas palavras com frequência, elas provavelmente estão se referenciando ao mesmo assunto. Outro propósito da geração de tópicos aplicada ao processo implementado na ferramenta é tornar a análise realizada pelo gestor

[1] https://hubspot.com/
[2] https://www.salesforce.com/
[3] https://www.xero.com/
[4] https://slack.com/
[5] https://www.shopify.com/
[6] https://www.wix.com/
[7] https://www.eclipse.org/org/foundation/
[8] https://www.apache.org/
[9] https://www.sap.com/
[10] https://www.amazon.com/
[11] https://www.apple.com/ios/
[12] https://www.android.com/
[13] https://choosealicense.com/licenses/mit/
[14] https://www.postgresql.org/

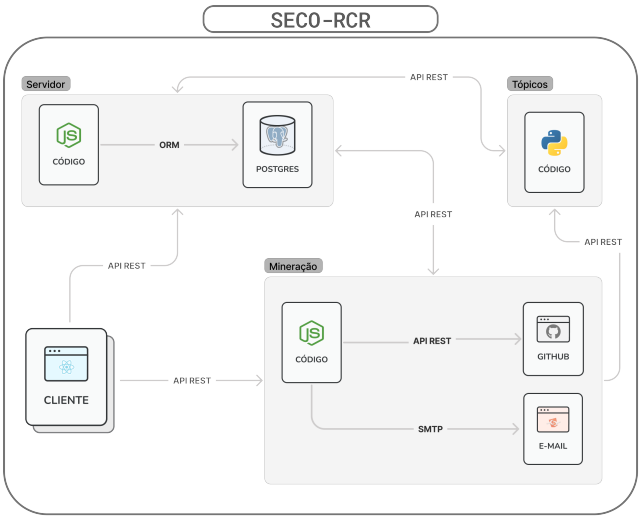

Figura 1: Arquitetura da ferramenta

de requisitos menos cansativa, com uma visão menos heterogeneizada das *issues* que possam tratar de um assunto relacionado. Após a geração dos tópicos, para cada tópico, é feita uma comparação de cada *issue* com as outras deste mesmo tópico para identificar quais *issues* possuem mais similaridade. Para gerar o valor comparativo de similaridade, os textos das *issues* são convertidos em vetores numéricos (utilizando Sentence Transformers [23]), que são comparados por meio do cálculo da similaridade de cosseno (utilizando Scikit-Learn [22]). A similaridade de cosseno utiliza o ângulo entre os vetores como métrica, ignorando sua magnitude. Dessa maneira, textos de tamanhos diferentes, mas que tratem do mesmo assunto, podem ser considerados similares. O objetivo é sinalizar, na posterior análise feita pelo gestor de requisitos, quais são as *issues* em destaque naquele tópico. Esta é uma abordagem de sumarizar o assunto do tópico sem recorrer a um segundo modelo para geração de resumo. Portanto, a *issue* que possuir maior similaridade com as demais tende a ser uma *issue* que representa bem uma possível mudança de requisito. Este módulo foi desenvolvido em Python.

3.1.3 Módulo Servidor. É responsável pelas interações diretas com o banco de dados e a autenticação dos usuários. Também possui rotinas automatizadas para o processamento das votações. Interage diretamente com os outros dois módulos que constituem a lógica do sistema (Mineração e Tópicos). O propósito deste módulo é receber as requisições relacionadas ao registro, atualização ou leitura dos dados da ferramenta, validá-las e executá-las. Este módulo foi desenvolvido em Javascript utilizando o *framework* NodeJS.

3.1.4 Módulo Cliente. É responsável pela interface a ser utilizada pelo usuário, além da interação com o banco de dados via Módulo Servidor e requisições de mineração via módulo Mineração. Aloca a interface web com a qual o usuário interage diretamente. Este módulo foi desenvolvido em Javascript utilizando o *framework* React.

