
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO JOSÉ DE VASCONCELOS MATOS

# **DRL TC Classifier** : Classificação de casos de teste para automação de testes de dispositivos móveis

Recife

2020

EDUARDO JOSÉ DE VASCONCELOS MATOS

# **DRL TC Classifier** : Classificação de casos de teste para automação de testes de dispositivos móveis

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

**Área de Concentração**: Inteligência Computacional

**Orientadora**: Flávia de Almeida Barros

Recife

2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M433d Matos, Eduardo José de Vasconcelos
DRL TC classifier: classificação de casos de teste para automação de testes de dispositivos móveis / Eduardo José de Vasconcelos Matos. – 2020.
130 f.: il., fig., tab.

Orientadora: Flávia de Almeida Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Engenharia do conhecimento. I. Barros, Flávia de Almeida (orientadora). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 203

**Eduardo José de Vasconcelos Matos**

**“"DRL TC Classifier: Classificação de Casos de Teste para Automação de Testes de Dispositivos Móveis”**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 05/10/2020.

**BANCA EXAMINADORA**

---

Prof. Dr. Alexandre Cabral Mota
Centro de Informática/ UFPE

---

Prof. Dr. Lucas Albertins de Lima
Departamento de Computação / UFRPE

---

Profa. Dra. Flavia de Almeida Barros
Centro de Informática / UFPE
**(Orientadora)**

## AGRADECIMENTOS

Gostaria de agradecer inicialmente a minha esposa e companheira, Lorena, por todo amor, apoio, incentivo e carinho, além de toda compreensão a mim demonstrada nos momentos em que precisei ficar isolado para concluir esse trabalho. Obrigado por ser quem você é na minha vida. Te amo muito.

À minha orientadora, professora Dra. Flávia de Almeida Barros, por toda sua atenção, paciência, troca de experiências e orientações, que me foram extremamente úteis para a conclusão deste trabalho. Seu papel foi imprescindível para meu desenvolvimento acadêmico e deste trabalho. Muito obrigado por tudo, professora.

Agradeço aos meus amigos de trabalho, ao time de Upgrade Validation (em especial aos times Upgrade Validation Automation e de Pesquisa) e ao Projeto CIn/Motorola (um convênio entre o Centro de Informática da UFPE e a Motorola Mobility), que me proporcionou o ambiente e todos os meios necessários para que este trabalho fosse realizado. Deixo aqui registrado o meu muito obrigado!

## RESUMO

Buscando garantir a qualidade de produtos de software, são realizadas diversas atividades de teste. Contudo, esta é uma tarefa cara e complexa, responsável por aproximadamente 50% do custo de desenvolvimento de um software. A automação de testes surge como uma solução para diminuir o custo e aumentar a eficiência dos processos de teste. Um dos primeiros estágios do processo de automatização de Casos de Testes (CTs) é a identificação da viabilidade dessa automação. Atualmente, essa é uma problemática real da nossa empresa parceira, no ramo de dispositivos móveis, que mantém uma equipe responsável por criar os *scripts* executáveis para os CTs automatizáveis. A classificação é realizada manualmente por especialistas com conhecimento sobre as funcionalidades dos dispositivos (i.e., que funcionalidades podem ou não ser testadas automaticamente), e sobre o ambiente de desenvolvimento de CTs automáticos. No entanto, especialistas são profissionais "caros", cujo tempo livre é escasso e o conhecimento adquirido é muito valioso. Nesse contexto, este trabalho teve por objetivo auxiliar no processo de automação de teste, criando um sistema para classificação automática de CTs entre automatizáveis ou não. Além de otimizar essa tarefa, este trabalho também contribuiu para registrar o conhecimento do especialista, que pode ser perdido se o colaborador sair da empresa. O protótipo implementado utiliza conceitos da Classificação de Texto e da Engenharia do Conhecimento baseada em regras e inferência (através da ferramenta Drools). A avaliação do protótipo apresentou resultados muito satisfatórios de valores de acurácia, precisão, sensibilidade e F1 Score, de 84.1%, 80.2%, 90.6% e 85.08%, respectivamente. Além disso, observamos que a classificação realizada de um conjunto de teste com 662 CTs levou apenas alguns minutos, enquanto que a mesma tarefa levaria dias ou semanas para ser realizada manualmente.



**Palavras-chaves**: Classificação de texto. Engenharia do conhecimento. Regras. Engenharia de Testes.

## ABSTRACT

In order to guarantee the quality of software products, several testing activities are performed. However, this is an expensive and complex task, responsible for approximately 50% of the cost of softwares development. Test automation emerges as a solution to lower the cost and increase the efficiency of the test processes. One of the first stages of the Test Case (TCs) automation process is the identification of the feasibility of this automation. Currently, this is a real problem for our partner company in the field of mobile devices, which maintains a team responsible for creating the executable scripts for the automated TCs. The classification is performed manually by specialists with knowledge about the features of the devices (i.e., which features may or may not be tested automatically), and about the automatic TC development environment. However, specialists are "expensive" professionals, whose spare time is scarce and the knowledge acquired is very valuable. In this context, this work aimed to assist in the test automation process, creating a system for automatic classification of TCs, whether automatable or not. In addition to optimizing this task, this work also contributed to recording the specialist's knowledge, which can be lost if the employee leaves the company. The implemented prototype uses concepts of Text Classification and Knowledge Engineering based on rules and inference (through the Drools tool). The evaluation of the prototype showed very satisfactory results for values of accuracy, precision, sensitivity and F1-score, with 84.1%, 80.2%, 90.6% e 85.08%, respectively. In addition, we observed that the classification of a set with 662 TCs was performed in just a few minutes, whereas the same task would take days or weeks to be performed manually.



**Keywords**: Text classification. Knowledge engineering. Rules. Test Engineering.

## LISTA DE FIGURAS

# LISTA DE TABELAS

## SUMÁRIO

# 1 INTRODUÇÃO

Buscando garantir qualidade e confiabilidade, empresas de software realizam diversas atividades de teste durante todo o processo de desenvolvimento de seus produtos. O objetivo principal é identificar erros e defeitos antes do lançamento dos produtos, bem como acompanhar o software em uso pelos clientes (LI; ZHANG; KOU, 2010). Contudo, esta é uma tarefa cara e complexa, responsável por aproximadamente 50% do custo de desenvolvimento dos sistemas de software (BEIZER, 1990),(PARGAS; HARROLD; PECK, 1999).

Neste cenário, a automação de testes surge como uma solução para aumentar a eficiência dos processos de teste. De fato, testes automáticos podem reduzir significativamente o custo e o tempo de desenvolvimento e manutenção de software como um todo (FEWSTER; GRAHAM, 1999). Contudo, dependendo do tipo de produto que está sendo testado, nem todos os Casos de Teste (CTs) podem ser automatizados. Assim, um dos primeiros estágios do processo de automatização de CTs é a identificação da viabilidade dessa automatização.

Atualmente, essa é uma problemática real da empresa parceira, no ramo de dispositivos móveis, junto à qual este trabalho foi desenvolvido. A empresa mantém uma equipe responsável por criar scripts executáveis para os CTs considerados automatizáveis por funcionários, que verificam manualmente cada passo de cada CT proposto. Este é um trabalho demorado e que depende de profissionais com conhecimento especializado na área. Esses profissionais precisam ter conhecimento sobre as funcionalidades dos dispositivos (i.e., que funcionalidades podem ou não ser testadas automaticamente), e sobre o ambiente de desenvolvimento de CTs automáticos.

No entanto, especialistas são profissionais caros, cujo tempo livre é escasso e o conhecimento adquirido é muito valioso. E com o desenvolvimento de novos produtos, novas funcionalidades ou a simples atualização das funcionalidades já existentes, é necessário criar novos CTs ou atualizar os casos já automatizados. Assim, o trabalho de análise e classificação de CTs é contínuo e custoso. Nesse contexto, este trabalho teve por objetivo auxiliar no processo de automação de teste, criando um sistema para classificação automática de CTs entre automatizáveis ou não (DRL TC Classifier).

