UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

ADELMO JOSÉ CABRAL DE ALMEIDA

MAPEAMENTO DE INFORMAÇÕES DE GARANTIA DE SEGURANÇA DE ISSUE TRACKING SYSTEMS PARA USO EM ONTOLOGIAS

Recife
2021

ADELMO JOSÉ CABRAL DE ALMEIDA

MAPEAMENTO DE INFORMAÇÕES DE GARANTIA DE SEGURANÇA DE ISSUE TRACKING SYSTEMS PARA USO EM ONTOLOGIAS

Monografia da Disciplina de Trabalho de Graduação do Curso de Engenharia da Computação da Universidade Federal de Pernambuco.

Orientador(a): Prof.ª Dr.ª Carla Taciana Lima Lourenço Silva Schuenemann

Co-orientador(a): Prof. Camilo Camilo Almendra

Recife
2021

AGRADECIMENTOS

Inicialmente, gostaria de agradece a Deus, pelo privilégio de chegar até aqui nesse momento atual de pandemia que vivemos e pela oportunidade de poder estudar numa universidade de referência, e que se Ele quiser e a banca permitir, poder finalizar o curso de Engenharia da Computação.

À minha mãe, Maria do Carmo, que desde o ano passado não está mais no plano terrestre, me deu a vida e me criou só, com todas as dificuldades vividas nunca deixou faltar nada dentro de casa, minha referência em caráter e honestidade. Mãe, onde a senhora tiver, esse trabalho é pra VOCÊ!

Ao meu filho, Bernardo, uma dádiva que Deus me deu, teve paciência nas suas férias escolares em ficar brincando e assistindo de maneira silenciosa, me permitindo concentrar e realizar esse trabalho, me dando força, com sua alegria e entusiasmo, tudo é por ele e pra ele.

Aos meus orientadores, Professora Carla e Professor Camilo, que me propiciaram realizar esse trabalho, pelo material e referencial teórico, pelos ensinamentos e paciência demonstrada comigo, sou muito grato a vocês.

Ao orientador da cadeira de TG, professor Adriano, entendeu minha dificuldade inicial no processo dentro da disciplina e fez a ponte com a professora Carla, meu muito obrigado.

Ao irmão que o CIn me deu, Lúcio Ribeiro, sempre me apoiou dentro da universidade e fora dela, muitos conselhos, muitas oportunidades, me ajudou em algumas dúvidas técnicas no desenvolvimento desse trabalho, mesmo na correria com meu sobrinho Inácio de apenas 1 ano.

“Não sou um Manoel Bandeira,

Drummond, nem Jorge de Lima;

Não espereis obra prima

Deste matuto plebeu!

Eles cantam suas praias,

Palácios de porcelana,

Eu canto o chão, a cabana

Canto o Sertão, que ele é meu.”

Rogaciano Leite

RESUMO

Atualmente as empresas certificadoras estão na sua maioria cobrando certificações de garantia que atestem a confiabilidade e o bom funcionamento de sistemas críticos. Essas certificações englobam uma série de documentações que mostram argumentos em nível de projeto de forma detalhada que garantam que o sistema está pronto para uso. A partir de um estudo onde mostrou a possibilidade de elaboração de modo iterativa e incremental de casos de garantia de segurança em conjunto com a engenharia de requisitos, personalizando um sistema de gerenciamento ágil de projeto personalizado em um metamodelo, surgiu a necessidade de realizar um mapeamento dos itens de segurança desse sistema em formato que alimente uma ontologia para que se possa gerenciar o conhecimento da informação dos itens de segurança de sistemas.

ABSTRACT

Currently, most certification companies are demanding warranty certifications that attest to the reliability and proper functioning of critical systems. These certifications encompass a series of documentation that show detailed project-level arguments to ensure that the system is ready to use. From a study where it showed the possibility of iteratively and incrementally elaborating security assurance cases in conjunction with requirements engineering, customizing a custom agile project management system in a metamodel, the need to carry out a mapping arose. of the security items of that system in a format that feeds an ontology so that knowledge of the information of the system security items can be managed.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Os sistemas críticos estão cada vez mais presentes no cotidiano das pessoas, desde viagens de avião e trem, a sistemas da área médica e nos negócios. Falhas nesses sistemas podem trazer problemas a vidas das pessoas ou perdas ponto de vista econômico. O desenvolvimento desses sistemas se não for bastante analisado e documentado pode acarretar problemas de certificação ou descoberta tardia de falhas de segurança, consequentemente aumentando os custos e o retrabalho (HATCLIFF *et al*, 2014).

Uma das documentações que são exigidas em certificações de sistemas críticos é o caso de garantia de segurança, que tem por objetivo apresentar uma visão clara e abrangente e argumento defensável de que um sistema é considerado seguro para ser utilizado em um determinado ambiente (BLOOMFIELD e BISHOP, 2010). Esses casos de garantia de segurança estão cada vez mais sendo exigidos como pré-requisitos em vários padrões e normas relacionados a sistemas críticos (HAWKINS *et al*, 2013).

Iremos realizar a contextualização deste trabalho, o objetivo e como o mesmo estará estruturado.

1.1. CONTEXTUALIZAÇÃO

Com o desenvolvimento ágil de projetos, onde são realizadas entregas interativas e incrementais, é notório o benefício na construção de sistemas críticos. Ge et al (2010) afirma que diferentemente do desenvolvimento de sistemas não críticos onde é priorizada a satisfação das necessidades do cliente, no caso de sistemas críticos, a qualidade do sistema é tão importante quanto à funcionalidade. Um desafio na utilização de metodologias ágeis no desenvolvimento de sistemas críticos é tratar a engenharia de requisitos e casos de garantia em segurança de forma integrada na vida útil do projeto (ALMENDRA; SILVA; BARROS, 2019).

O projeto de pesquisa de Almendra, Barros e Silva (2019) propõe um framework para gerar casos de garantia de segurança a partir de informações armazenadas em ferramentas de gerenciamento de projetos, em especial ITS. Esse framework fornece suporte para que informações regulares encontradas em projetos de software sejam agregadas com informações de garantia de segurança. O framework prevê a extração automática de informações das ferramentas de projetos, para serem processados por uma ontologia (Figura 1 A e B).