3.2 Funcionamento da ferramenta

Uma visão sistemática do processo adotado na ferramenta é apresentada na Figura 2 por meio de um diagrama de atividades da UML. Os dois papéis de atores envolvidos no processo são o de gestor de requisitos do ECOS e o da multidão, que é uma grande base de usuários heterogênea fornecendo requisitos em vários canais de comunicação no ECOS [12]. Os detalhes para cada atividade mapeada na ferramenta são descritos a seguir:

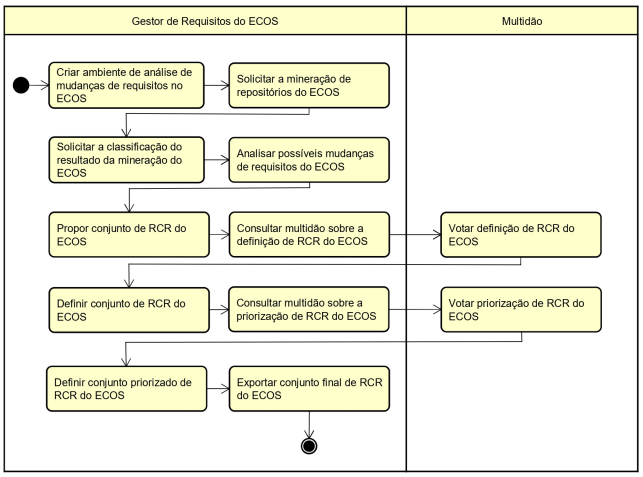

Figura 2: Diagrama de atividades da ferramenta

- **Criar ambiente de análise de mudanças de requisitos no ECOS**: O processo da ferramenta tem início nesta atividade, onde o gestor de requisitos autenticado no sistema deve solicitar a criação de um ambiente de análise. No contexto da ferramenta, um ambiente de análise é interpretado como um ECOS, i.e., um conjunto de projetos que coevoluem em um mesmo ambiente, seguindo a definição de Lungu et al. [18]. Para criar um ambiente, o gestor de requisitos deve definir um nome, uma descrição, o tipo de mineração (se é via repositórios de uma única organização do GitHub ou diferentes repositórios do GitHub) e se as *issues* mineradas serão filtradas de acordo com palavras-chave pré-definidas.
- **Solicitar a mineração de repositórios do ECOS**: Após a criação do ambiente, o módulo *Mineração* é acionado para obter as *issues* dos repositórios GitHub do ambiente. Após a obtenção das *issues*, é realizada a seguinte atividade de limpeza no corpo do texto de cada uma delas: (i) remoção de conteúdo não relevante (*links*, espaços demasiados, tags HTML, códigos, *stop-words*) e elementos textuais comuns em *issues* do GitHub (como [ ], [x], [x]); e (ii) conversão do texto para caixa baixa. Após a obtenção das *issues* de todos os repositórios do ambiente, se o gestor de requisitos optou por utilizar o filtro por palavras-chave, as *issues* que contiverem ao menos uma das palavras-chave em seu corpo ou em suas *labels* serão registradas no banco de dados (via módulo *Servidor*) para a próxima etapa. As demais *issues* serão descartadas. Para englobar mais *issues* que contivessem as palavras-chave, é aplicada a lematização (processo ao qual

as palavras são convertidas a sua forma básica, conhecida como lema [19]) nas palavras-chave e em cada *issue*;