Como já mencionado acima, este trabalho foi desenvolvido no contexto de uma parceria entre o CIn-UFPE e uma empresa de teste de software para dispositivos móveis. As aplicações para dispositivos móveis (por exemplo, aplicativos executados em *smartphones* ou *tablets*

de nova geração) se tornaram na última década tão populares que representam uma nova revolução no setor da Tecnologia da Informação. É relevante mencionar, além disso, que elas movimentam alguns bilhões de dólares por ano no comércio mundial (MUCCINI; FRANCESCO; ESPOSITO, 2012).

Absolutamente, um teste automatizado pode reduzir significativamente o custo e o tempo de desenvolvimento de software. Entretanto, é imprescindível otimizar a atividade de teste e de sua automação quando possível. Um dos primeiros estágios do processo de automatizar casos de teste existente é a identificação da viabilidade dessa automação. Atualmente, essa é uma problemática real de uma empresa que classifica manualmente os seus casos de teste entre automatizáveis ou não.

## 1.1 TRABALHO REALIZADO

Este trabalho de mestrado, na área de classificação automática de texto, teve por objetivo principal desenvolver um sistema para classificar casos de teste em formato textual entre manuais ou automatizáveis, dentro de um framework institucional. Trata-se de um Sistema Especialista (SE) (DUDA; SHORTLIFFE, 1983), uma vez que ele se propõe a substituir o trabalho de um profissional especialista nesse domínio. Vale salientar que SEs são geralmente construídos utilizando-se técnicas da área de Inteligência Artificial (IA) (RUSSELL; NORVIG, 2013).

Inicialmente, foram estudados métodos e técnicas para a construção de sistemas de classificação de texto, com foco nas abordagens tradicionais da Inteligência Artificial: Engenharia do Conhecimento (EC) (DUDA, 1981), e a Aprendizagem de Máquina (AM) (MICHIE et al., 1994). Além da EC utilizando regras e inferência, investigamos alguns métodos de aprendizagem de máquina (K-NN, Naïve Bayes e Support Vector Machines), a fim de identificar qual seria o método mais adequado para a construção do nosso classificador.

Como resultado desse estudo inicial, decidimos desenvolver nosso classificador dentro da abordagem da EC, com inferência sobre regras de produção. A escolha da abordagem se deu pela natureza dos documentos, os casos de teste são bastantes similares, e pelo conjunto de documentos ser muito pequeno para utilização de técnicas de aprendizagem de máquina. Os benefícios da utilização da EC serão mostrado nas próximas seções e também influenciaram na escolha da técnica.

Por ser uma técnica já bem consolidada, a Engenharia do Conhecimento se mostrou bas-

tante adequada ao nosso caso, possibilitando a criação de um ambiente onde o conhecimento pode ser facilmente armazenado e compartilhado. Aqui, ressaltamos que é de grande importância para as empresas a criação de bases de conhecimento especialista, capazes de preservar o conhecimento (a experiência) de profissionais caros, que podem se afastar da empresa a qualquer momento.

Depois de um estudo mais aprofundado sobre EC baseada em regras, desenvolvemos um sistema especialista para classificação de CTs: DRL TC Classifier. Para isto, foi utilizado um framework de processamento de regras, o Drools (HAT, 2020a). A base de regras foi construída a partir de entrevistas com especialistas na tarefa de classificação de CTs na empresa parceira. Assim sendo, trata-se de um repositório valioso, que será usado não só na tarefa de classificação, como também como registro desse conhecimento valioso, que pode ser perdido se o colaborador sair da empresa.

A avaliação do protótipo implementado apresentou resultados muito satisfatórios, com valores de acurácia, precisão, sensibilidade e F1 Score, de 84.1%, 80.2%, 90.6% e 85.08%, respectivamente. Além disso, observamos que a classificação automática de um plano de teste com mais de 662 CTs foi realizada em apenas poucos minutos, enquanto que a mesma tarefa levaria dias ou semanas para ser realizada manualmente.

## 1.2 ESTRUTURA DO DOCUMENTO

O presente trabalho está dividido em mais 5 capítulos, descritos a seguir.

- No Capítulo 2 apresenta-se um estudo sobre a classificação de documentos. Serão apresentados alguns conceitos básicos e problemas centrais que podem ser encontrados quando se precisa utilizar uma solução com classificação textual. Além disso, serão mostradas as abordagens mais clássicas e populares na literatura, bem como os trabalhos relacionados;

- No Capítulo 3 segue-se a apresentação dos conceitos de "conhecimento" e de "engenharia do conhecimento", e como essa engenharia define, organiza, modela, utiliza ferramentas e cria processos para solucionar problemas com o conhecimento;

- No Capítulo 4 é mostrado um problema real de uma empresa parceira do CIn-UFPE na área de dispositivos móveis. Em seguida, será proposto um sistema para auxiliar no

processo de classificação de casos de teste, em automatizáveis ou não, com base em conhecimento de especialistas em forma de regras explícitas, e usando uma máquina de inferência;

- No Capítulo 5 é detalhada a validação do sistema DRL TC Classifier, com uma análise dos resultados obtidos;
- No Capítulo 6, por fim, é realizada a conclusão desse trabalho e são apresentados possíveis trabalhos futuros.

Além desses capítulos, o documento traz ainda 2 apêndices com informações complementares que foram julgadas relevantes para a compreensão do trabalho aqui relatado.

# 2 CLASSIFICAÇÃO DE TEXTO

O objetivo da classificação de texto ou documentos é determinar categorias em que eles se encaixam de acordo com seu conteúdo. É uma das tarefas fundamentais do Processamento de Linguagem Natural (PNL) (MANNING; MANNING; SCHÜTZE, 1999) com vasta possibilidade de aplicação, como análise de sentimentos, rotulagem de tópicos, detecção de spam e detecção de intenção, entre muitas outras. Estes últimos serão explicados nas seções subsequentes.

A importância dessa tarefa se deve também ao fato de que os dados não estruturados em forma de texto estão em toda parte: *e-mails*, *chats*, páginas *web*, mídias sociais, chamados de suporte, respostas a pesquisas, etc. O texto pode ser uma fonte extremamente rica de informações. Contudo, a tarefa de extração de informação pode ser difícil e demorada devido à natureza não estruturada dos textos livres. Por este motivo, muitas empresas estão recorrendo à classificação de texto para oferecer alguma informação sobre o texto de maneira rápida e econômica, a fim de aprimorar e automatizar processos dessas organizações como, por exemplo, a tomada de decisões.

No caso da empresa em que este trabalho está sendo aplicado, uma empresa de teste de software para aplicativos móveis, é possível listar algumas utilidades da classificação de texto. Pode-se citar, por exemplo, a classificação de: falhas reportadas, logs, pedidos de mudança (*change requests* - CR), atividades a serem realizadas em extensos backlogs, casos de teste utilizados, quais desses casos de teste podem ou não ser automatizados, entre outros. E, este último, como já dito, será o tema centra deste trabalho.

Neste capítulo, então, será mais aprofundada a definição da classificação de texto e alguns dos pontos relevantes no processo de classificação de texto. Entre estes pontos, estão a necessidade de representar textos não-estruturados e de reduzir a dimensionalidade dos textos. Além disso, serão vistas algumas abordagens para classificar documentos: a engenharia do conhecimento e regras, Naïve Bayes, Support Vector Machines (SVM) e K-Nearest Neighbors (K-NN). Por fim, serão realizadas as considerações finais acerca da classificação de texto e como ela será utilizada neste trabalho.