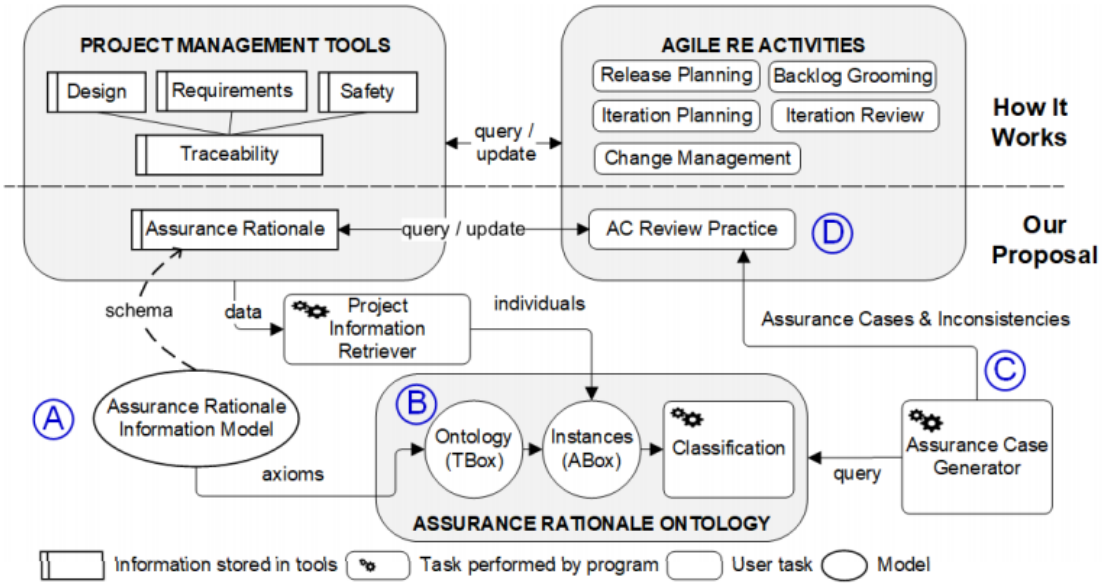

Figura 01. Framework que integra Casos de garantia com Agile RE (Almendra, Barros e Silva, 2019)

Dentro do contexto do projeto de pesquisa, Almendra e Silva (2020) propuseram o metamodelo de gerenciamento de informações chamado ARM (*Assurance Rationale Model*) que pode ser usado em conjunto com ferramentas de gerenciamento de issues (ITS - *Issue Tracking System*), tais como Azure Devops, Gitlab e Atlassian Jira (Figura 2). Tais ferramentas são bastante usadas no desenvolvimento ágil de software, que está cada vez mais em adoção no desenvolvimento de sistemas críticos (HEEAGER e NIELSEN, 2018). O uso do modelo ARM em conjunto com um ITS permite os desenvolvedores registrar informações mais ricas dos projetos, incluindo aquelas necessárias para a posterior elaboração da documentação de certificação de sistemas críticos.

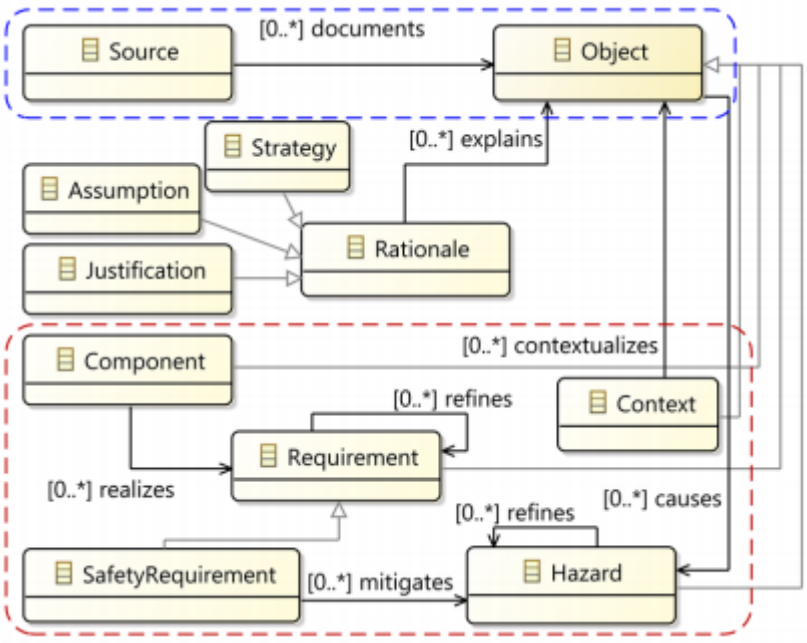

Figura 02. Metamodelo ARM
(ALMENDRA, SILVA 2020)

1.2. OBJETIVO

Esse trabalho tem como objetivo implementar uma ferramenta para mapear informações de projeto de um ITS adaptado ao metamodelo ARM para o formato de entrada de uma ontologia criada também com base no ARM. Como cada ITS pode ser configurado de diferentes formas para o mesmo metamodelo do ARM, a ferramenta implementa as heurísticas de mapeamento das informações.

1.3. ESTRUTURA DO TRABALHO

O presente trabalho de graduação está organizado da seguinte forma:

a) **Capítulo 2:** apresentação do referencial teórico, que servirá como fundamento para o leitor conhecer os conceitos utilizados neste documento. Conceitos de metodologia ágil, casos de garantia de segurança, metamodelo ARM, sistemas de gerenciamento de issues e framework RDF.

b) **Capítulo 3:** apresenta a ferramenta criada para fazer o mapeamento de itens de um projeto cadastrado em um Issue Tacking System personalizado. Este módulo foi dividido como: descrição do extrator, como foi construído e que linguagem de programação foi utilizada e como usá-lo.
c) **Capítulo 4:** apresenta as considerações finais referentes ao objetivo atingido da ferramenta, mostrando sua contribuição. Pra concluir, mostraremos algumas recomendações que podem ser utilizados para um trabalho futuro.

2. REVISÃO DA LITERATURA

Nesta seção iremos apresentar o referencial teórico que estrutura a ideia deste trabalho, mostrando conceitos de casos de garantia no desenvolvimento de sistemas críticos. Vamos mostrar também as ideias centrais de desenvolvimento de software utilizando metodologias ágeis, bem como sistemas de gerenciamento de issues que são utilizados como suporte nesse desenvolvimento. Será apresentado o estudo do ARM (Assurance Rational Model), metamodelo mostrado em (ALMENDRA, SILVA, 2020). Por fim o conceito e utilização do framework RDF.

O objetivo dessa revisão não é ser minuciosa e tratar de forma exaustiva os assuntos, mas apresentar de forma a apoiar no entendimento e contextualizar os conceitos do presente trabalho. Este capítulo está estruturado da seguinte forma: a Seção 2.1 apresenta as definições de desenvolvimento ágil de software, a Seção 2.2 apresenta as definições de casos de garantia de segurança, a seção 2.3 aborda sistemas de gerenciamento de issue (ITS), a seção 2.4 apresentará o ARM, por fim, na seção 2.5 serão apresentadas as definições do framework RDF.

2.1. DESENVOLVIMENTO ÁGIL DE SOFTWARE

Em 2001, 17 pessoas reconhecidas na área de engenharia de software se reuniram para discutir formas de desenvolvimento de software de maneira mais simples. Chegaram ao consenso e formalizaram um documento conhecido como Manifesto Ágil, onde apresentava as crenças e valores que consideravam essenciais para o desenvolvimento de projetos de software. Os quatro valores são mostrados na figura 03 abaixo:

Figura 03. Valores Manifesto Ágil
Aela.io(n, d)

Os itens que não estão destacados da figura acima têm seus valores, mas procura-se priorizar mais os itens destacados.

Nos últimos anos, está cada vez mais comum às empresas adotarem as metodologias ágeis, os principais motivos são: acelerar a entrega dos produtos, melhorar a produtividade e melhorar o gerenciamento de prioridades.

No desenvolvimento ágil de software, os projetos são realizados de forma mais flexível e iterativa, são realizadas pequenas entregas de acordo com o seu valor para o negócio das empresas diminuindo assim os riscos com erros e alterações de escopo. No desenvolvimento tradicional, conhecido como Waterfall, qualquer alteração no projeto pode ser bastante custosa dependendo do andamento na construção do mesmo, vide Figura 04.

