ERS-Tutor: Uma Ferramenta Computacional Baseada em Tutoria Inteligente e Aprendizado de Máquina para o Auxílio ao Ensino de Engenharia de Requisitos

Elton Júnior da Fonseca* Luiz Alberto Ferreira Gomes†

Resumo

A Engenharia de Requisitos é uma sub-área da Engenharia de Software cujas atividades envolvem o levantamento, análise e compreensão dos requisitos de um produto de software. Estudantes e profissionais apresentam dificuldades em compreender todos os processos envolvidos para a geração do artefato documento de requisitos de forma coerente e que atenda as necessidades do cliente. Este artigo apresenta uma ferramenta computacional baseada em tutoria inteligente para auxílio ao ensino e aprendizagem dos processos de Engenharia de Requisitos de forma guiada. Uma das funcionalidades dessa ferramenta é a classificação automática os requisitos inseridos pelo usuário de acordo com sua categoria, funcionais e não funcionais.

Palavras-chaves: aprendizado de máquina. engenharia de requisitos. engenharia de software. requisitos de software. tutoria inteligente.

1 Introdução

Requisitos de um sistema de software podem ser conceituados como descrições do que o sistema deve fazer, dos serviços que ele deve oferecer, bem como das restrições ao seu funcionamento(SOMMERVILLE, 2015). Deveriam refletir plenamente as necessidades dos clientes e usuários para um sistema que servirá a uma finalidade específica, como controlar um dispositivo eletrônico, colocar um pedido em um site de compras ou encontrar alguma informação na web. Entretanto, por diversas razões, muitas equipes de desenvolvimento encontram dificuldades em atingir essa objetivo. Entre os obstáculos enfrentados − conhecidos e documentados na literatura (LARMAN, 2007; LEFFINGWELL; REINERTSEN, 2012; SOMMERVILLE, 2015) − , estão os seguintes:

- Falta de conhecimento do usuário da suas reais necessidades e, também, do que o produto de software pode lhe oferecer

*<elton.fonseca@sga.pucminas.br>
†luizgomes@pucpcaldas.br

- Ausência de conhecimento por parte dos desenvolvedores do domínio do problema;
- Comunicação inadequada entre desenvolvedores e usuários, para todos os propósitos práticos, o usuário e os desenvolvedores falam linguagens diferentes;
- Domínio do processo de elicitação, especificação e análise de requisitos pelos desenvolvedores de software.

Muitos problemas em projetos de software tem sua origem a partir de falhas cometidas pelas pessoas durante a coleta, a documentação, a especificação e a modificação de requisitos de produtos de software(WIEGERS; BEATTY, 2015). Especialistas reconhecem que erros cometidos durante as etapas de engenharia de requisitos são responsáveis por um percentual elevado de todos os defeitos encontrados em projetos de software(DAVIS, 1993; LEFFINGWELL; REINERTSEN, 2012). Uma pesquisa realizada em grandes projetos de software executados na indústria Europeia apontam que os dois mais problemas mais frequentemente reportados nesses projetos dizem respeito a especificação e o gerenciamento de requisitos(LEFFINGWELL; REINERTSEN, 2012).

A Figura 1 mostra que cerca de 40% dos projetos de software que estão em desenvolvimento fracassam por falhas nos processos de elicitação de requisitos, devido a incompreensão desses processos por parte dos desenvolvedores e em razão do entendimento incorreto do domínio do problema por parte dos clientes. Isto tudo provoca falhas na comunicação e, consequentemente, leva a equipe de desenvolvimento elaborar um documento de requisitos incoerente, inconsistente e impreciso.