## 2.1 DEFINIÇÃO DO PROBLEMA

Como já visto, o objetivo da categorização de texto ou de documento textual é determinar categorias (ou rótulos) predefinidas com base em suas informações ou características (JOHNSON et al., 2002; SEBASTIANI, 2002; IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005). De acordo com os tipos de tarefas no CommonKADS, adaptado de Harmelen, Lifschitz e Porter (2008), pode-se detalhar o problema de classificação da seguinte forma:

- **Tipo de tarefa**: Classificação de texto ou documento
- **Entrada**: O documento ou as características do documento
- **Saída**: Classe(s) do documento
- **Conhecimento**: Associação entre a(s) classe(s) e as características do documento
- **Característica da tarefa**: O conjunto de classes é predefinido

Korde e Mahender (2012) complementam a definição de classificação de texto como sendo uma tarefa de grande importância, que pode ser realizada utilizando técnicas de **engenharia do conhecimento** com regras, conceito que será aprofundado no capítulo 3. Ou seja, define-se de forma manual um conjunto de regras que representam o conhecimento de especialistas em como classificar documentos a partir de determinado conjunto de categorias. E, após isto, é possível atribuir a classe correta (categoria) a um novo documento do domínio.

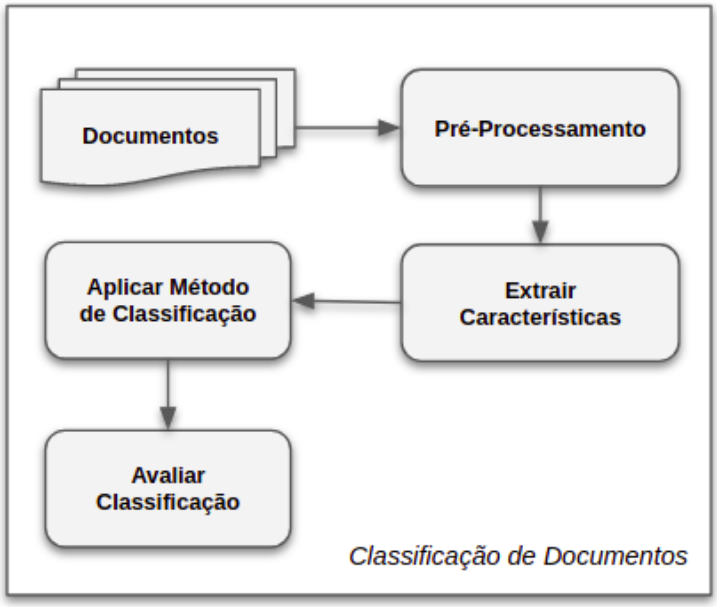
Mais formalmente, pode-se explicar a classificação de documentos como: se $d_i$ é um documento do conjunto de documentos D e $\{ c_1, c_2, ..., c_n \}$ é o conjunto de todas as categorias de interesse para D, então a classificação de texto atribui uma categoria $c_j$ a um documento $d_i$. O problema principal de classificação pode ser compreendido como sendo *a atribuição de uma ou mais classes a documentos de texto*.

Em relação à quantidade de rótulos associados (também chamados na literatura de etiquetas) durante uma classificação, Korde e Mahender (2012) mostram a existência de dois grandes grupos. Nos casos em que um documento é associado apenas a uma categoria, trata-se de um problema de classificação ***single-label*** (tal como, por exemplo, uma mensagem de e-mail ser classificada como spam ou não-spam). Também existem os casos em que um documento pode ser classificado em mais de uma categoria, sendo então um problema ***multi-label*** (como, por exemplo, o diagnóstico de um paciente ser classificado em mais de um tipo de doença).

Já Dalal e Zaveri (2011), Sebastiani (2002) indicam a possibilidade de classificação automática de texto, usando uma abordagem de aprendizagem de máquina (MICHIE et al., 1994). Nessa abordagem, assim como em outras abordagens de classificação textual, é feita com base em palavras ou características significativas extraídas do documento de texto. Como as classes são predefinidas, é uma tarefa supervisionada de aprendizagem de máquina.

Dado o que já foi visto, é possível afirmar que classificadores de texto podem ser criados usando técnicas baseadas em **engenharia do conhecimento**, **regras** e/ou **aprendizagem de máquina**. Ainda, ilustram-se os conceitos vistos até aqui na Figura 1, onde é apresentado um desenho de processo genérico para classificação textual.

Figura 1 – Estratégia genérica para classificação de texto




**Fonte:** DHAR; DASH; ROY (2019, tradução livre)

Este processo recebe como entrada os documentos a serem classificados, que passam pela etapa de pré-processamento, onde é realizada, por exemplo, a tokenização, remoção de ruídos e dados inconsistentes. Em seguida, os documentos pré-processados seguem para a etapa de extração de características, onde são extraídas as características mais relevantes para a classificação. Após a extração de características, realiza-se aplicação do método de classificação dos documentos. E, por fim, avalia-se a classificação realizada.

A partir dos conceitos apresentados, a seguir, serão abordadas algumas das etapas da estratégia genérica para classificação de texto. Essas serão: como representar os documentos no pré-processamento e reduzir sua dimensionalidade com a seleção de características para classificação.

## 2.2 REPRESENTAÇÃO DOS DOCUMENTOS

De forma geral, pode-se considerar um documento como uma sequência de palavras, frases, sentenças ou parágrafos (LEOPOLD; KINDERMANN, 2002). E cada documento geralmente é representado por uma lista de termos mais relevantes. O conjunto de todos os termos que ocorrem nos documentos do corpus (ou base de documentos) a ser considerado é chamado de vocabulário ou conjunto de características.

Uma problemática registrada por Dalal e Zaveri (2011) para a classificação de texto é determinar a metodologia de trabalho dependendo da estrutura dos seus artefatos sob análise. Alguns tipos de documentos de texto, como trabalhos de pesquisa científica, geralmente são escritos estritamente em um formato pré-especificado. Isto facilita a classificação deles, pois pode-se basear na posição das características no texto, informações posicionais.

No entanto, a maioria dos documentos de texto é escrita de maneira não estruturada, portanto, a classificação deve ser feita com base em características como presença ou ausência de palavras e sua frequência de ocorrência. Algumas abordagens de classificação representam os documentos como vetores de pesos binários ou não (por exemplo, utilizando a frequência de ocorrência dos termos do conjunto de documentos).

## 2.3 PRÉ-PROCESSAMENTO

As **características** úteis na classificação de texto podem ser palavras simples do vocabulário do idioma; palavras-chaves especificadas pelo usuário ou extraídas do próprio texto; e termos compostos (*multi-words*) (ZHANG; YOSHIDA; TANG, 2007). Na literatura de classificação de textos, outras fases envolvidas na etapa de pré-processamento podem ser: remoção de *stop-words* (KIM et al., 2006), (ZHANG; YOSHIDA; TANG, 2007), (HAO; HAO, 2008), *stemming* (PORTER et al., 1980), uso de tesauros, etc.

Segundo (ZHANG; YOSHIDA; TANG, 2007) e (CHURCH; HANKS, 1990), uma *multi-word* é uma sequência ou coocorrência de palavras com significado semântico (por exemplo, "Tecnologia da Informação", "Universidade Federal de Pernambuco" ou "Centro de Informática"). Por isto, as *multi-words* são úteis na classificação e na eliminação de ambiguidade. Para extrair os termos compostos dos documentos, muitos métodos de extração, que empregam principalmente métodos estatísticos baseados em informações mútuas e métodos linguísticos baseados em propriedades sintáticas, são propostos em (CHURCH; HANKS, 1990; PARK; BYRD;

BOGURAEV, 2002).

Por sua vez, na fase de remoção de *stop-words*, são removidas dos textos palavras que têm sua classe gramatical dentre artigo, preposições, pronomes, conjunções, interjeições, por exemplo: *de*, *para*, *ela*, *a*, *um*, etc.