- **Solicitar a classificação do resultado da mineração do ECOS**: Após a mineração dos repositórios, o gestor de requisitos é notificado por e-mail sobre o fim da mineração e orientado a entrar na ferramenta para solicitar o inicio da geração de tópicos. Em seguida, o módulo *Tópicos* é acionado para realizar a modelagem dos tópicos do ambiente determinado. Após a geração dos tópicos, é realizada uma comparação entre as *issues* contidas em cada tópico, visando sinalizar quais são as mais similares. Com estas informações processadas, o módulo envia os dados para o banco de dados (via módulo *Servidor*);
- **Analisar possíveis mudanças de requisitos do ECOS e Propor conjunto de RCR do ECOS**: Com o fim da geração dos tópicos, o gestor de requisitos é notificado por e-mail sobre o fim da geração de tópicos e orientado a entrar na ferramenta para iniciar a avaliação manual dos tópicos gerados e *issues* obtidas. Esta avaliação consiste em identificar se as *issues* contidas nos tópicos e as *issues* relacionadas a esta originam uma RCR, e definir e descrever qual a mudança de requisito identificada;
- **Consultar multidão sobre a definição de RCR do ECOS**: Após a análise do gestor de requisitos, se for identificada ao menos uma RCR, é possível iniciar o primeiro processo de consulta a multidão, que é relacionado a definição das RCR. O gestor de requisitos seleciona as RCR que consistirão a votação e define uma data de encerramento da votação. Após esta seleção, é gerado um *link* para a votação das RCR;
- **Votar definição de RCR do ECOS**: O membro da multidão de um ECOS (e.g., usuário final ou desenvolvedor externo), ao receber o *link* de votação para a definição das RCR enviado pelo gestor de requisitos da organização central por meio dos diferentes canais de comunicação do ECOS, indica sua concordância (ou discordância) em relação à RCR (se aprova, recusa ou não sabe). O votante também pode registrar comentários sobre as RCR (este comentário é obrigatório em caso de recusa da RCR). O votante necessita indicar o voto em pelo menos uma RCR para enviar seu voto. Para registrar o voto, o votante deve possuir um e-mail válido e um código de confirmação será enviado para este e-mail com o objetivo de confirmar o registro;
- **Definir conjunto de RCR do ECOS**: Na data de encerramento da votação de definição de RCR, a ferramenta processa os votos e envia um e-mail para o gestor de requisitos que criou o ambiente. Este e-mail informa sobre o encerramento da votação e orienta o acesso a ferramenta para a definição das RCR. O gestor de requisitos analisa o resultado da votação das RCR e seleciona quais são as RCR que constituirão o conjunto final para a priorização. A RCR pode ser modificada nesse processo e o histórico das modificações é registrado;
- **Consultar multidão sobre a priorização de RCR do ECOS**: Após definir o conjunto final de RCR, o gestor de requisitos pode iniciar o primeiro processo de consulta a multidão para a priorização das RCR. O gestor de requisitos define uma data de encerramento da votação. Em seguida, é gerado um *link* para a votação de priorização das RCR. Além disso, um e-mail com o *link* para a votação de priorização das RCR é enviado para todos que participaram da primeira votação;
- **Votar priorização de RCR do ECOS**: O votante, ao receber o *link* de votação de priorização para as RCR de determinado ambiente enviado pelo gestor de requisitos da organização central por meio dos diferentes canais de comunicação do ECOS ou pela própria ferramenta por e-mail, indica a prioridade sobre cada RCR por meio de níveis (quanto menor o valor de prioridade, mais prioritária). Para registrar o voto, o votante deve possuir um e-mail válido e um código de confirmação será enviado por e-mail para confirmar o registro;
- **Definir conjunto priorizado de RCR do ECOS**: Ao chegar a data de encerramento da votação de definição, a ferramenta processa os votos e envia um e-mail para o gestor de requisitos que criou o ambiente. Este e-mail informa sobre o encerramento da votação e orienta o acesso a ferramenta para a priorização final das RCR. A ferramenta apresenta a sugestão de priorização a partir de todos os votos registrados. Em seguida, o gestor de requisitos pode repriorizar as RCR se houver necessidade, e após, pode encerrar a análise;
- **Exportar conjunto final de RCR do ECOS**: Após o encerramento da análise, o gestor de requisitos possui o conjunto final de RCR e o ambiente é finalizado. É permitido ao gestor de requisitos exportar as RCR em arquivo no formato de valores separados por vírgula (.csv).