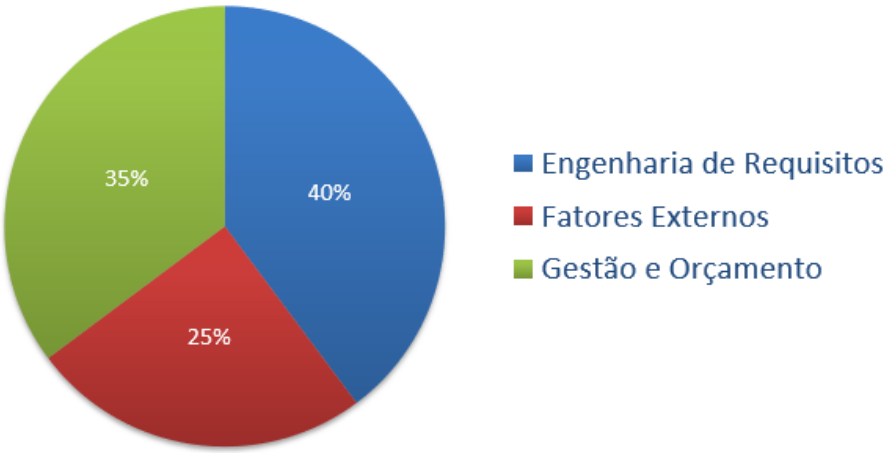

Figura 1 – Processos de um projeto de software que mais tendem levar o projeto ao fracasso

Fonte: https://www.analisederequisitos.com.br/o-que-sao-requisitos-de-software-seu-projeto-vai-fracassar/

Estudos recentes (SOMMERVILLE, 2015) confirmam que os problemas mencionados acima persistem e se aprofundam em razão da maior complexidade do software existente, levando a indústria de software a prejuízos de milhões de dólares anualmente.

Os parágrafos anteriores exibem as consequências de que muitas organizações ainda praticam métodos ineficazes para essas atividades essenciais do projeto. Suposições errôneas e não realistas sobre "o quê"o usuário realmente precisa, além de um processo casual e

não sistemático de controle de mudanças de requisitos, quase sempre leva a requisitos inadequadamente definidos e gerenciados. A raíz desses problemas está intrinsecamente relacionada à falhas na base educacional dos profissionais envolvidos.

Neste contexto, emerge a necessidade de uma ferramenta computacional que possa aprimorar as práticas de ensino de especificação de requisitos ajudando a reduzir possíveis lacunas que possam ocorrer durante o processo de aprendizagem de um aluno da área de computação. Mesmo ainda no início da sua formação - quando aluno de graduação - é primordial ensiná-lo a especificar os requisitos na forma um documento que possa ser completo, consistente e preciso. Um documento que leve ao desenvolvimento de um produto que, além de tecnicamente bem feito, satisfaça às necessidades do usuário e do cliente.

A Ferramenta ERS-Tutor, apresentada neste artigo, foi construída utilizando conceitos de Engenharia de Software, que permite a visualização de todas as etapas do processo de elicitação de requisitos e análise de casos de uso de forma orientada, onde o usuário é direcionado por um Workflow (Fluxo de Trabalho) de quais processos devem ser realizados primeiro antes de iniciar outros. Na implementação o ERS-Tutor, empregou-se ainda algoritmos de tutoria inteligente e aprendizado máquina para aprimorar o processo de ensino proposto pela ferramenta.

O restante do artigo está subdivido em 4 seções. A Seção 2 apresenta os conceitos básicos para se compreender este artigo. A Seção 3 apresenta a metodologia de trabalho. A Seção 4 detalha a ferramenta de desenvolvida. Por último, a Seção 5 conclui o artigo.

2 Conceituação Básica

Esta seção apresenta uma visão geral dos conceitos necessários compreensão deste trabalho de pesquisa, tais como requisitos de software, tutoria inteligente e aprendizado de máquina.

2.1 Requisitos de Software

Requisitos, de acordo com dicionário Aurélio da língua portuguesa, é uma condição necessária para obtenção de certo objetivo ou para procedimento de certo fim. Os requisitos são as descrições das funcionalidades de um sistema de software, o que ele deve fazer, quais serviços deve oferecer e como eles são feitos, com restrições ao seu funcionamento. Dessa forma, os requisitos são escritos em diversos níveis de detalhamento(WIEGERS; BEATTY, 2015):

- **Requisitos de Negócio** - São declarações que descrevem porque a organização está implementando o sistema.
- **Requisitos de Usuário** - Descrevem, em uma linguagem natural, opcionalmente com diagramas, os objetivos ou tarefas que o usuário deverá ser capaz de atingir ou realizar utilizando o produto.
- **Requisitos do Sistema** - São declarações mais detalhadas das funções e serviços e restrições operacionais do sistema de software para o usuário possa executar as suas tarefas, dessa forma satisfazendo aos requisitos de negócio. Dentre esses requisitos ainda se encontra 2 novas sub-categorias:

– **Requisitos Funcionais** - São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Reflete no o que o sistema deve fazer.
– **Requisitos Não Funcioanis** - São as restrições ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Reflete em como o sistema deve fazer.