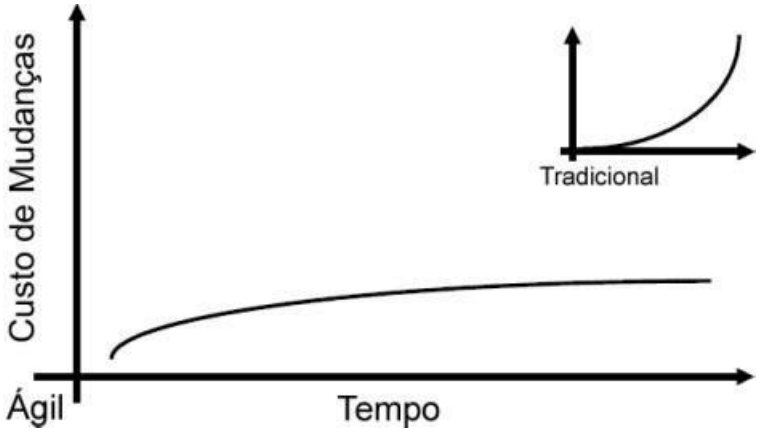

Figura 04. Custo alteração desenvolvimento Ágil x Tradicional
(DEVMEDIA, n. d.)

2.2. CASOS DE GARANTIA DE SEGURANÇA

Casos de garantia de segurança é uma abordagem de tendência para se comunicar e discutir sobre a segurança de um sistema (HAWKINS *et al*, 2013), oferecendo mais flexibilidade para justificar a recuperação do software peculiar e decisões de projeto (R. Bloomfield and P. Bishop, 2010). Esses casos compreendem uma junção de características como requisitos do produto e padrão, decisões de projeto e justificativas para tal e conclusões de risco, estabelecem argumentos e provas detalhadas de que um sistema é seguro ou atingiu o nível exigido de garantia e ou confiança.

Originalmente os documentos de casos de garantia eram escritos de forma textual, incluindo muitas páginas não estruturadas. Com o passar do tempo, se viu a necessidade de uma organização mais formal, desenvolvendo textos mais formais e notações gráficas para o conceito de estruturas de argumentos (T. Kelly, 2018). Essas estruturas são:

- Requisitos de alto nível e objetivos do sistema;
- Itens de evidência que suportam a satisfação dos requisitos e objetivos;
- Argumentos que explicam e justificam como os itens de evidência se relacionam aos requisitos e objetivos;

Na figura 05, temos os conceitos e como se relacionam da notação GSN(*Goal Structured Notation*) bastante utilizada em casos de garantia. Essa notação possui os seguintes componentes: **objetivo**, representado por um retângulo, **estratégia de argumentação**, representado por um paralelogramo e **solução**, apresentado através de um círculo. O contexto é usado para suporte de informações de metas. Suposições e justificativas podem ser usadas para apoiar o argumento (Tech| 2021, n. d.). Exemplo na Figura 05:

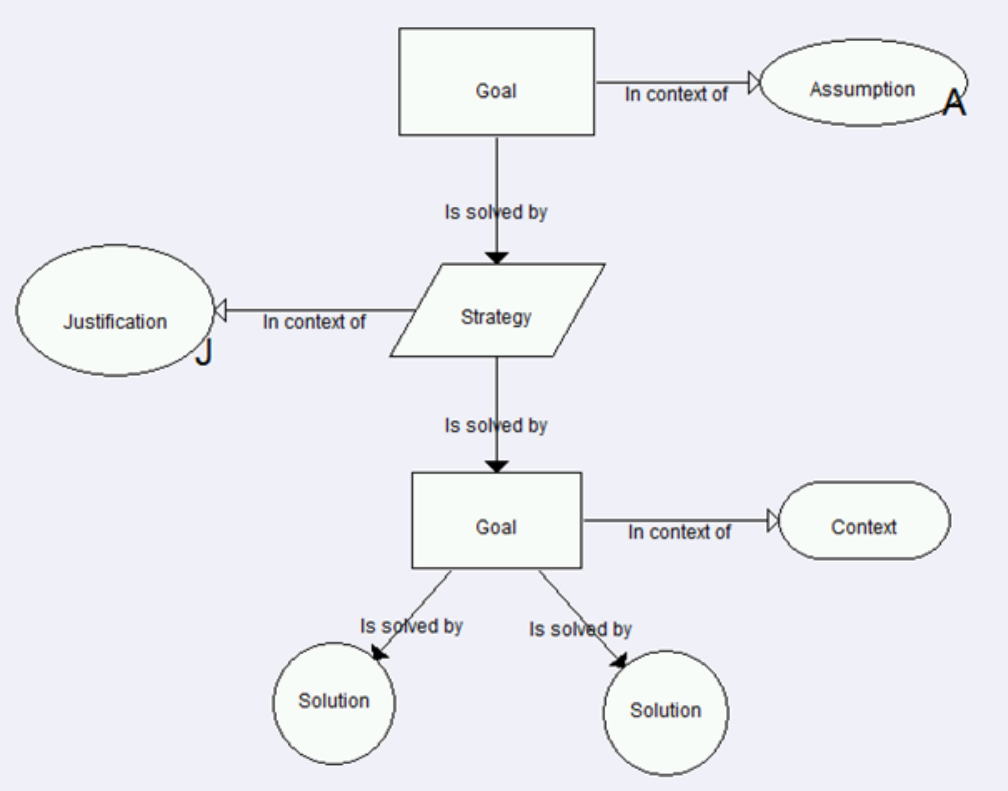

Figura 05. Notação estruturação GSN
(ADELARD, n. d.)

Abaixo, é exibido na Figura 06 um exemplo parcial de um caso usando o GSN que descreve uma estrutura de argumento de segurança mostrando como o projeto de uma bomba de infusão lida com o perigo relacionado até o esgotamento da bateria (ALMENDRA, SILVA, 2020).