4 TRABALHOS RELACIONADOS

Em relação à gerência de mudanças de requisitos em ambientes colaborativos, dinâmicos e distribuídos, Lloyd et al. [17] propuseram uma ferramenta que apoia a gerência de mudanças de requisitos no desenvolvimento ágil distribuído. O principal objetivo da ferramenta é mitigar alguns dos desafios enfrentados pelo desenvolvimento ágil distribuído usando uma árvore de recursos. Por sua vez, Koh e Chua [15] propuseram uma ferramenta semiautomática para cobrir o processo de gerência de requisitos com tarefas automatizadas utilizando técnicas de aprendizado de máquina para apoiar a classificação de requisitos, identificação de ambiguidade de requisitos e priorização de requisitos.

Em um contexto mais próximo ao de ECOS, Knauss et al. [13] propuseram uma ferramenta de suporte para gerência de requisitos em desenvolvimento ágil de sistemas em larga escala baseada no sistema de controle de versão Git[15]. A ferramenta implementa uma abordagem para confiar em requisitos textuais baseados em um formato Markdown[16] e gerenciá-los no Git. Ela combina convenções úteis, *templates* e *scripts* auxiliares com soluções existentes do ecossistema Git, permitindo que equipes multifuncionais ágeis estejam cientes dos requisitos no nível do sistema e proponham atualizações eficientes para esses requisitos. No entanto, SECO-RCR surge para preencher uma lacuna relacionada à gerência de mudanças de requisitos realizada em ECOS, visto a complexidade deste processo em ECOS e a inclusão de distintas multidões. Até onde se sabe, não há uma ferramenta completa quanto a apresentada neste trabalho que inclua em todo o processo a mineração de *issues* e a

[15]https://git-scm.com/
[16]https://www.markdownguide.org/

inclusão de multidões distintas na tomada de decisões durante a gerência de mudanças de requisitos em ECOS.

5 EXEMPLO DE USO

Para descrever a utilização da ferramenta SECO-RCR, foi selecionado o ECOS do projeto Gerenciamento Livre de Parque de Informática (do francês, Gestionnaire Libre de Parc Informatique - GLPI) que está disponível no GitHub. O GLPI é um sistema de código aberto voltado à gestão de ativos e *helpdesk*. A organização deste projeto inclui o repositório do sistema GLPI, bibliotecas para facilitar a interação com a *application programming interface* (API) provida pelo sistema do GLPI, plugins, documentação e artefatos que consolidam o sistema GLPI. Um dos autores, que participa da comunidade de desenvolvedores, interpreta o papel do gestor de requisitos do ECOS neste exemplo. Após realizar o cadastro de usuário na ferramenta, foi solicitada a criação de um ambiente de análise, conforme Figura 3, com as seguintes características:

- **Nome:** ECOS GLPI;
- **Descrição:** Ambiente do ECOS GLPI;
- **Repositórios:** *glpi-project/android-inventory-agent*, *glpi-project/android-inventory-library*, *glpi-project/angularjs-glpi*, *glpi-project/glpi*, *glpi-project/glpi-agent*,*glpi-project/glpi-agentmonitor*, *glpi-project/glpi-inventory-plugin*, *glpi-project/java-library-glpi*, *glpi-project/javascript-library-glpi*, *glpi-project/node-module-glpi*, *glpi-project/php-library-glpi*, *glpi-project/plugins*, *glpi-project/swift-library-glpi*, *glpi-project/sysobject.ids*, *glpi-project/telemetry*;
- **Filtro de palavras-chave:** Sim;
- **Palavras-chave do ambiente:** linux, php, apache.