Já na operação de *stemming*, aplica-se a redução da palavra ao seu ***stem***. Assim, palavras como, por exemplo, ***cert****-o*, ***cert****-eza* e *in-****cert****-eza* são representadas pelo seu radical ***cert***. É interessante observar que nesta operação, removem-se prefixos e sufixos da palavra.

Um tesauro, muitas vezes chamado de dicionário de sinônimos, pode ser descrito como uma base de conhecimento que representa o modelo conceitual de determinado domínio (Larsen; Yager, 1993). Para cada entrada, o tesauro pode ter além de sinônimos, antônimos, relação de tipo/generalização (classe superior), relação de partes, classe gramatical, etc. Além disto, o tesauro pode ser de domínio geral ou específico, possibilitando um vocabulário controlado para cada contexto. Ele pode ser utilizando também para restringir ou expandir os termos de um texto.

Com essas técnicas mencionadas acima, entre outras, é possível flexibilizar a quantidade de palavras distintas e relevantes em um documento, ou seja, sua dimensionalidade.

## 2.4 SELEÇÃO DE CARACTERÍSTICAS

É sabido que documentos de texto geralmente usam palavras de um vocabulário amplo, mas nem todas as palavras que ocorrem em um documento são úteis para classificação.

O objetivo dos métodos de seleção de características, assim como de algumas fases do pré-processamento já citadas, é a redução da dimensionalidade do conjunto de características, removendo os recursos que são considerados irrelevantes para a classificação. Ikonomakis, Kotsiantis e Tampakas (2005) mostram que esse procedimento de transformação apresenta uma série de vantagens, incluindo tamanho menor do conjunto de características, requisitos computacionais menores para os algoritmos de categorização de texto (especialmente aqueles que não se adaptam bem a grandes conjunto de características). O objetivo é a redução da dimensionalidade das características para gerar uma precisão de classificação aprimorada.

Assim, pesquisadores propõem técnicas de ranqueamento de termos ou características como, por exemplo, o *Term-Frequency Inverse Document-Frequency*, TF-IDF (JONES, 1972; ZHANG; YOSHIDA; TANG, 2008; JONES, 2004), frequência do termo no documento (*documento frequency*), frequência do termo na base de documentos (*term frequency*), informações mútuas

(*mutual information*), ganho de informações (*information gain*), estatística $\chi^2$ ($\chi^2$ *statistics*) e força do termo (*term strength*) (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

Uma das mais populares e relevantes técnicas, TF-IDF (JONES, 1972), é puramente estatística e é usada para avaliar a importância de uma palavra com base em sua frequência de ocorrência no documento e no corpus de documentos relevantes. Essa técnica calcula a frequência normalizada do termo no documento (*Term Frequency*) com o inverso da frequência do termo nos documentos da base (*Inverse Document Frequency*). A equação 2.1 é a fórmula clássica de TF-IDF usada para ponderação de um termo.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{2.1}$$

onde, $w_{i,j}$ é o peso do *i*-ésimo termo no *j*-ésimo documento, *N* é o número de documentos na coleção, $tf_{i,j}$, é a frequência do *i*-ésimo termo no *j*-ésimo documento e $df_i$ é o número de documentos contendo o termo *i*.

## 2.5 ABORDAGENS E TÉCNICAS PARA CONSTRUÇÃO DE CLASSIFICADORES TEXTUAIS

Nesta seção, serão apresentadas duas abordagens para classificação de documentos: a engenharia do conhecimento e a aprendizagem de máquina. É possível citar entre as técnicas mais populares dessas abordagens: regras do tipo SE-ENTÃO (*IF-THEN*), árvore de decisão, redes neurais, entre outras técnicas. Sendo a primeira técnica da abordagem engenharia do conhecimento e as demais da abordagem aprendizagem de máquina.

Neste trabalho, a estratégia de construção manual de classificadores sugerida na engenharia do conhecimento com a utilização de regras que será vista a seguir na Seção 2.5.1. Já para as estratégias de aprendizagem de máquina, na Seção 2.5.2, serão contemplados três algoritmos comumente utilizados.

### 2.5.1 Engenharia do conhecimento e regras

No mundo atual com a informação em rápida e constante mudança, a natureza do crescimento econômico global foi alterada pelo desenvolvimento da ciência e da tecnologia, que foi possível graças à rápida evolução dos modelos de comunicação, ciclos de vida mais curtos

dos produtos e maior taxa de desenvolvimento de novos produtos. Com importante participação nisso, a engenharia do conhecimento é um campo dentro da inteligência artificial que desenvolve sistemas baseados em regras explícitas manualmente construídas.

Segundo Darai, Singh e Biswas (2010), a engenharia do conhecimento pode ser entendida como o aspecto da engenharia de sistemas que trata de problemas com grau de incerteza, enfatizando a aquisição de conhecimento sobre um processo e representando esse conhecimento em forma de regras com probabilidades associadas. Tais sistemas são programas de computador que contêm grandes quantidades de conhecimento, regras e outros mecanismos de raciocínio para fornecer soluções para problemas do mundo real.

Rouse (2018) cita o MYCIN (SHORTLIFFE; BUCHANAN, 1985) como um dos primeiros exemplos de sistemas baseado no conhecimento criado para ajudar os médicos a diagnosticar infecções bacterianas. O mesmo autor ainda afirma que sistemas baseados no conhecimento ainda são bastante empregados em aplicações tão diversas como análise de caminhos de avalanches, diagnóstico de falhas de equipamentos industriais e gerenciamento de caixa.

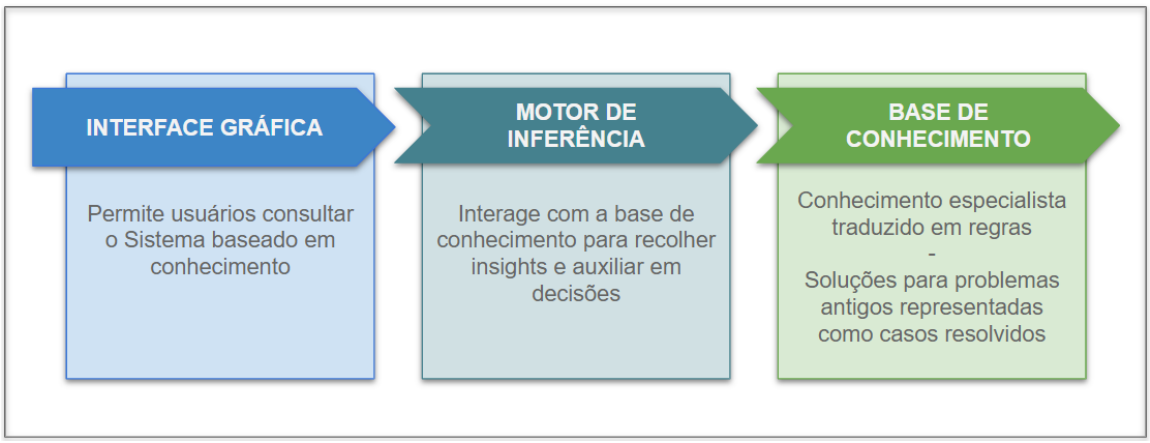
No capítulo 25 do seu trabalho *"Knowledge engineering," in Handbook of Knowledge Representation* (HARMELEN; LIFSCHITZ; PORTER, 2008), A. Schreiber mostra que a disciplina da engenharia do conhecimento surgiu do trabalho inicial em sistemas especialistas (*expert systems*) nos anos setenta. E com a crescente popularidade dos sistemas baseados no conhecimento (como eram então chamados), surgiu também a necessidade de uma abordagem sistemática para a construção de tais sistemas, semelhante às metodologias na engenharia de software de fluxo principal.

Ao longo dos anos, a disciplina de engenharia do conhecimento evoluiu para o desenvolvimento de teoria, métodos e ferramentas para o desenvolvimento de aplicações que ajudassem no gerenciamento de conhecimento. Em outras palavras, ela fornece orientação sobre quando e como aplicar técnicas específicas de representação de conhecimento para resolver problemas específicos (HARMELEN; LIFSCHITZ; PORTER, 2008). E, apesar da evolução também da aprendizagem de máquina para classificação de textos, pode ser visto em (LU, 2020) que a área da engenharia do conhecimento ainda está bastante aquecida e relevante.