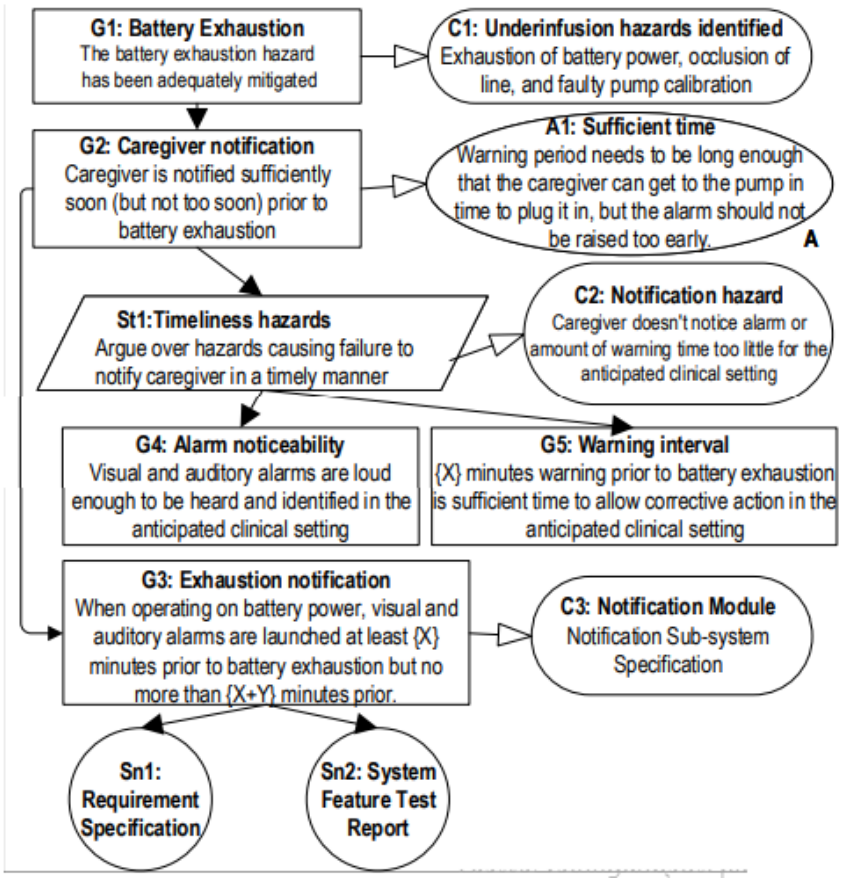

Figura 06. Exemplo de argumento de caso de garantia de segurança (adaptado de WEINSTOCK e GOODENOUGH, 2009).

2.3. ARM

Os ITS (*Issue Tracking Systems*) são ferramentas de apoio ao desenvolvimento ágil de software fornecendo infraestrutura para o ciclo de vida do projeto. As informações como requisitos, links, questões, entre outras, podem ser gerenciadas e adaptadas de acordo com as necessidades de cada equipe de desenvolvimento. Enriquecendo-os projetos com segurança e informações lógicas, evitamos a necessidade de manter artefatos separados, fornecer melhor suporte para classificação automática, análise, e geração de casos de garantia. (ALMENDRA, SILVA, 2020).

Em (ALMENDRA; SILVA, 2020), foi apresentado o ARM (*Assurance Rationale Model*) um metamodelo que serve como base na gestão das informações e fundamentos de garantias dentro do ITS, aumentando assim o desenvolvimento de casos de garantia de segurança de um projeto.

O ARM foi baseado em 2 metamodelos: O metamodelo de rastreamento de requisitos de Ramesh e Jarke (RAMESH; JARKE. 2001), focado em 3 componentes (objetos, partes interessadas e fontes), e o modelo de análise de requisitos e especificação de requisitos em segurança (VILELA *et al.*, 2018). A figura 07 mostra os conceitos e relações referentes ao metamodelo ARM (ALMENDRA, SILVA, 2020):

Concept	Description
Object	Input or output of the development process, in any level of granularity, that is managed in the ITS.
Source	An external artefact to the ITS.
Rationale	An explanation that provides additional information for an Object.
Strategy	A statement behind a project's decision. It can be a design decision, a requirements decision, or a mitigation strategy selected for a hazard.
Justification	A statement that provides an explanation for a project decision.
Assumption	A statement that reflects a common situation in the domain so ingrained into stakeholders general beliefs, that it is unquestioned.
Hazard	A single hazard or category of hazards.
Requirement	A system requirement.
SafetyReq.	A requirement related to system safety. It represents actions and constraints should or should not be performed to maintain the system in a safe state.
Context	An environmental or operational condition that may occur, and it is relevant to safety.
Component	A unit of architecture, design or code of the system.
Relationship	**Description**
documents	It links a Source to Objects that it documents.
refines	It represents a refinement of Requirement or Hazard.
mitigates	It links a SafetyRequirement to the Hazards that it addresses.
causes	It links a Hazard to Objects that are identified as its cause.
explains	It links a Rationale to Objects it provide explanation.
realizes	It links a Component to Requirements allocated to it.
contextualizes	It links a Context to Objects.

Figura 07. Conceitos e relações ARM
(ALMENDRA, SILVA, 2020)

Na configuração do ARM em um ITS, o estudo utilizou o Jira, a ferramenta mais utilizada para o desenvolvimento e gerenciamento ágil de software por equipe ágeis (Atlassian, n. d.). Por ser bastante flexível, a ferramenta permite realizar personalização de campos e issue, facilitando a customização, podendo assim incluir as informações de segurança em conjunto com informações de requisitos do projeto. Na tabela 01 abaixo, a personalização do Jira para uso do ARM é apresentada.

Item do modelo ARM	**Como é registrado no Jira**
Assumption	Assumption field (custom)
Component	Component configuration
Context	Context issue (custom)
Hazard	Hazard issue (custom)
Justification	Justification field (custom)
Object	Any kind of issue, labels and components
Requirement	Story issue
SafetyReq.	Story issue with Label “Safety”
Source	Any weblink/attachment associated to an issue
Strategy	Strategy issue (custom)
causes	i) causes issue link (custom), or ii) a Component associated to Hazard issue
contextualizes	Context issue linked by relates to any issue
documents	Weblinks/attachments associated with an issue

explains	Explains issue link (custom)
refines	Refines issue link (custom)
mitigates	Mitigates issue link (custom)
realizes	Component associated to Story issue

Tabela 01. Personalização do Jira com conceitos ARM

2.4. RDF

Segundo (W3C, 2014), RDF – Resource Description Framework é um framework para representar informações na web. Essas representações são chamadas de recursos e inicialmente foi pensado como modelo de dados para metadados, mas atualmente está sendo bastante utilizado para descrição conceitual ou de representação da informação e no uso de gerenciamento do conhecimento.

A sintaxe RDF é bastante simples, consiste em uma estrutura de triplas, consistindo em sujeito, predicado e objeto. Onde os recursos são relacionados pelo sujeito e o objeto e o predicado figura-se como a natureza desta relação. Uma coleção de triplas é conhecida como grafo RDF Figura 08.

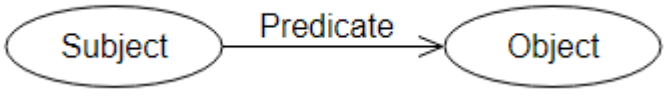

Figura 08. Grafo RDF
(W3C, 2021)

Utilizando esse tipo de declaração que estabelece um modelo de recursos, propriedades e seus valores objeto correspondentes, o RDF tenta proporcionar uma

maneira clara e não ambígua de representar a semântica dos dados por meio de uma codificação para ser compreendida pela máquina (Miller, 2018).

Existem algumas formas para definir um documento RDF, para o contexto desse trabalho, será explicada a definição de N-triplas (W3C, 2014) que nada mais é que uma sequência de triplas RDF separadas por um ponto. Esse formato permite que um recurso possa ser utilizado em várias triplas. Um recurso que é um sujeito de uma tripla pode ser referenciado como objeto em outra, tornando possível realizar conexões entre triplas. Em uma coleção de dados RDF, as relações entre triplas através de formato de grafos, onde sujeito e objeto são representados por vértices e o predicado por arestas Figura 09.