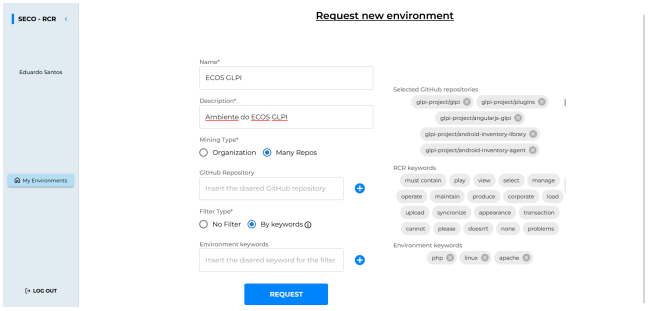

Figura 3: Cadastro de um ambiente no SECO-RCR

Vale ressaltar que, por escolha dos autores, o modo de criação deste ambiente de análise foi modificado para "vários repositórios", visto a remoção de repositórios de documentações. Esta funcionalidade auxilia o gestor de requisitos na análise de diferentes projetos (repositórios) em um ECOS. Ele pode escolher todos os repositórios de uma organização no GitHub ou apenas os que ele tiver interesse em analisar no ambiente a ser criado.

A análise de *issues* do GLPI ocorreu no dia 27 de maio de 2024, obtendo 310 *issues* e minerando 270 *issues* por meio da filtragem por palavras-chave. Então, foi solicitada a geração de tópicos no ambiente. Neste exemplo, a geração dos tópicos durou aproximadamente 2 minutos. Em seguida, a identificação de similaridade entre as issues de cada tópico durou entre 20 a 30 minutos. Foram obtidos dois tópicos para este ambiente. Por conta do curto tempo de análise, não foi possível fazer o experimento com alguma(s) comunidade(s) do GLPI. Portanto, para prosseguir a descrição dos próximos passos da ferramenta, o autor que exerceu o papel de gestor de requisitos do ECOS gerou os dados do exemplo. Inicialmente, foi possível identificar que em uma das *issues* do "*tópico 0*" houve similaridade com outras *issues* de outros repositórios do ecossistema, conforme pode ser visto na Figura 4.

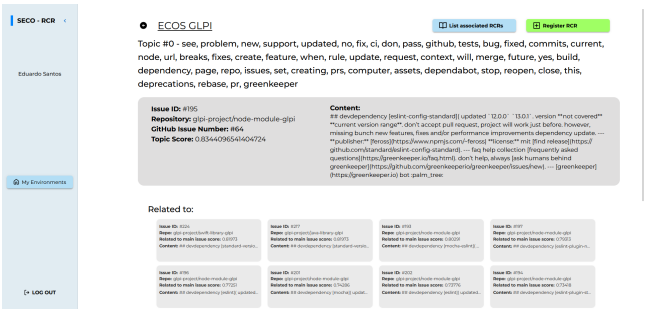

Figura 4: Detalhes de uma *issue* contida no tópico 0 do ambiente ECOS GLPI

O registro de RCR se origina da análise realizada pelo gestor de requisitos após a geração dos tópicos. Durante esta análise, o gestor identifica as RCR a partir das *issues* mineradas e as registra. O registro de uma RCR deve conter nome, descrição, uma *issue* principal relacionada e um conjunto de *issues* relacionadas (Figura 5). As nove RCR foram submetidas ao processo de votação de definição,