Alguns dos motivos da popularização dessa abordagem também foram apresentados já na arquitetura dos sistemas baseados em conhecimento apresentados por (ROUSE, 2018) na Figura 2:

Em relação às principais características da utilização da Engenharia do conhecimento, podemos pontuar: pode ser utilizado como processo de modelagem de conhecimento; estimula

Figura 2 – Arquitetura genérica de um Sistema Baseado em Conhecimento




**Fonte:** ROUSE (2018, tradução livre)

um planejamento, entendimento e representação mais explícita do problema tratado; procura formalizar o conhecimento antes da implementação da base de conhecimento; motiva também a realização de avaliações e refinamentos da base de conhecimento construída. Além disto, a utilização de regras para representação do conhecimento permite a otimização de manutenção nas regras de negócio do sistema implementado. Todos esses pontos serão abordados com mais detalhes no Capítulo 3.

### 2.5.2 Aprendizagem de Máquina

Na Aprendizagem de Máquina (AM) (MICHIE et al., 1994), um sistema que aprende melhora seu comportamento futuro com base na experiência do passado. O processo de construção de aprendizagem também envolve: aquisição de conhecimento e melhoria da performance.

É importante ressaltar duas abordagens em AM: aprendizado supervisionado, que dispõe de um conjunto de dados etiquetados — rotulados manualmente ou não; e aprendizado não supervisionado, onde não há etiquetagem do conjunto de dados, ou seja, sem classificação prévia. As tarefas mais comuns do aprendizado supervisionado são a classificação e a regressão, enquanto a tarefa mais comum do aprendizado não supervisionado é o agrupamento (*clustering*) (CAMILO; SILVA, 2009).

Na aprendizagem supervisionada, existem diversos conjuntos de dados, onde cada instância de dado possui uma variável-alvo pré-definida (*target*). Para atribuir a variável-alvo correspondente para uma nova instância, um algoritmo de AM realiza uma série de procedimentos, como indução de regras e cálculos probabilísticos, a partir das características observadas nos conjun-

tos de dados etiquetados inicialmente fornecidos. Assim, os algoritmos conseguem "aprender" as características gerais desses dados e realizar poderosas inferências sobre novos dados (ex.: atribuição da *target* correta para uma nova instância, em um problema de classificação). O algoritmo treinado é então "persistido" em um modelo, que poderá ser utilizado futuramente em novas amostras de dados, isto é, que não foram previamente etiquetadas.

Essa abordagem tem dois ciclos distintos: criação/indução e uso do modelo. O ciclo de criação do modelo é subdividido em duas fases: treinamento e testes. Tão logo o processo de aprendizado (indução) estiver concluído, o modelo deve ser avaliado, de modo a verificar estatisticamente a qualidade dos resultados alcançados. Essa avaliação deve utilizar dados novos, isto é, dados ainda não conhecidos pelo modelo. O uso de dados novos nessa etapa contribui para uma avaliação honesta e realista do desempenho do modelo criado, uma vez que esses dados não estavam presentes no processo de treinamento. Por isso é necessário que haja uma divisão dos dados em treinamento e teste (DAMASCENO, 2020).

Existem casos em que dados são divididos em 3 categorias: treinamento, teste e validação. Os dados de validação servem para ajustar os parâmetros do modelo e para aumentar sua capacidade de inferência para dados desconhecidos. Em termos percentuais, a divisão de dados em treinamento e teste, resulta em 70% e 30% respectivamente; já a divisão em treinamento, teste e validação, resulta em 70%, 20% e 10% respectivamente (DAMASCENO, 2020).

A seção anterior, 2.5.1, trata de uma abordagem para construção de classificadores e a aprendizagem de máquina é outra abordagem que se tornou muito popular nas últimas décadas. Dentro dessa abordagem estão incluídas as seguintes técnicas: K-Nearest Neighbors, Support Vector Machines e Naïve Bayes. A seguir temos uma breve descrição de cada uma delas, respectivamente.

### *K-Nearest Neighbors (K-NN)*

O método mais simples de aprendizagem de maquina é o algoritmo *K-Nearest Neighbor*, kNN ou K-NN (COVER; HART, 1967). Este algoritmo assume que todas as instâncias correspondem a pontos no espaço n-dimensional $\mathbb{R}^n$. Os vizinhos mais próximos de uma instância são definidos em termos da distância euclidiana padrão.

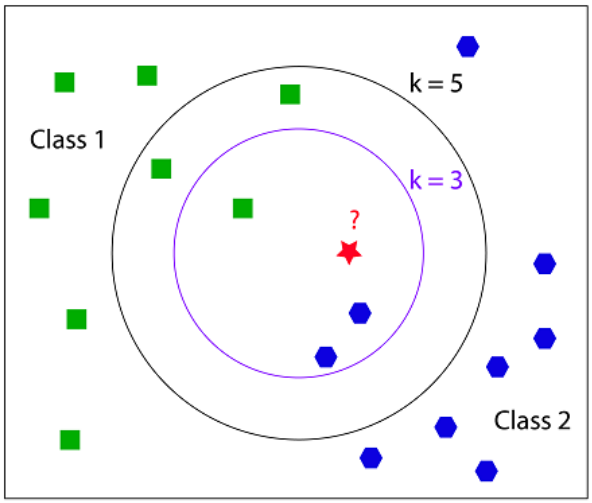
O K-NN considera que as instâncias são representadas por pontos num espaço de dimensão $n$. Para classificar uma nova instância $x$, o K-NN atribui a $x$ a classe mais frequente dentre as $k$ instâncias de treinamento mais próximas, ou seja, que possuam a menor distância de $x$

(RODRIGUES, 2009). Portanto, os dois principais pontos-chave do K-NN são: o valor de $k$ e a métrica de distância. Onde a variável $k$ determina a quantidade de instâncias mais próximas que serão usadas para verificar a qual classe a nova instância pertence.

As fraquezas mais comumente comentadas sobre o K-NN estão relacionadas com: a definição de um valor ótimo para $k$ e o uso de todas as características das instâncias de treinamento para a computação das distâncias (KORDE; MAHENDER, 2012). Em relação à constante $k$, não há um valor padrão a ser sempre atribuído, pois esse valor pode variar conforme a base de dados. Dependendo do problema que se queira resolver, pode ser usado um algoritmo de otimização para encontrar o melhor valor de $k$. Entretanto, isso geralmente impacta a performance do algoritmo no momento de determinar $k$.

Por exemplo, como pode ser visto na Figura 3 abaixo, na definição da classe da estrela vermelha. Aplicando o algoritmo K-NN para $k = 3$, obtemos que a estrela vermelha seria da classe hexágono azul. Entretanto, alterando o valor de $k$ para 5, a estrela vermelha seria da classe quadrado verde. A medida que o valor de $k$ aumenta, mais cálculos de distância tem que ser realizados e por isto, a performance do algoritmo cai.

Figura 3 – K-NN aplicado a estrela com k=3 e k=5




**Fonte:** Elaborada pelo autor (2020)

Outra alternativa é utilizar o método de tentativa-erro de modo a encontrar $k$ por meio empírico. Nesse caso, é aconselhável utilizar valores ímpares/primos. Em relação à distância, computá-la é tarefa custosa, principalmente se o problema em questão possuir um número grande de instâncias, levando o algoritmo a consumir um tempo de cálculo muito longo.