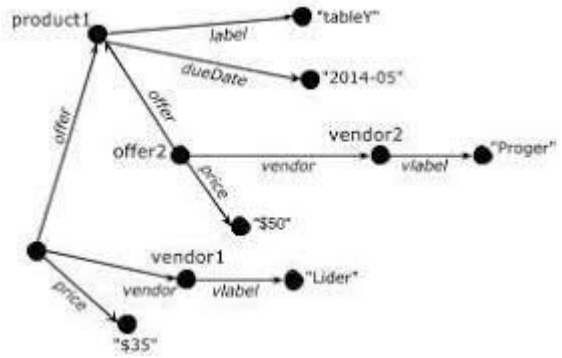

Figura 09. Conjunto de dados RDF
(Gasparini, Schroeder. 2018)

3. EXTRATOR DE MAPEAMENTO DE DADOS

Neste capítulo iremos apresentar o estudo e o contexto que serviram como base para o desenvolvimento da ferramenta que faz o mapeamento dos dados de projeto cadastrado em um ITS personalizado com os conceitos do ARM para o formato de entrada de uma ontologia criada com base também no ARM. Será mostrada a forma como foi construído, como configurado e a saída esperada.

ALMENDRA (2017) mostra um estudo onde se procura realizar a engenharia de requisitos e elaboração de casos de garantia de forma integrada e interativa, foi mostrado que a falta dessa integração pode levar a problemas de segurança no desenvolvimento de sistemas críticos.

Através de personalização de um ITS, no caso desse estudo o Jira, incluindo os conceitos do ARM (ALMENDRA, SILVA, 2020), integrando recursos de engenharia de requisitos com informações de segurança do projeto.

3.1. ASPECTOS TECNOLÓGICOS

A ferramenta foi construída em cima da linguagem Java amplamente difundida no meio de desenvolvimento de software que pesquisa realizada em 2019 foi a linguagem mais utilizada no mundo (COMPUTER WORD, *n.d.*). A escolha levou em consideração o conhecimento do autor e da orientação deste trabalho.

A linguagem Java é baseada no paradigma da orientação a objetos. Desenvolvida na década de 90 pela Sun Microsystems, sendo posteriormente adquirida pela Oracle em 2008. A linguagem permite a modularização das aplicações, facilitando a manutenção e reuso de código. A linguagem pode ser utilizada para uma grande variedade de aplicações de diferentes plataformas.

O Java possui um vasto conjunto de bibliotecas e APIs que facilitam o desenvolvimento de aplicações. Para esse trabalho utilizamos a biblioteca openCSV.jar, que é um analisador CSV (*Comma-separated values*) que oferece

suporte para todas as operações básicas (leitura e escrita). necessária para a leitura e processamento de arquivos CSV (Comma-separated values).

Foi utilizado o Jira de forma personalizada como já citado anteriormente com os itens do conceito do metamodelo ARM cadastrados de maneira integrada, os requisitos do projeto com os itens de segurança.

3.2. PROJETO EXEMPLO

Na ferramenta Jira, seguindo a personalização descrita na Tabela 02, informações de um projeto de exemplo foram registradas. Esse projeto exemplo representa um sistema de bomba de infusão de analgésicos, baseado nos artefatos abertos do projeto Open PCA Pump (HATCLIFF *et al*., 2018). Os itens de projetos estão descritos no quadro abaixo.

Item	Tipo	Relacionamentos
Normal operation	Requirement	
Clinician starts/stops pump	Requirement	
Basal flow rate limits	SafetyRequirement	
Basal flow rate tolerance	SafetyRequirement	
Measure flow	SafetyRequirement	
Flow rate alarm	SafetyRequirement	
Patient authentication	Requirement	

Dose of Drug Request	Requirement	
Clinician request a dose	Requirement	
Patient requests a dose	Requirement	
Minimum time between patient-requested doses	SafetyRequirement	
Maximum flow rate	SafetyRequirement	
Improper flow	Hazard	caused by Clinician starts/stops pump mitigated by Measure flow mitigated by Flow rate alarm
Underinfusion	Hazard	caused by Clinician starts/stops pump mitigated by Basal flow rate limits mitigated by Basal flow rate tolerance
Overinfusion	Hazard	caused by Clinician starts/stops pump caused by Clinician request a dose caused by Patient requests a dose mitigated by Maximum flow rate mitigated by Minimum time between patient-requested doses mitigated by Basal flow rate limits mitigated by Basal flow rate tolerance
Control Panel	Component	realizes Clinician starts/stops pump realizes Clinician request a dose realizes Patient requests a dose
Pump Controller	Component	realizes Basal flow rate limits realizes Basal flow rate tolerance realizes Minimum time between

		patient-requested doses realizes Maximum flow rate
Flow Checker	Component	realizes Measure flow realizes Flow rate alarm
Security	Component	realizes Patient authentication

Tabela 02. Informações registradas no Jira para o projeto de exemplo

3.3. UTILIZAÇÃO DA FERRAMENTA

Serão mostradas nessa seção como geramos o arquivo de entrada do extrator através de um projeto configurado de forma personalizada no ITS Jira, como preparar a ferramenta para ser executada, a sua execução e o resultado esperado.

3.3.1. EXPORTANDO ARQUIVO NO JIRA

Para exportar o arquivo primeiramente o usuário deve acessar o endereço eletrônico do ITS Jira[1], fazendo o login Figura 10 e posteriormente buscar o projeto configurado e personalizado com o metamodelo ARM.

[1] https://arcade-dare.atlassian.net

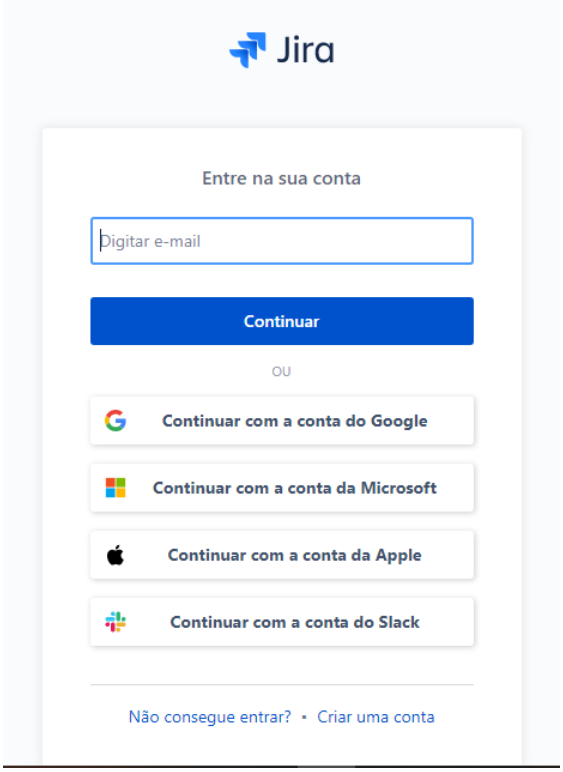

Figura 10. Login no Jira

Após realizar o login, o Jira irá exibir todos os projetos em que seu usuário faz parte Figura 11.