2.2 Especificação de Requisitos

A Especificação de Requisotos é uma atividade responsável por descrever o sistema a ser desenvolvido em termos de requisitos funcionais e não funcionais elicitados com o usuário ou cliente. Entre as diversas abordagens para especificar requisitos que se destacam na indústria de software, a ferramenta proposta neste projeto, apoiará a linguagem natural em formato estruturado e análise de casos de uso(SOMMERVILLE, 2015).

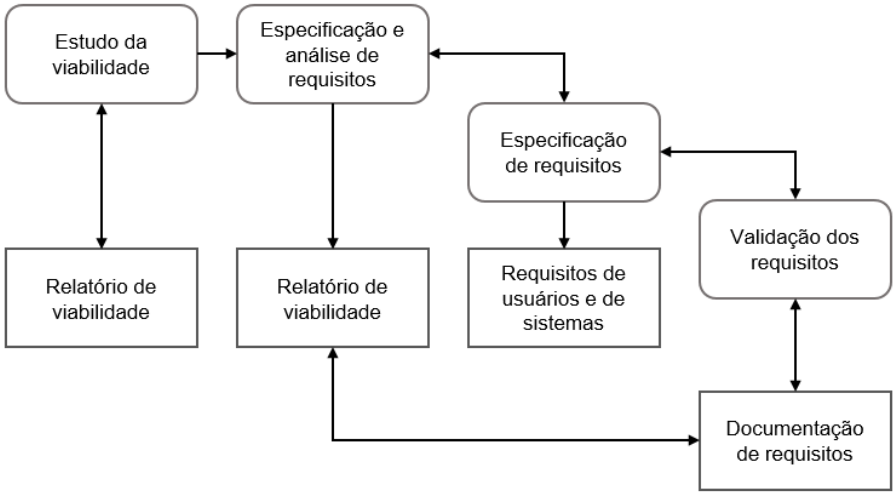

Figura 2 – Atividades do Processo de Engenharia de Requisitos.

Fonte: (SOMMERVILLE, 2015)

2.2.1 Casos de Uso

Um caso de uso é uma especificação de um conjunto de ações realizadas pelo sistema que gera um resultado observável que é tipicamente, de valor para um ou mais atores ou *stackeholders* do sistema. É uma maneira popular para especificar os requisitos funcionais do sistema e, também, uma maneira simples e compreensiva para descrever as necessidades de um usuário(SCHNEIDER, 2000).

Esses Casos de Uso são constituídos por um conjunto de cenários que possuem um objetivo de demonstrar o fluxo do sistema. Dessa forma, cada cenário é composto de uma sequência de passos que descreve uma interação entre o usuário e um sistema.

Cada Caso de Uso é subdivido em dois tipos de cenários: cenário principal e cenário alternativo. Enquanto que o cenário principal descreve a sequência de passos mais esperada para a execução da funcionalidade do caso de uso, os cenários alternativos descrevem desvios do fluxo principal. Esses fluxos podem ser especificados, incluindo uma descrição textual informal de tratamento de exceções ou fluxos alternativos que o sistema proporciona.

2.3 Validação de Requisitos

A validação de requisitos tem como objetivo garantir que as especificações dos requisitos estejam corretas, que exibam as características daquelas de assegurada qualidade e que atenda completamente às necessidades do usuário(WIEGERS; BEATTY, 2015). As técnicas modernas de validação de requisitos se apoiam em revisão, prototipação, casos de testes e análise automática de consistência(SOMMERVILLE, 2015). Essas validações permite que o documento de visão gerado como um artefato do sistema, seja completo e coerente, que realmente atenda, de forma clara, todos os requisitos necessários para satisfazer as necessidades do cliente.

A validação que será utilizada nesse projeto é baseado em heurísticas de Inteligência Artificial, usando metodologias que permitem classificar e validar os requisitos a fim de obter o objetivo. Essas validações permitirá que o usuário tenha maior compreensão de como que os requisitos são escritos e estruturados, dando apoio para a geração da documentação.

2.4 Sistema de Tutoria Inteligente

Por definição um Sistema de Tutoria Inteligente (STI) pode ser considerado como um sistema que provê instruções personalizadas e *feedbacks* aos alunos, sem a intervenção de seres humanos, ao mesmo tempo que executa a tarefa(BOYER et al., 2014). Outra definição é que "Os STI's são programas de computador com propósitos educacionais e que incorporam técnicas de Inteligência Artificial. Sistemas desta natureza Oferecem vantagens sobre os sistemas de Instrução Assistida por Computador(CAI), pois podem simular o processo do pensamento humano para auxiliar na resolução de problemas ou em tomadas de decisões"(FOWLER, 1991).