Mais precisamente, considerando uma instância arbitrária $x$ ser descrita pelo vetor de característica abaixo:

$$< a_1(x), a_2(x), ..., a_n(x), > \tag{2.2}$$

onde $a_r$ (x) denota o valor da $r$-ésima característica da instância $x$. Então, a distância entre duas instâncias $x_i$ e $x_j$ é definida como $d(x_i, x_j)$, em que no aprendizado do vizinho mais próximo a função pode ser de valor discreto ou de valor real. Considerando inicialmente o aprendizado de funções-alvo com valores discretos, da forma $f : \mathbb{R}^n \rightarrow V$, onde $V$ é o conjunto finito $v_l, ...v_n$, Mitchell (1997) descreve em seu livro, *Machine Learning*, o algoritmo *K-Nearest Neighbor* para uma função alvo de valor discreto:

---

Algoritmo de treinamento:

- Para cada exemplo de treinamento $(x, f(x))$, adicione o exemplo à lista *exemplos de treinamento*. Isso pode ser considerado o conjunto de treinamento para o algoritmo, embora nenhuma etapa de treinamento explícita seja necessária.

Algoritmo de classificação:

- Dada uma instância de consulta xq a ser classificada,
  - considere $x_l ... x_k$ denotando as $k$ instâncias de exemplos de treinamento mais próximos de $x_q$
  - resulta em :

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{argmax} \sum_{i=1}^{k} \delta(v, f(x_i)) \tag{2.3}$$

  - onde, $\delta(a, b) = 1$ se $a = b$ ou, caso contrário, $\delta(a, b) = 0$.

---

Como mostrado acima, o valor $\hat{f}(x_q)$ retornado por esse algoritmo como sua estimativa de $f(x_q)$ é apenas o valor mais comum de $f$ entre os $k$ exemplos de treinamento mais próximos de $x_q$. Se escolhermos $k = 1$, o algoritmo *1-Nearest Neighbor* atribui a $\hat{f}(x_q)$ o valor $f(x_i)$ em que $x_i$ é a instância de treinamento mais próxima de $x_q$. Para valores maiores de $k$, o algoritmo atribui o valor mais comum entre os $k$ exemplos de treinamento mais próximos.

De acordo com (SAMMUT; WEBB, 2011), o K-NN é um algoritmo baseado em instâncias, que são agrupadas de acordo com a maior proximidade que elas têm entre si. O aprendizado

deste algoritmo é considerado tardio. Esse atraso se deve ao fato da classificação de novas instâncias acontecer sob demanda, necessitando que os dados de treinamento sejam armazenados e recuperados à medida que for solicitada a classificação para uma nova instância. Entretanto, esse tipo de algoritmo permite o aprendizado de forma incremental.

Na classificação textual e para grande maioria dos conjuntos de dados textuais, apenas uma pequena parcela do vocabulário total é útil para determinar a classe de novas instâncias. Uma abordagem que pode resolver tal problema é atribuir pesos para diferentes características (i.e., termo/palavras dos documentos) (KORDE; MAHENDER, 2012).

Por fim, as principais características do K-NN de acordo com Rodrigues (2009) são: não gerar regras explícitas; ter alto custo computacional; requerer definição de parâmetros (ex.: valor de $k$); ser estável em relação aos dados de treinamento; ser sensível a características redundantes e irrelevantes; também ser sensível a classes desbalanceadas.

*Support Vector Machines (SVM)*

O algoritmo SVM (Máquinas de Vetores Suporte do inglês *Support Vector Machines*) utiliza conceitos de modelos lineares e aprendizagem baseada em instâncias. Este tipo de modelo tem se mostrado eficiente comparado com a grande maioria dos classificadores em diversas aplicações. Ele baseia-se no fato que em altas dimensões do espaço de características, todos os problemas tendem a se tornar linearmente separável.

Segundo Lorena e Carvalho (2007), as SVMs são embasadas pela teoria de aprendizado estatístico, desenvolvida por Vapnik (VAPNIK, 2013) a partir de estudos iniciados em (VAPNIK, 1999). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, definida como a sua capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu.

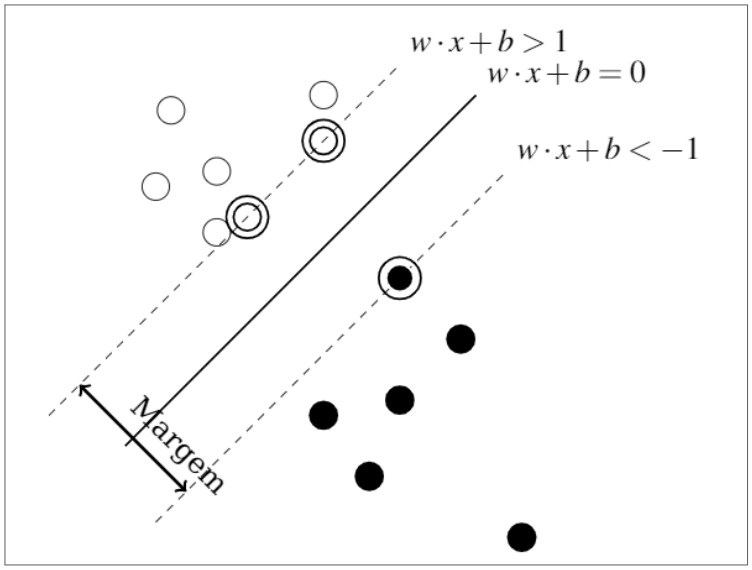
Em complemento, SVM é um algoritmo discriminativo formalmente definido por um hiperplano de separação, ou seja, dadas as instâncias etiquetadas de treinamento, é gerado um hiperplano ótimo capaz de classificar novas instâncias. Se considerarmos um espaço de duas dimensões, esse hiperplano é representado por uma linha que separa um plano em duas partes, onde cada classe fica em uma dessas partes (PATEL, 2017).

Já para Rodrigues (2009) e Sebastiani (2002), o SVM baseia-se na ideia de encontrar um hiperplano ideal que separe dois conjuntos de pontos linearmente separáveis no espaço. Tão logo o hiperplano é encontrado, basta, para realizar a classificação de uma nova instância

(ponto), verificar em qual área (lado esquerdo ou direito) deste hiperplano ela se encontra. O SVM consiste numa abordagem geométrica para o problema de classificação, onde cada uma das instâncias de treinamento é vista como um ponto $x_i$ num espaço $\mathbb{R}^M$, e o aprendizado dá-se pela divisão de elementos positivos e negativos neste espaço.

Abaixo, na Figura 4, é demonstrado visualmente a separação dos círculos pretos preenchidos em relação aos círculos brancos. Essa separação é dada pela reta $w \cdot x + b = 0$. Esse hiperplano ideal é ortogonal às linhas mais próximas, denominadas fronteiras, que conectam as margens das duas classes (linhas pontilhadas) e o cruza na metade do caminho.

Figura 4 – Explicação visual de um hiperplano ótimo por SVM




**Fonte:** Elaborada pelo autor (2020)

Vale ressaltar que podem existir infinitos hiperplanos que separam dois conjuntos de pontos linearmente separáveis. O desafio do SVM é encontrar o hiperplano ótimo que maximize a **margem** (distância entre dois elementos de classes diferentes) para os pontos mais próximos (vetores de suporte) a ele diminuindo assim as chances de classificar novos pontos deforma equivocada. Na ilustração da Figura 4, as linhas tracejadas são os vetores de suporte para as instâncias de treinamento, e cada instância que representa um vetor de suporte é um circulo branco ou um círculo preto.

Apesar da ideia do SVM ser melhor compreendida nas situações em que os pontos positivos e negativos são linearmente separáveis, ele também é aplicável em casos não linearmente separáveis. Para isso, o SVM utiliza uma função de kernel. Segundo Rodrigues (2009), essa função tenta mapear os pontos originais para um novo espaço onde eles sejam linearmente separáveis, e encontrar o hiperplano ótimo em seguida. Várias funções de kernel têm sido

usadas em SVMs, como funções polinomiais (lineares e quadráticas), e funções gaussianas (SEBASTIANI, 2002).