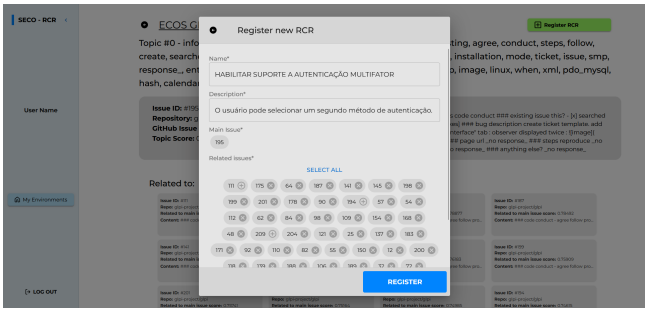

Figura 5: Registro de uma RCR a uma issue do tópico 0

conforme Figura 6. Cinco personas foram criadas para as etapas de votação deste ambiente, pois não foi possível obter colaboradores do ECOS indicado em tempo hábil para colaborar no exemplo descrito. A disponibilização de um *link* para a votação da multidão em relação à definição de RCR permite que o gestor de requisitos do ECOS compartilhe este *link* em diferentes canais de comunicação, o que melhora o alcance e a possível participação da multidão. O usuário votante (um participante da multidão) não precisa estar autenticado para votar. Porém, para registrar o voto, o usuário deve validá-lo por meio de um e-mail válido, a partir do qual receberá um código de acesso para validar o e-mail e registrar o voto.

Após o recebimento dos votos e seu processamento, o gestor de requisitos selecionou as cinco RCR mais relevantes, de acordo com

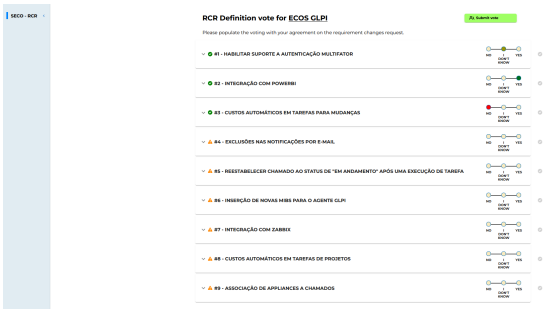

Figura 6: Votação de definição para o ambiente ECOS GLPI

a ordem calculada pela ferramenta, que foi definida a partir dos votos recebidos pela multidão. Em seguida, foi iniciada a votação de priorização das RCR (Figura 7). Um *link* para a votação da multidão em relação à priorização de RCR é gerado, permitindo que o gestor de requisitos do ECOS novamente compartilhe este *link* em diferentes canais de comunicação. Além disto, os usuários votantes que atuaram na votação de definição são notificados via e-mail sobre a abertura da votação de priorização e são motivados a participar.

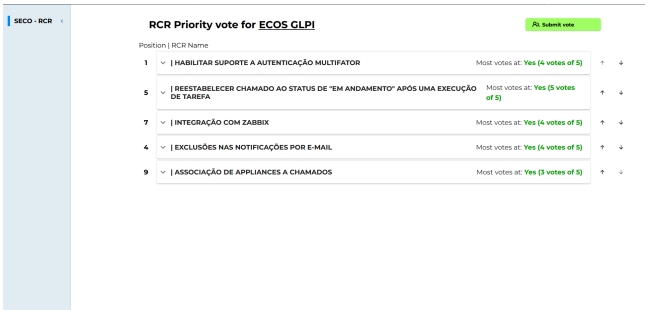

Figura 7: Votação de prioridade para o ambiente ECOS GLPI

As mesmas cinco personas votaram novamente na fase de priorização. Após esta fase, o gestor de requisitos optou por não modificar a priorização da multidão e encerrou a análise. Assim, foi gerado o conjunto final de RCR, como pode ser visto na Figura 8.

Figura 8: Conjunto final de RCR

6 CONCLUSÃO

Este artigo apresentou uma ferramenta que auxilia gestores de requisitos a gerenciar mudanças de requisitos em ECOS. A ferramenta apresenta potencial para o contexto de ECOS, pois inclui as distintas multidões no processo que gerencia requisitos que emergem de forma descentralizada por meio de diferentes canais de comunicação. Vale ressaltar que esta é a primeira versão da ferramenta, que ainda será avaliada por um grupo de especialistas. Esta versão também pode conter *bugs*.