Especificamente, um STI pode ser caracterizado considerado da seguinte maneira (JONASSEN; WANG, 1993; OHLSSON, 2016):

- Conteúdo do tema ou especialidade deve ser codificado de modo que o sistema possa acessar as informações, fazer inferências ou resovler problemas.
- Sistema deve ser capaz de avaliar a aquisição deste conhecimento pelo aluno.
- As estratégias tutoriais devem ser projetadas para reduzir a discrepância entre o conhecimento do especialista e o conhecimento do aluno.

Assim, de uma forma genérica, os STI se caracterizam por representar o modelo de domínio e o modelo pedagógico separadamente, tendo como o modelo do aluno o objetivo de obter um ensino individualizado. Outra característica importante é o tutor fornecer uma interface amigável e de fácil manipulação para que favoreça uma interatividade com o aluno.

2.5 Arquitetura de um STI

O principal objetivo dos STI's é proporcionar um ensino adaptado a cada aluno, tentando se aproximar ao comportamento de um professor humano. Estes sistemas baseiam em uma arquitetura composta basicamente por 4 módulos(BOYER et al., 2014):

- **Módulo Interface:** Módulo responsável por fornecer uma interface gráfica para que o aluno possa interagir com o sistema.

- **Módulo de Especialista:** Módulo que referencia a um especialista ou modelo de domínio que contém uma descrição dos conhecimentos ou comportamentos que representam competências no objeto do domínio no qual o sistema está ensinando. Nesse caso pode ser representado, por exemplo, uma base de conhecimento, regras e soluções.
- **Módulo Estudante:** Módulo que contém descrições do conhecimento do aluno, incluindo os seus erros e acertos e lacunas do conhecimento.
- **Módulo Tutor** - Módulo responsabilizado pelas medidas corretivas, tais como fornecimento de *feedbacks* e instruções que auxiliam na orientação do conteúdo lecionado.

Essa etapa de construção do STI, exige uma descrição cuidadosa dos conhecimentos e comportamentos possíveis de especialistas, estudantes e tutores. Para que o STI possa processar essas informações, com a finalidade de fornecer *feedback* e instruções adequadas em formato da linguagem formal. A Figura 3 representa a estrutura básica de um tutor inteligente:

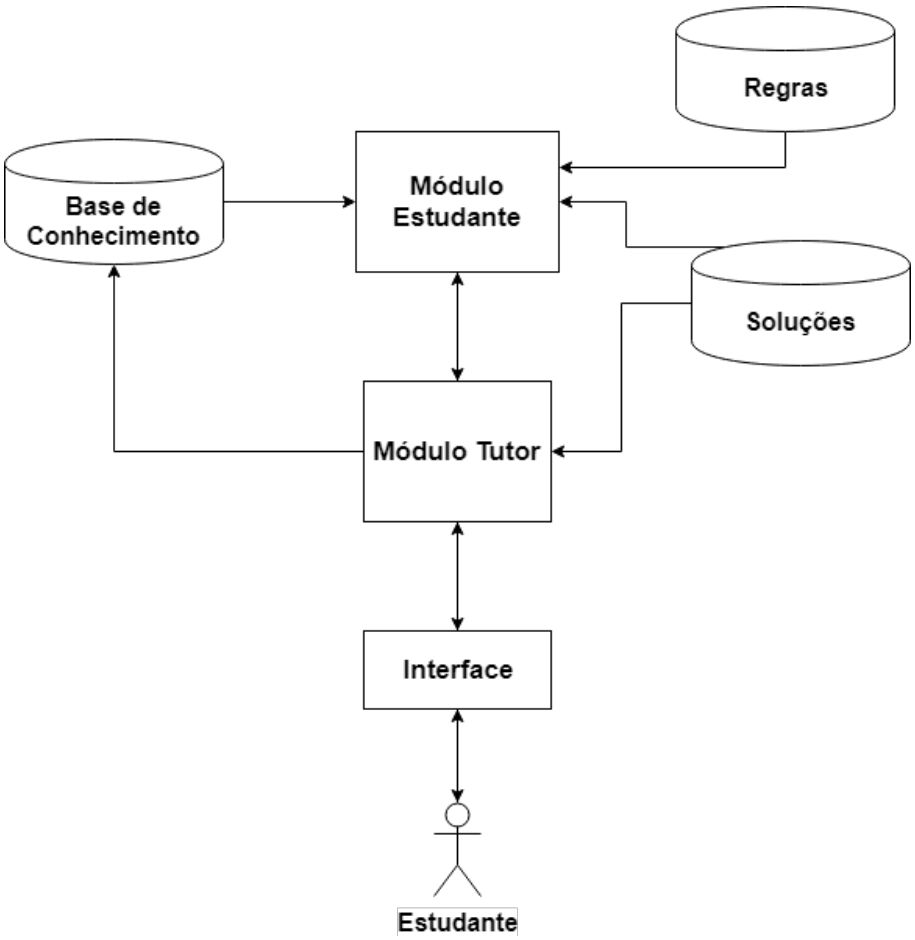

Figura 3 – Estrutura de um STI.

Fonte: (SURAWEERA; MITROVIC, 2002)

2.6 Aprendizado de Máquina

O Aprendizado de Máquina (em inglês: *Machine Learning*) (FLACH, 2012; FACELLI et al., 2015) é uma sub-área de Inteligência Artificial (AI) que provês mecanismos aos sistemas para que eles tenham capacidade de aprender e melhorar com a experiência sem serem explicitamente programados. Existem dois tipos de algoritmos de Aprendizado de Máquina(AM): preditivo (ou supervisionado) e descritivo (ou não supervisionado). Um algoritmo preditivo constrói um modelo com base em dados históricos de treinamento e usa esse modelo para prever, a partir dos valores dos atributos de entrada, um rótulo de saída (atributo de classe) para uma nova amostra. Uma tarefa preditiva é chamada de classificação quando o valor do rótulo é discreto ou regressão quando o valor do rótulo é contínuo.

Por outro lado, um algoritmo descritivo explora ou descreve um conjunto de dados. Não há um rótulo de saída associado a uma amostra. O armazenamento em cluster de dados e a descoberta de padrões são dois exemplos de tarefas descritivas. A previsão de que um requisito está bem escrito ou não é considerada um problema de classificação; portanto, mais detalhes sobre algoritmos descritivos estão fora do escopo deste artigo.

2.6.1 Support Vector Machine

O Support Vector Machine (SVM) é um algoritmo de aprendizado de máquina bastante popular onde cada vetor de características de cada instância é um ponto em um espaço muldimensional. Esse algoritmo aprende nesse espaço uma maneira ideal de separar as instâncias de treinamento de acordo com seus rótulos de classe. A saída desse algoritmo é um hiperplano, que maximiza a separação entre vetores de características de instâncias de diferentes classes. Dada uma nova instância, o SVM atribui um rótulo com base no subespaço em que seu vetor de característica pertence(FACELLI et al., 2015).

3 Metodologia

O trabalho de pesquisa adotada neste projeto baseou-se nas seguintes atividades:

- Pesquisa documental sobre os métodos de especificação de requisitos de sistemas de software baseados em casos de uso;
- Pesquisa documental sobre a aplicabilidade de heurísticas na validação de requisitos de sistemas de software;
- Pesquisa documental sobre padrões reconhecidos para documentação de requisitos de sistemas de software, e boas práticas reconhecidas e aplicáveis às atividades da engenharia de requisitos de um modo geral;
- Pesquisa documental sobre tutoria inteligente e aplicabilidade ao ensino de engenharia de software.
- Projeto arquitetônico de cada módulo da ferramenta (módulo de interface, especialista, aluno e tutor) proposta baseado em tecnologias orientadas a objetos e em tecnologias que compõe o ambiente web.
- Implementação iterativa e incremental de cada módulo da ferramenta proposta.

4 A Ferramenta ERS-Tutor

O ERS-Tutor foi desenvolvido utilizando Ruby on Rails, que é um framework Web que utiliza a linguagem Ruby com suporte ao paradigma orientado a objetos. Foi desenvolvido uma página inicial com JavaScript, HTML e CSS, com recursos gráficos sofisticados e animações que aumentam a interatividade do usuário. Além disso foi desenvolvido um painel onde o usuário irá interagir com o tutor, aplicando seus conhecimentos sobre requisitos e obtendo resultados do tutor através de um semáforo (Figura 4), onde o tutor provê *feedbacks* ao aluno.