Por fim, em relação às principais características do SVM, podemos pontuar, segundo Rodrigues (2009): alto custo computacional; não gera regras explícitas; exige definição de parâmetros; estável em relação às instâncias de treinamento; pouca sensibilidade a características redundantes; sensível a classes desbalanceadas.

*Naïve Bayes (NB)*

Langley e Sage (1994) e Mitchell (1997) afirmam que o objetivo do *Naïve Bayes* (ou NB) é determinar a classe $C_map$ mais provável de uma nova instância (no nosso caso, novo documento de texto) a partir das características que descrevem essa instância. A vantagem do Naïve Bayes é a simplicidade do seu cálculo, pois lida "ingenuamente"com as características das instâncias como sendo independentes, de modo a obter a classe que maximize o resultado de sua fórmula. O NB também tem apresentado bons resultados em vários problemas de alta complexidade, principalmente os de classificação de textos (WITTEN; FRANK, 2002).

O NB é um método de aprendizado supervisionado bastante popular quando o assunto é classificação de texto e é um método probabilístico de aprendizado (MANNING; RAGHAVAN; SCHÜTZE, 2008). A probabilidade de um documento $d$ estar na classe $c$ é calculada. Como na classificação de texto nosso objetivo é encontrar a melhor classe para o documento, a melhor classe no NB é a classe com maior probabilidade (MAP) $c_{map}$ representada pela equação 2.4:

$$c_{map} = argmax_{c \in C}[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)] \tag{2.4}$$

Escrevemos $\hat{P}$ para a probabilidade $P$ porque não sabemos os valores dos parâmetros $P(c)$ e $P(t_k|c)$, mas serão calculados a partir do conjunto de treinamento. A equação 2.4 tem uma interpretação simples. Cada parâmetro condicional $\log \hat{P}(t_k|c)$ é um peso que indica quão bom é um indicador $t_k$ para $c$. Da mesma forma, o $\log \hat{P}(c)$ é um peso que indica a frequência relativa de $c$. As classes mais frequentes têm maior probabilidade de ser a classe correta do que as classes pouco frequentes. E a soma dos pesos dos registros anteriores e dos termos é então uma medida da quantidade de evidências existentes para o documento estar na classe, e a Equação 2.4 seleciona a classe para a qual temos mais evidências.

Algumas das principais características do NB, segundo Rodrigues (2009), são: utiliza baixo

custo computacional; não gera regras explícitas; é instável em relação às instâncias de treinamento; é sensível à redundância nas características e também ao balanceamento das classes; não é sensível a características irrelevantes.

Pranckevicius e Marcinkevicius (2017) afirmam que estudos comparativos têm demonstrado que os algoritmos bayesianos, em alguns casos, podem apresentar bons resultados de desempenho. Isto é observado principalmente se comparados a técnicas como árvore de decisão e redes neurais; mas também utilizando métodos para seleção e remoção de redundância de características.

## 2.6 AVALIAÇÃO DE CLASSIFICADORES

O método mais comum de avaliação de classificadores é através da construção da matriz de confusão. Uma matriz de confusão mostra o total de itens classificados em cada categoria, comparando a classificação manual (etiquetagem) com a classificação automática. Assim, é possível verificar o desempenho do classificador automático. A matriz de confusão é quadrada LxL, onde L é o número de classificações ou rótulos possíveis. Na Tabela 1, vemos a matriz de confusão para L=2, ou seja, para uma classificação binária.

Significado dos rótulos:

- VP - **verdadeiro-positivo**: A classe do item é positiva e foi classificada corretamente como positiva;
- FP - **falso-positivo**: A classe do item é negativa, mas foi classificada erroneamente como positiva;
- FN - **falso-negativo**: A classe do item é positiva, mas foi classificada erroneamente como negativa;

Tabela 1 – Matriz de confusão para classificação binária

| Teste - Real | Positivo | Negativo |
|---|---|---|
| Positivo | VP | FP |
| Negativo | FN | VN |

**Fonte:** Elaborada pelo autor (2020)

- VN - **verdadeiro-negativo**: A classe do item é negativa e foi classificada corretamente como negativa.

Em seguida, conceitos mais específicos serão detalhados: **acurácia**, **sensibilidade** (ou *recall*), **especificidade**, **eficiência**, **precisão** (ou valor preditivo positivo), **valor preditivo negativo** e ***F1 Score***.

**Acurácia** - A proporção de predições corretas, sem levar em consideração o que é positivo e o que é negativo. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

$$\text{Acurácia} = (VP + VN) / (P + N), \text{ ou seja,}$$

*Total de Acertos / Total de Dados no Conjunto*

**Sensibilidade ou *Recall*** - A proporção de verdadeiros positivos em relação ao total de positivos no corpus: a capacidade do sistema em predizer corretamente exemplos da classe positiva.

$$\text{Sensibilidade} = VP / (VP + FN), \text{ ou seja,}$$

*Total de positivos corretamente classificados / Total de Positivos*

**Especificidade** - A proporção de verdadeiros negativos em relação ao total de negativos no corpus.

$$\text{Especificidade} = VN / (VN + FP), \text{ ou seja,}$$

*Total de negativos corretamente classificados / Total de Negativos*

**Eficiência** - A média aritmética da Sensibilidade e Especificidade. Na prática, a sensibilidade e a especificidade variam em direções opostas. Isto é, geralmente, quando um método é muito sensível a positivos, tende a gerar muitos falso-positivos, e vice-versa. Assim, um método de decisão perfeito (100% de sensibilidade e 100% especificidade) raramente é alcançado, e um equilíbrio entre ambas deve ser atingido.

$$\text{Eficiência} = (\text{Sens} + \text{Espec}) / 2$$

**Precisão** ou Valor Preditivo Positivo - A proporção de verdadeiros positivos em relação a todas as predições positivas. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

$$\text{Precisão} = VP / (VP + FP), \text{ ou seja,}$$

*Total de positivos corretamente classificados / Total de Predições Positivas*

**Valor Preditivo Negativo** - A proporção de verdadeiros negativos em relação a todas as predições negativas. Esta medida também é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

Valor Preditivo Negativo = VN / (VN + FN), ou seja,

*Total de negativos corretamente classificados / Total de Predições Negativas*

**F1 Score** - Essa métrica combina precisão e sensibilidade de modo a trazer um valor único que indique a qualidade geral do modelo, e trabalha bem até com conjuntos de dados que possuem quantidade desbalanceada de exemplos das classes positiva e negativa. A fórmula que define o F1 é a seguinte:

$$F1 = \frac{2 * precisao * sensibilidade}{precisao + sensibilidade} \tag{2.5}$$

Logo, um modelo perfeito (1/1) chegaria a um score de no máximo 1.

A seguir, veremos trabalhos relacionados ao tema desta dissertação.

## 2.7 TRABALHOS RELACIONADOS

Considerando o escopo de características bastante específicas deste trabalho, faz-se necessário mencionar a dificuldade de encontrar trabalhos relacionados que englobem simultaneamente classificação de texto, automação de casos de teste e utilização da engenharia do conhecimento e regras. Outros pontos que corroboraram com esse desafio foram a já mencionada similaridade entre os casos de teste e a baixa quantidade de exemplos de casos de teste, dificultando o uso de técnicas de Aprendizagem de Máquina.

Esse contexto nos levou a considerar um escopo mais amplo na área de classificação de artefatos da engenharia de testes para nossos trabalhos relacionados. Dentro desse novo escopo, foram encontrados trabalhos datados desde 1970. A seguir, serão apresentados alguns trabalhos relacionados de interesse, em ordem cronológica.