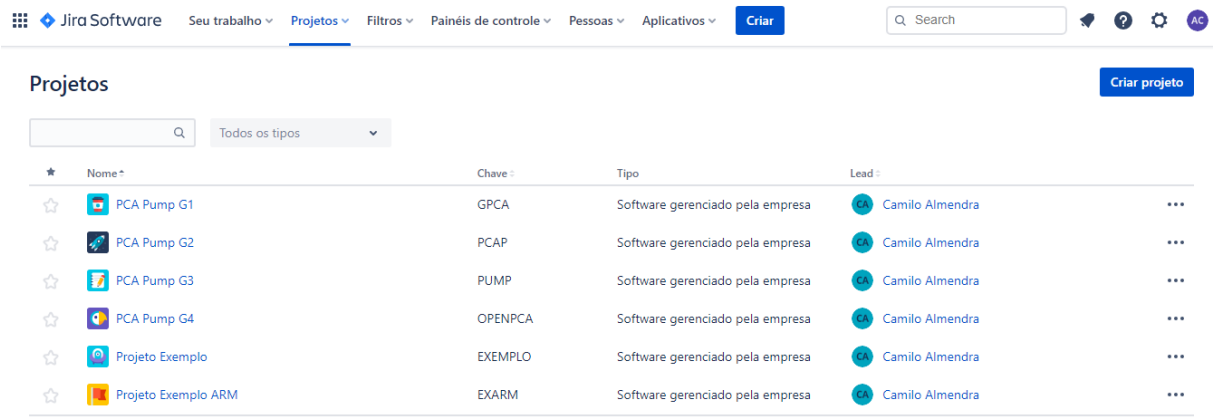

Figura 11. Quadro de Projetos no Jira

Para o exemplo do Extrator, será selecionado o projeto personalizado PCA Pump G1, onde serão mostrados os itens cadastrados:

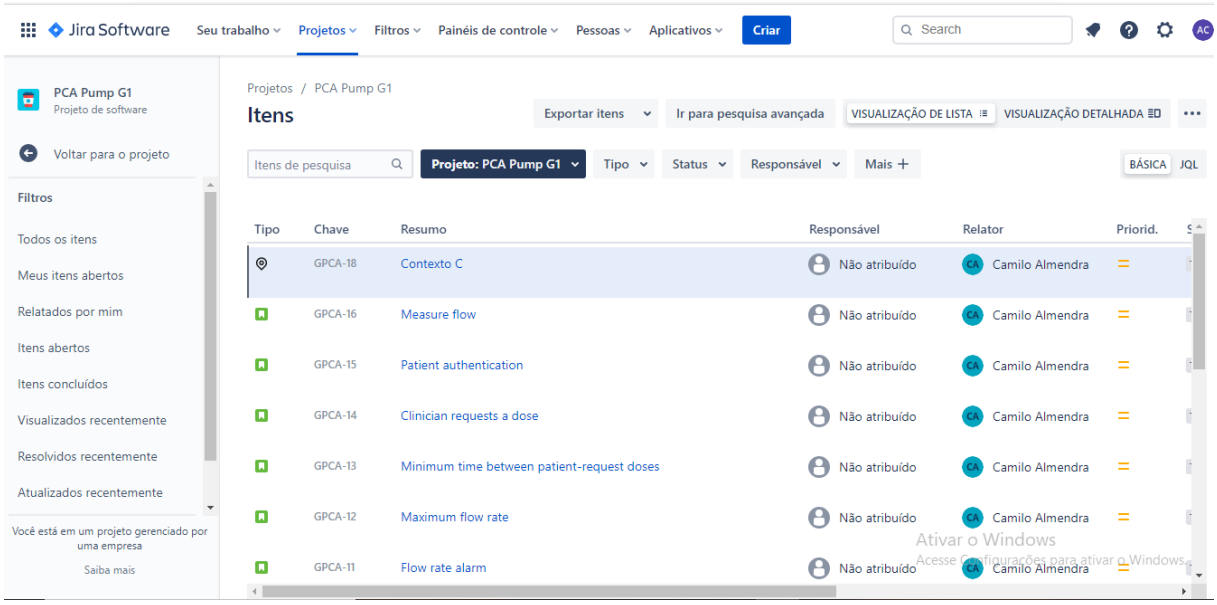

Figura 12. Itens do Projeto PCA Pump G1

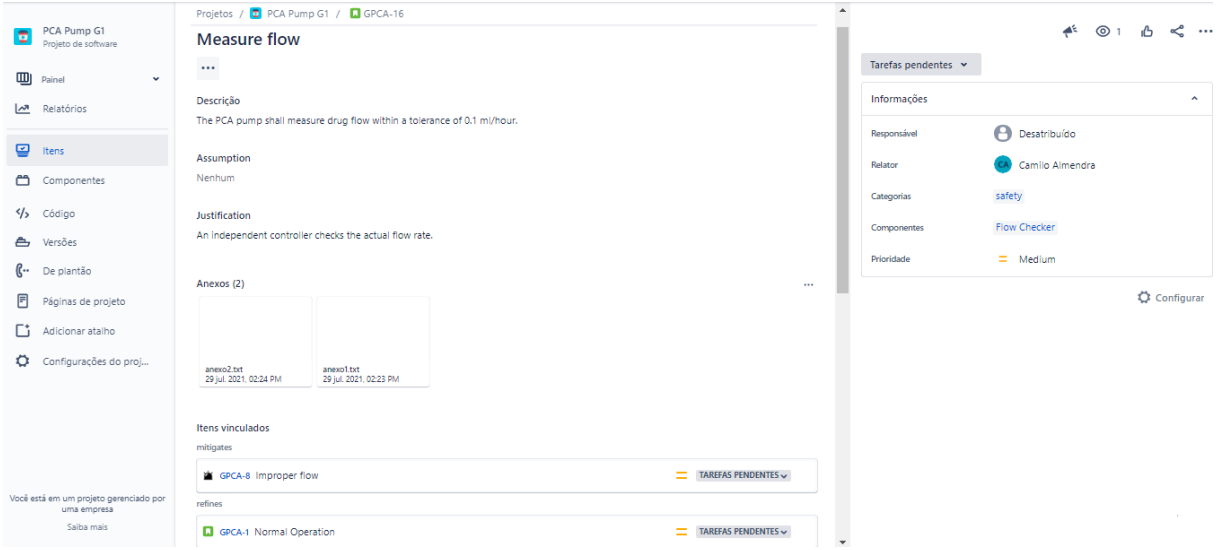

Figura 13. Exemplo Item com personalização

Para realizar a exportação, logo na tela de entrada do projeto, onde são exibidos os itens, é apresentada a opção de Exportar Itens, para o extrator, a exportação tem que ser no formato CSV.