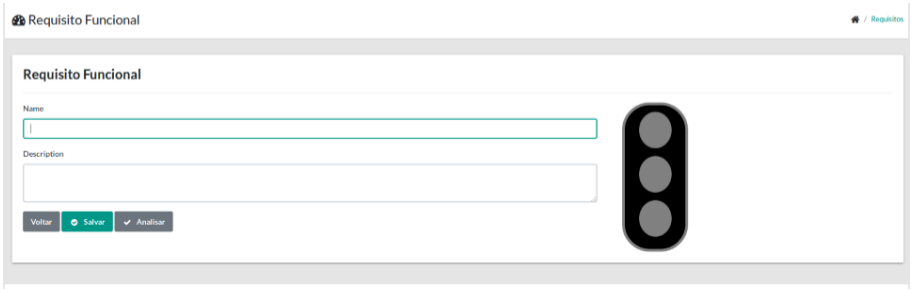

Figura 4 – Interface de Interação com o Semáforo

4.1 Workflow

O Worflow é um fluxo de trabalho (Figura 5) mostra passo a passo o processo de especificação de requisitos, ele faz parte da interface da ferramenta. Ele orienta o aluno nos processos de análise, sendo eles: levantamento de requisitos funcionais e não funcionais, casos de uso, atores e cenários. Ele permite que o aluno tem uma visão mais clara de todas as etapas do processo de elicitação. Dessa forma, as etapas seguintes só são liberadas para cadastro se as etapas obrigatórias forem concluídas.

4.2 Base de Conhecimento

A base de conhecimento da ferramenta é o coração do ERS-Tutor, pois é que acordo com essa base que o tutor irá auxiliar os *feedbacks* aos alunos. O desempenho do modelo do classificação depende da qualidade dos dados dessa base de conhecimento, pois é necessário que se tenha uma base completa com diversos exemplos para que se obtenha melhores resultados. Atualmente, a base da ferramenta conta com 122 requisitos, dentre eles funcionais e não funcionais classificados de acordo com sua categoria (Figura 6). Os requisitos da base de conhecimento foram extraídos do domínio de problema de gestão de estoques .

4.3 Web Service

Um *web wervice* é utilizado para transferir dados através de protocolos de comunicação para diferentes plataformas, independentemente das linguagens de programação utilizadas nessas plataformas. No contexto do tutor, o *web service* é utilizado para a comunicação entre a linguagem Python e o Ruby, pois o classificador é escrito em Python, enquanto que o ERS-Tutor é escrito em Ruby. Dessa forma houve a necessidade de se utilizar a ferramenta *Flask* em Python para a construção do Web Service, onde se encontrará o SVC. A comunicação entre os dados é feita em notação JSON, mais flexível com