Em relação aos artefatos de aprendizado de máquina, possíveis problemas de qualidade e representatividade do texto das *issues* são mitigados nos processos de mineração de dados, geração de tópicos e similaridade entre as *issues* de um tópico. Na mineração de dados, é realizada a limpeza no corpo do texto. Na geração de tópicos, são mantidas apenas palavras importantes em um contexto treinado com um conjunto maior de dados. Na similaridade entre as *issues* de um tópico, é utilizado um modelo de vetores numéricos pré-treinado por meio da biblioteca Sentence Transformers. Além disso, a modelagem de tópicos é realizada somente para cada conjunto de *issues* recuperadas, ou seja, a necessidade de atualização contínua do modelo é baixa. Portanto, o que necessita de atualização contínua é o modelo pré-treinado, mas em frequência menor (somente quando novos modelos pré-treinados mais robustos forem disponibilizados).

A ferramenta também apresenta potencial de expansão, podendo ser incluídas em versões futuras: (i) a integração com outras plataformas para extração de dados; (ii) a criação de ambientes de análise mais personalizáveis em relação aos estados das *issues* (abertas, fechadas ou ambas) e ao período de sua criação; (iii) a definição de novas estratégias para motivar a votação, gerenciá-la e analisar seus resultados; e (iv) a utilização de outros algoritmos e técnicas de aprendizagem de máquina para a filtragem de *issues*.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, CNPq (Proc. 316510/2023-8), FAPERJ (Procs. E-26/210.688/2019 e 211.583/2019) e UNIRIO (PPQ 2023, PPInst 2023 e IC/UNIRIO).

REFERÊNCIAS

[1] Ghadah Alamer and Sultan Alyahya. 2023. A Proposed Approach to Crowd Selection in Crowdsourced Requirements Engineering for Mobile Apps. In *Proceedings of the 7th International Conference on Information Systems Engineering*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3573926.3573927

[2] Dimo Angelov. 2020. Top2Vec: Distributed Representations of Topics. arXiv:2008.09470 [cs.CL]

[3] Jakob Axelsson and Mats Skoglund. 2016. Quality assurance in software ecosystems: A systematic literature mapping and research agenda. *Journal of Systems and Software* 114 (2016), 69–81. https://doi.org/10.1016/j.jss.2015.12.020

[4] Vittorio Dal Bianco, Varvana Myllärniemi, Marko Komssi, and Mikko Raatikainen. 2014. The Role of Platform Boundary Resources in Software Ecosystems: A Case Study. In *2014 IEEE/IFIP Conference on Software Architecture*. 11–20. https://doi.org/10.1109/WICSA.2014.41

[5] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to break an API: cost negotiation and community values in three software ecosystems. In *2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA). Association for Computing Machinery, New York, NY, USA, 109–120. https://doi.org/10.1145/2950290.2950325

[6] Zhifei Chen, Wanwangying Ma, Lin Chen, and Wei Song. 2022. Collaboration in software ecosystems: A study of work groups in open environment. *Information*

and Software Technology 145 (2022), 106849. https://doi.org/10.1016/j.infsof.2022.106849

[7] Daniela Damian, Johan Linåker, David Johnson, Tony Clear, and Kelly Blincoe. 2021. Challenges and Strategies for Managing Requirements Selection in Software Ecosystems. *IEEE Software* 38, 6 (2021), 76–87. https://doi.org/10.1109/MS.2021.3105044

[8] Iris Figalist, Christoph Elsner, Jan Bosch, and Helena Holmström Olsson. 2019. Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems. In *Agile Processes in Software Engineering and Extreme Programming*. Springer, Cham, 189–206. https://doi.org/10.1007/978-3-030-19034-7_12

[9] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. 2017. Open source software ecosystems: A Systematic mapping. *Information and Software Technology* 91 (2017), 160–185. https://doi.org/10.1016/j.infsof.2017.07.007

[10] Bachan Ghimire, Ze Shi Li, and Daniela Damian. 2024. Understanding User Feedback in Software Ecosystems: A Study on Challenges and Mitigation Strategies. In *Software Business*, Sami Hyrynsalmi, Jürgen Münch, Kari Smolander, and Jorge Melegati (Eds.). Springer Nature Switzerland, Cham, 132–147. https://doi.org/10.1007/978-3-031-53227-6_10