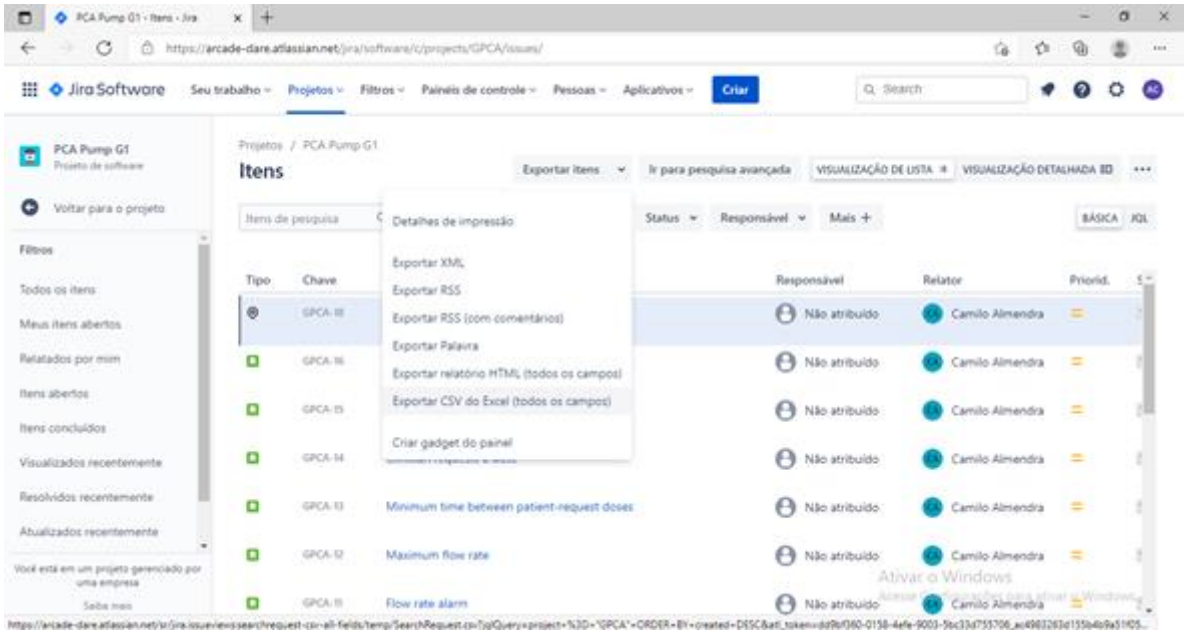

Figura 14. Exportação dos Itens

3.2.2. PREPARANDO PARA A EXECUÇÃO

Para o Extrator estar pronto para ser executado, será preciso ter a pasta ‘Assets’ no mesmo local onde o jar do programa está localizado. Nessa pasta encontra-se o arquivo **baseFileRDF.txt**.

O arquivo **baseFileRDF** consta a configuração padrão que será incluída na geração do arquivo de saída do extrator. Essa configuração padrão são as propriedades e classes do metamodelo ARM no formato de n-triplas de RDF.

3.2.3. EXECUTANDO O PROGRAMA

Com o arquivo de entrada (exportação dos itens do Jira em formato CSV) e a pasta Assets localizada no mesmo local do executável **ExtratorARM_RDF.jar**, o programa está pronto para ser executado.

Foi definido que a execução do programa seria feito através de linha de comando via terminal CMD para simplificar a utilização da ferramenta. Na execução do programa, deve localizar o local onde os arquivos já citados se encontram e deve dar o comando de executar o extrator e informar o local onde o arquivo de entrada se encontra no formato abaixo:

Java -jar ExtratorARM_RDF.jar -i local do arquivo de entrada

Na demonstração, o executável a pasta assets estão localizados em D:\TG Program.

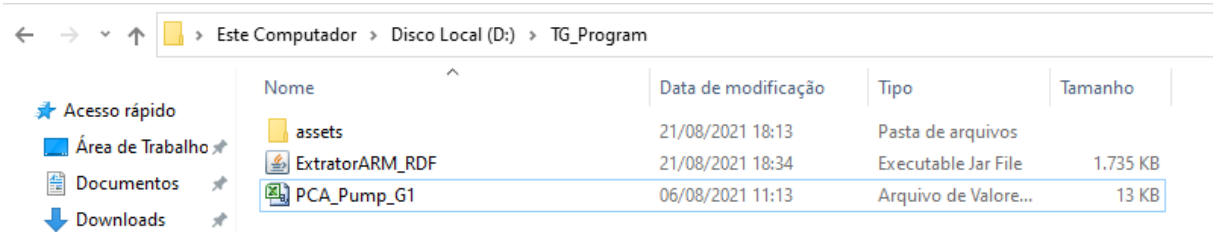

Figura 15. Configuração ExtratorARM

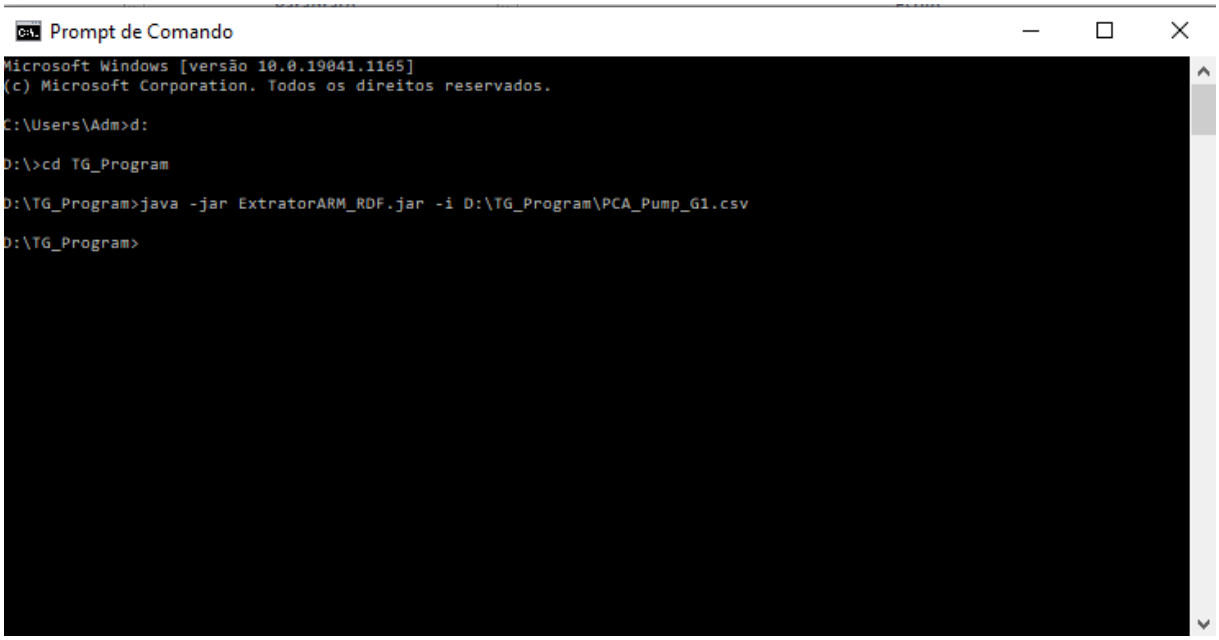

Figura 16. Execução por linha de comando

Com a execução do programa via CMD, é gerado um arquivo no mesmo local onde o arquivo de entrada está localizado com o nome **outputARM_RDF**. Esse arquivo consta com todos os elementos ARM em formato de n-triplas RDF.