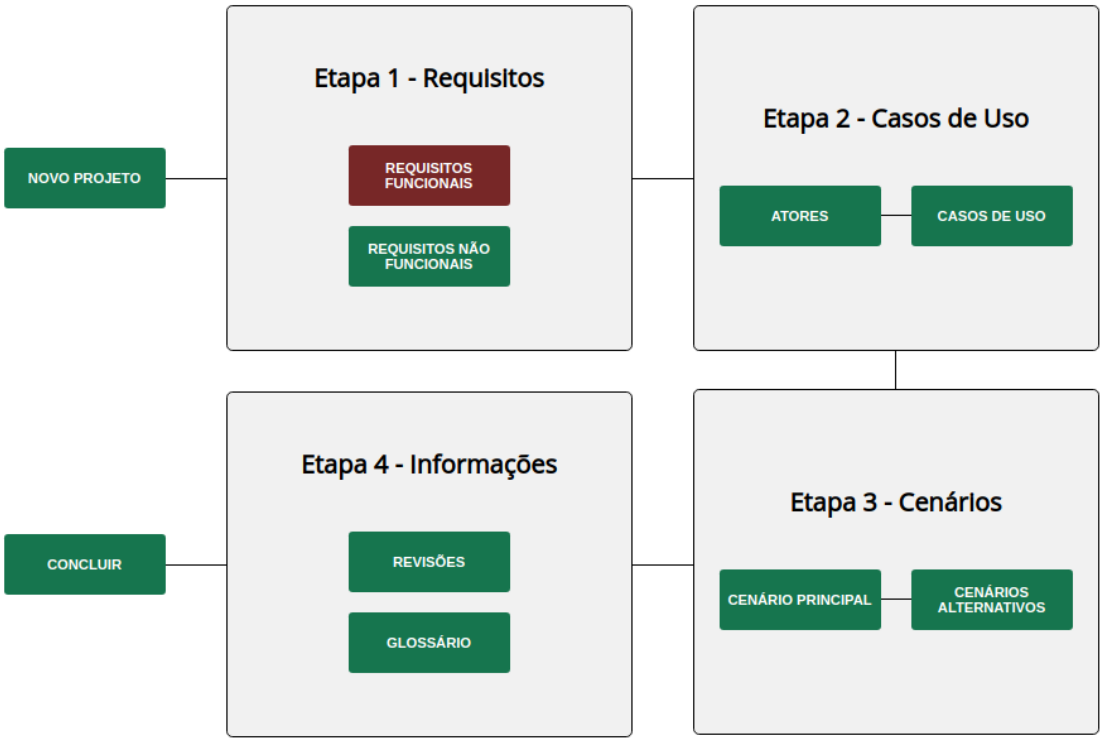

Figura 5 – Workflow de Requisitos

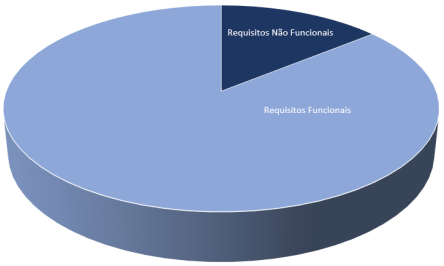

Figura 6 – Distribuição de tipos de requisitos na base de conhecimento.

mais facilidade de trabalhar. A Figura 7 apresenta o esquema de comunicação entre as aplicações com base em *webservices*.

4.4 Máquina de Aprendizado

O ERS-Tutor utilizou um classificador SVM escrito na linguagem Python e na biblioteca *scikit-learn* para o treinamento da base de conhecimento. Para isto, primeiramente, um processo de pré-processamento para remover todas as palavras que não fazem sentido para o contexto do problema foi executado. Esse pré-processamento foi feito usando os recursos a biblioteca Python NLTK(NLTK, 2019). Após isto, foi feito um cálculo probabilístico que identifica a importância de uma palavra no documento. Esse cálculo baseou-se na métrica *Term Frequency Inverse Document Frequency* (TF-IDF) expressa na fórmula(SCIKIT-LEARN, 2019):

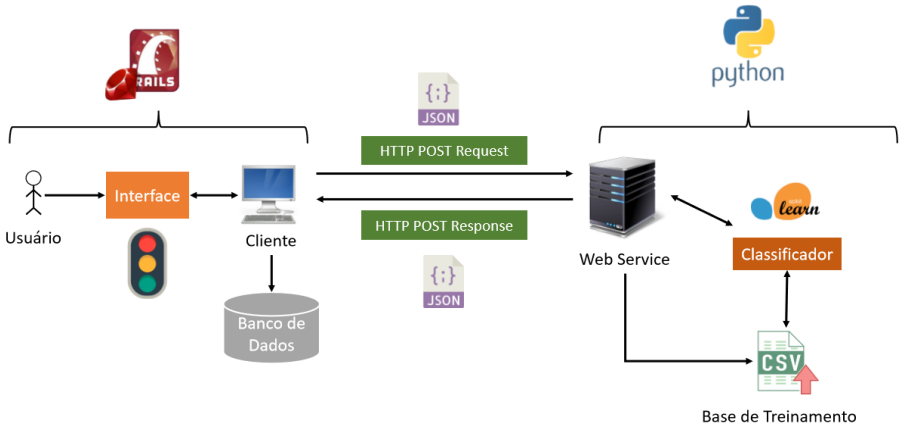

Figura 7 – Arquitetura baseada em web service.

$$W_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}$$

Cada instância então é transformada em um conjunto numérico que representa essa instância, resultando em uma matriz que identifica a importância de cada termo nos documentos. Isto é necessário pois o classificador SVM só permite entradas numéricas. Por último, 30% das instâncias da base de conhecimento foram reservadas para testes e 70% para treinamento.