[11] Slinger Jansen. 2020. A focus area maturity model for software ecosystem governance. *Information and Software Technology* 118 (2020), 106219. https://doi.org/10.1016/j.infsof.2019.106219

[12] David Johnson, James Tizard, Daniela Damian, Kelly Blincoe, and Tony Clear. 2020. Open CrowdRE Challenges in Software Ecosystems. In *2020 4th International Workshop on Crowd-Based Requirements Engineering (CrowdRE)*. 1–4. https://doi.org/10.1109/CrowdRE51214.2020.00007

[13] Eric Knauss, Grischa Liebel, Jennifer Horkoff, Rebekka Wohlrab, Rashidah Kasauli, Filip Lange, and Pierre Gildert. 2018. T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*. 502–503. https://doi.org/10.1109/RE.2018.00073

[14] Eric Knauss, Aminah Yussuf, Kelly Blincoe, Daniela Damian, and Alessia Knauss. 2018. Continuous clarification and emergent requirements flows in open-commercial software ecosystems. *Requirements Engineering* 23, 1 (2018), 97–117. https://doi.org/10.1007/s00766-016-0259-1

[15] Shi Jing Koh and Fang Fang Chua. 2023. ReqGo: A Semi-Automated Requirements Management Tool. *International Journal of Technology* 14, 4 (2023), 713.

[16] Johan Linåker, Björn Regnell, and Daniela Damian. 2020. A method for analyzing stakeholders' influence on an open source software ecosystem's requirements engineering process. *Requirements Engineering* 25, 1 (2020), 115–130. https://doi.org/10.1007/s00766-019-00310-3

[17] Domia Lloyd, Ramadan Moawad, and Mona Kadry. 2017. A supporting tool for requirements change management in distributed agile development. *Future Computing and Informatics Journal* 2, 1 (2017), 1–9. https://doi.org/10.1016/j.fcij.2017.04.001

[18] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. 2010. The Small Project Observatory: Visualizing software ecosystems. *Science of Computer Programming* 75, 4 (2010), 264–275. https://doi.org/10.1016/j.scico.2009.09.004

[19] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK. http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[20] Gabriel Matute, Alvin Cheung, and Sarah Chasins. 2021. Change in Software Ecosystems. In *Plateau Workshop*. https://doi.org/10.1184/R1/19799314.v1

[21] Tom Mens and Coen De Roover. 2023. *An Introduction to Software Ecosystems*. Springer International Publishing, Cham, 1–29. https://doi.org/10.1007/978-3-031-36060-2_1

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, null (nov 2011), 2825–2830.

[23] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1908.10084

[24] Kati Saarni and Marjo Kauppinen. 2021. Requirements Engineering in the Planning Phase of a Software Ecosystem. In *Requirements Engineering: Foundation for Software Quality*, Fabiano Dalpiaz and Paola Spoletini (Eds.). Springer International Publishing, Cham, 133–148. https://doi.org/10.1007/978-3-030-73128-1_10

[25] Rodrigo Santos, Eleni Constantinou, Pablo Antonino, and Jan Bosch. 2024. Software Engineering for Systems-of-Systems and Software Ecosystems. *Information and Software Technology* 165 (2024), 107335. https://doi.org/10.1016/j.infsof.2023.107335

[26] Marco Tulio Valente. 2020. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Universidade Federal de Minas Gerais.

[27] Aparna Vegendla, Anh Nguyen Duc, Shang Gao, and Guttorm Sindre. 2018. A Systematic Mapping Study on Requirements Engineering in Software Ecosystems. *Journal of Information Technology Research (JITR)* 11, 1 (2018), 49–69. https://doi.org/10.4018/JITR.2018010104