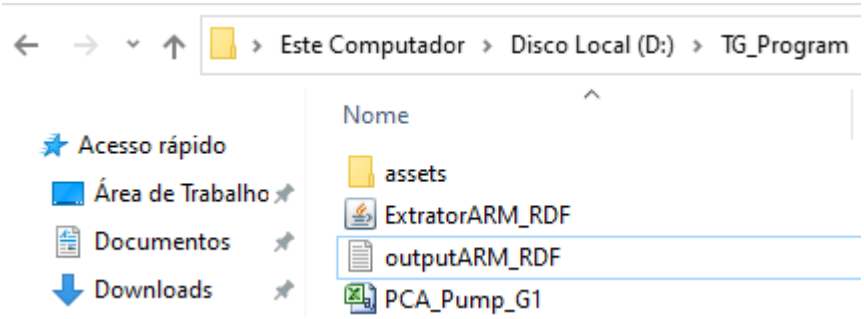

Figura 17. Arquivo de saída gerado

Abaixo é exibido trecho do arquivo de saída de acordo com os conceitos do ARM no formato de n-triplas RDF.

```
###  urn:absolute:arcade/aro#GPCA-18
:GPCA-18 rdf:type owl:NamedIndividual .
:GPCA-18 rdf:type :Context .

###  urn:absolute:arcade/aro#GPCA-16
:GPCA-16 rdf:type owl:NamedIndividual .
:GPCA-16 rdf:type :SafetyRequirement .
:Flow_Checker :realizes :GPCA-16 .
:GPCA-16 :mitigates :GPCA-8 .
:GPCA-16 :refines :GPCA-1 .
:anexo1 rdf:type owl:NamedIndividual .
:anexo1 :documents :GPCA-16 .
:anexo2 rdf:type owl:NamedIndividual .
:anexo2 :documents :GPCA-16 .
:GPCA-16-J1 rdf:type owl:NamedIndividual .
:GPCA-16-J1 rdf:type :Justification .
:GPCA-16-J1 :explains :GPCA-16 .
```

Figura 18. Exemplo de n-tripla arquivo de saída

As triplas geradas podem posteriormente podem alimentar uma ontologia e serem visualizadas através de ferramentas linked data[2].

[2] https://www.w3.org/2001/sw/wiki/LLDtools#RDF_Visualization

4. CONSIDERAÇÕES FINAIS

Esta monografia teve o objetivo de produzir uma ferramenta que realiza mapeamento dos itens de um projeto cadastrado num ITS personalizado com o metamodelo ARM em um formato que possa ser consumido por uma ontologia OWL. A ferramenta foi construída em Java. O ITS utilizado foi o Jira, e os dados extraídos de arquivos CSV exportados a partir da interface gráfica do Jira. O extrator implementou internamente heurísticas para identificar a personalização descrita na Tabela 1, a fim de identificar os conceitos e relacionamentos presentes no metamodelo ARM. O formato de saída gerado foi RDF N-triplas, que é largamente utilizado para representação da informação e no adotado em ferramentas de gerenciamento do conhecimento. Essas triplas estão prontas para serem utilizadas como indivíduos de entrada para processamento de ontologias (ver Figura 1). O passo a passo da utilização da ferramenta que serve como um guia do usuário, e está descrito na seção 3.3.

4.1. LIMITAÇÕES E TRABALHOS FUTUROS

A dificuldade encontrada no desenvolvimento deste trabalho foi a questão de implementação do software, algumas ideias tiveram que serem deixadas de lado pelo prazo curto para a construção da ferramenta.

Como projeção para trabalhos futuros da ferramenta, ficam apontados:

1. Preparar o extrator para receber arquivos extraídos de outros ITS como GitLab e Azure DevOps.

2. Viabilizar a extração das informações do Jira através da própria API[3].

[3] https://developer.atlassian.com/cloud/jira/software/rest/intro/#introduction

REFERÊNCIAS

AELA.IO. (n. d). **Tudo Sobre Agile: A Filosofia que Fará Você Muito mais Eficiente**. Acesso em 17 de ago. de 2021. Disponível em https://medium.com/aela/tudo-sobre-agile-a-filosofia-que-far%C3%A1-voc%C3%AA-muito-mais-eficiente-ac1a5721fe20.

ATLASSIAN. (n.d). **Jira issue tracking system**. Acesso em 19 de ago de 2021. Disponível em https://www.atlassian.com/software/jira.

ADELARD. (n.d). **Goal Structuring Notation (GSN)**. Acesso em 18 de ago. de 2021. Disponível em https://www.adelard.com/gsn.html.

ALMENDRA, Camilo; SILVA, Carla. (2020)**. Managing Assurance Information: A Solution Based on Issue Tracking Systems**. SBES '20, October 21–23, 2020, Natal, Brazil.

ALMENDRA, Camilo; SILVA, Carla; BARROS, Flavia (2019)**. Using Assurance Cases in Requirements Engineering for Safety-Critical Systems**.

ALMENDRA, Camilo; SILVA, Carla; VILELA, Jéssyka**.** (2020). **Incremental Development of Safety Cases: a Mapping Study**. SBES '20, October 21–23, 2020, Natal, Brazil.

R. BLOOMFIELD e P. BISHOP. (2010). **Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective. In Making Systems Safer**, Chris Dale and Tom Anderson (Eds.). Springer, London, 51–67

COMPUTER WORD (n.d). **Estas são as 10 linguagens de programação mais utilizadas em 2019**. Acesso em 19 de ago de 2021. Disponível em https://computerworld.com.br/carreira/estas-sao-as-10-linguagens-de-programacao-mais-utilizadas-em-2019/

DEVMEDIA. (n.d). **Introdução ao Desenvolvimento Ágil**. Acesso em 17 de ago. de 2021. Disponível em https://www.devmedia.com.br/introducao-ao-desenvolvimento-agil/5916.

GASPARINI. Vinicius; SCHROEDER, Rebeca. (2018). FragRDF: **Um Fragmentador de dados RDF baseado em Esquemas**. Anais do Computer on the Beach, pp.248-257.

X. GE, R. PAIGE, and J. MCDERMID. (2010). **An Iterative Approach for Development of Safety-Critical Software and Safety Arguments**. In Agile Conf. 35–43

J. HATCLIFF, A. WASSYNG, T. KELLY, C. COMAR e P. JONES. (2014). **Certifiably Safe Software-dependent Systems: Challenges and Directions. In Future of Software Engineering Proceedings**. ACM, Índia, 182–200.

R. HAWKINS, I. HABLI, T. KELLY e J. MCDERMID. (2013). **Assurance cases and prescriptive software safety certification: A comparative study**. Safety Science 59, 55–71.

L. HEEAGER e P. NIELSEN. (2018). **A conceptual model of agile software development in a safety-critical context: A systematic literature review**. Information and Software Technology 103, julho (2018), 22–39.

J. HATCLIFF, B. LARSON, T. CARPENTER, P. JONES, Y. Zhang, J. JORGENS. (2018) **The Open PCA Bomb Project: An Exemplary Open Source Medical Device as a Community Resource**. In: Proceedings of the 2018 Medical Cyber-Physical Systems (MedCPS) Workshop. [Sl: sn].

T. KELLY. (2018). **Safety Cases**. John Wiley & Sons, Ltd, Chapter 16, 361–385.

MILLER, E. **An Introduction to the Resource Description Framework**. D-Lib Magazine, v. 4, n. 5, May, 1998. Disponível em: https://www.dlib.org/dlib/may98/miller/05miller.html. Acesso em: 19 ago. 2021.

B. RAMESH e M. JARKE. (2001). **Toward reference models for requirements traceability**. IEEE Trans. on Software Eng. 27, 1 (2001), 58–93.

TECH | 2021. (2021). **Caso de garantia: um caso de segurança fundamentado**. Acesso em 18 de ago. de 2021. Disponível em https://tech-pt.netlify.app/articles/pt525914/index.html.

J. VILELA, J. CASTRO, L. MARTINS e T. GORSCHEK. (2018). **Safe-RE: um requisito de segurança-mentos Metamodelo com base nos padrões de segurança da indústria**. Em Proc. da 32ª brasileiraSymp. no Softw. Eng. 196–201.

W3C**.** (2014). **RDF 1.1 N-Triples**. Acesso em 16 de ago. de 2021. Disponível em https://www.w3.org/TR/n-triples/.