O SVM efetua o treinamento das instâncias e realiza a construção de um mapa de instâncias linearmente separados, com base na seguinte fórmula(SCIKIT-LEARN, 2019):

$$sgn(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

Dessa forma, o classificador encontra padrões e aprende o conceito, assim podendo predizer as próximas entradas que o aluno for realizando.

5 Considerações finais

Este artigo apresentou uma ferramenta para auxílio ao ensino dos processos de engenharia de requisitos, incluindo a especificação de requisitos baseada em casos de uso. Essa ferramenta foi desenvolvida utilizando recursos de tutoria inteligente e aprendizado de máquina para melhorar a experiência de ensino ao aluno. Embora não tenham sido realizados testes de maneira ampla e profunda, os resultados iniciais com a avaliação dos algoritmos de aprendizado e da própria ferramenta, apontam para um caminho promissor.

Futuramente, outros domínios de problemas, bem como o aumento das instâncias de requisitos serão inseridos na ferramenta em questão para melhoria da sua acurácia. Além disto, vislumbra-se colocá-la disponível na *web* para que alunos e profissionais de todo o país possam utilizá-la.

Referências

BOYER, K. E. et al. *Intelligent tutoring systems*. 1ª. ed. Honolulu: Springer International Publishing, 2014. ISBN: 9783319072210. Citado na página 5.

DAVIS, A. M. *Software requirements : objects, functions, and states*. 1ª. ed. Englewood Cliffs, N.J.: PTR Prentice Hall, 1993. ISBN: 013805763X. Citado na página 2.

FACELLI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. Rio de Janeiro, RJ, Brasil: LTC, 2015. ISBN 0521196000, 9780521196000. Citado na página 7.

FLACH, P. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. New York, NY, USA: Cambridge University Press, 2012. ISBN 1107422221, 9781107422223. Citado na página 7.

FOWLER, D. G. A model for designing intelligent tutoring systems. *Journal of Medical Systems*, Volume 15, p. 47–63, 1991. Citado na página 5.

JONASSEN, H. D.; WANG, S. The physics tutor: Integrating hypertext and expert systems. *Journal of Educational Technology Systems*, Volume 22, p. 19–28, 1993. Citado na página 5.

LARMAN, C. *Utilizando UML e padrões: uma introdução á análise e ao projeto orientados a objetos e desenvolvimento irativo*. 3ª. ed. Porto Alegre: Bookman, 2007. ISBN 9788577800476. Citado na página 1.

LEFFINGWELL, D.; REINERTSEN, D. G. *Agile software requirements : lean requirements practices for teams, programs, and the enterprise*. 4ª. ed. Upper Saddle River (NJ): Addison Wesley, 2012. (The agile software development series). ISBN: 9780321635846. Citado 2 vezes nas páginas 1 e 2.

NLTK. *Natural Language Toolkit*. [S.l.: s.n.], 2019. Disponível em: <https://www.nltk.org/>. Citado na página 9.

OHLSSON, S. Constraint-based modeling: from cognitive theory to computer tutoring – and back again. *International Journal of Artificial Intelligence in Education [electronic only]*, Springer US, New York, NY; International Artificial Intelligence in Education (AIED) Society / IAIED (Society), Leeds, England, v. 26, n. 1, p. 457–473, 2016. ISSN 1560-4292. Citado na página 5.

SCHNEIDER, G. *Applying use cases a practical guide*. 1ª. ed. Boston: Addison Wesley, 2000. ISBN: 0201309815. Citado na página 4.

SCIKIT-LEARN. *Support Vector Machine*. 2019. Disponível em: <https://scikit-learn.org/stable/modules/svm.html>. Citado 2 vezes nas páginas 9 e 10.

SOMMERVILLE, I. *Software Engineering*. 10th. ed. [S.l.]: Pearson, 2015. ISBN 0133943038, 9780133943030. Citado 4 vezes nas páginas 1, 2, 4 e 5.

SURAWEERA, P.; MITROVIC, A. Kermit: A constraint-based tutor for database modeling. In: CERRI, S. A.; GOUARDÈRES, G.; PARAGUAÇU, F. (Ed.). *Intelligent Tutoring Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 377–387. ISBN 978-3-540-47987-1. Citado na página 6.

WIEGERS, E. K.; BEATTY, J. *Software requirements*. 3ª. ed. Washington: Microsoft Press, 2015. ISBN: 9780735679665. Citado 3 vezes nas páginas 2, 3 e 5.