Suphapala et al. (1970) propuseram um método algorítmico simples, mas prático, para reduzir a suíte de teste para conter apenas os casos de teste relevantes utilizando um método de partição por categoria (*Category-Partition*), trabalhando em conjunto com o método *pairwise* ou *all-pairs*. A ideia central do trabalho se dá na identificação das relações entre a partição de categoria das especificações de entrada e as restrições do programa que, subsequentemente, são empregadas para construir uma máquina de estados finitos. Como tal, todos os caminhos

que conectam os estados inicial e final representam os casos de teste necessários. A redução no número de quadros de teste — do inglês, *test frames* — gerados com base no método proposto em comparação com as abordagens convencionais mostrou-se bastante significativa. No exemplo mostrado, uma máquina de estados que geraria 108 casos de teste válidos, após a aplicação do método proposto, gerou apenas 15 casos de teste com cobertura equivalente aos 108. Com isso, obteve-se uma redução de aproximadamente 86% de casos gerados.

Koochakzadeh e Alhajj (2011) propuseram uma técnica automática para categorizar em grupos dinâmicos os casos de teste com base em sua cobertura (*test coverage*). Foi proposta a criação de uma rede social de casos de teste para identificação de grupos otimizados de teste, que pode ser executado dinamicamente ao longo do ciclo de vida do sistema. Nessa criação, as informações de cobertura foram usadas para definir os links entre os casos de teste. A rede também é usada para identificar grupos maiores de casos de teste, como pacotes de teste, utilizando um algoritmo de *clustering* (*K-means*), descobrindo os casos com maior similaridade, em termos de cobertura. Na avaliação da técnica utilizada, foram utilizados três sistemas de código aberto com conjuntos de teste unitários (*JUnit*) disponíveis para identificar pacotes de teste. O resultado mostrado indicou que a técnica pode ser usada para categorizar casos de teste automaticamente, melhorando inclusive a qualidade dos pacotes de teste.

Shimagaki et al. (2018) trabalharam em um ecossistema de aplicativos para *smartphones* Android, onde o tamanho do sistema é muito grande e o número de casos de teste necessários para cobrir as suas funcionalidades é substancial. Para o gerenciamento dos casos de teste e suas execuções são utilizados rótulos chamados de *features labels* (FLs). Uma vez que, para os desenvolvedores, a atribuição manual de FLs a milhares de casos de teste é uma tarefa demorada, podendo levar a casos de teste rotulados de maneira incorreta, foi criado um sistema automatizado que sugere os rótulos aos desenvolvedores para seus casos de teste, em vez de rotulagem manual. Na solução proposta, foram utilizados modelos de aprendizado de máquina para prever e rotular aproximadamente 10.000 casos de teste desenvolvidos na empresa com base nas funcionalidades testadas no Sistema Android. Para validação da solução, foram realizados experimentos quantitativos e qualitativos que mostraram um desempenho aceitável.

Lima et al. (2018) apresentaram um novo processo de composição de planos de teste para equipes exploratórias. Esse processo se baseia principalmente na automação da inspeção de *Release Notes*, com foco na extração de áreas de SW relevantes a serem exploradas pelos testadores. O protótipo implementado utiliza técnicas de Inteligência Artificial, PNL e Recupe-

ração de Informação para classificar os pedidos de mudança — *change requests* (CRs) — entre relevantes e não-relevantes; e extrair as áreas relevantes a serem exploradas. Os experimentos revelaram um ganho significativo de produtividade por parte dos testadores, superando o processo manual atualmente adotado pela empresa parceira no trabalho e, assim, contribuindo para um método mais eficiente de construção de planos de testes exploratórios. É interessante citar que no trabalho de Lima et al. (2018), a empresa parceira é a mesma empresa parceira deste trabalho, mas os escopos dos problemas são significativamente diferente.

Finizola (2019) criou um processo para automatizar parte das atividades de análise de *feedbacks* de *Dogfooding*, a fim de obter informações úteis para direcionar as atividades de Testes Exploratórios (TEs). O protótipo do sistema implementado utilizou conceitos de Aprendizagem de Máquina e Recuperação de Informação (RI). Para classificação dos *feedbacks* entre relevantes e irrelevantes, um classificador foi construído usando AM e testados utilizando diferentes algoritmos. O classificador deve ser periodicamente atualizado (retreinado) de forma semiautomática, para não se tornar obsoleto. Os testes realizados com o protótipo, principalmente os de classificação e usabilidade, apresentaram resultados positivos: foi possível obter um modelo de classificação com valores de acurácia e F1-score de 87% e 88% respectivamente; também foi possível obter um ganho de tempo de 47.5% e 64.29% em comparação com os procedimentos manuais nos dois testes de usabilidade realizados, além de um ganho de 55.56% de *feedbacks* relevantes identificados em um desses testes.

Abbas et al. (2020) propuseram uma solução semiautomática para categorização não-binária de casos de teste, baseada em semânticas de palavras-chave, para identificar a qual "Propriedades Extra-Funcionais" — *Extra-Functional Properties* (EFPs) — o caso de teste está associado. A solução é a primeira etapa para a seleção de casos de teste com base na cobertura das EFPs. No trabalho, foi relatada uma avaliação preliminar com dados industriais para a categorização de teste para aspectos de segurança. Os resultados da avaliação mostram que (1) a abordagem baseada em palavras-chave pode ser usadas para categorizar testes de acordo com as propriedades extra-funcionais dos sistema testado e (2) a abordagem utilizada pode ser melhorada considerando as informações contextuais das palavras-chave. No caso deste trabalho, os casos de teste são relacionados a uma empresa parceira da área de transporte de cargas, a Bombardier Transportation.

Nos trabalhos citados nesta seção, foi possível notar que a classificação ainda é uma atividade bastante realizada quando se trata de artefatos da engenharia de teste. Para realização dessa atividade de classificação, foi registrado a utilização de uma grande variedade de téc-

nicas: desde a representação dos casos de teste em uma máquina de estado e sua derivação; passando pela criação de uma rede social de casos de teste para identificação de grupos otimizados de teste; também utilizando modelos de aprendizado de máquina para rotular casos de teste; ainda utilizando PNL, recuperação de informação e aprendizagem de máquina para construção e refinamento de *charters* de testes exploratórios; bem como a utilização de palavras chaves para EFPs.

Entretanto, nenhum destes trabalhos se assemelha à solução proposta nesta dissertação: classificar casos de teste executados manualmente como automatizável ou não. Tampouco foi encontrado um trabalho que utilize e valide a técnica de engenharia do conhecimento e regras para classificação de casos de teste.

## 2.8 CONSIDERAÇÕES FINAIS

Como foi apresentado neste capítulo, o problema de classificação de texto ou documentos é bastante complexo e abarca diversas problemáticas envolvendo suas várias etapas, desde a estruturação dos documentos, passando pelo pré-processamento, redução de dimensionalidade da base de documentos e também pela seleção de características relevantes para classificação.

Ainda foram apresentadas algumas técnicas e algoritmos mais utilizados na classificação de texto. Eles foram, Naïve Bayes (NB), Support Vector Machines (SVM), K-Nearest Neighbors (K-NN) e a Engenharia do Conhecimento (EC) com aplicação de regras definidas.

Apesar de reconhecer a evolução no campo da aprendizagem de máquina, Langley e Simon (1995) reconhecem que a aprendizagem de máquina ainda não invalida inteiramente a engenharia do conhecimento como um framework para a construção de sistemas baseados no conhecimento. E, ainda, como foi apresentado por Dalal e Zaveri (2011), nenhum algoritmo é melhor que o outro, a escolha de uma abordagem passa pelo entendimento do problema e suas características.

Em seu trabalho, Langley e Simon (1995) mostram como são as etapas do processo da engenharia do conhecimento: formulação do problema, determinação da representação, coleta dos dados, avaliação do conhecimento, exibição do conhecimento e refinamento do processo realizado. Eles indicam também que nos casos trabalhados, o aprendizado de máquina não automatizou completamente o processo de engenharia do conhecimento, mas substituiu a engenharia do conhecimento por duas tarefas mais simples: caracterizar o problema e projetar uma boa representação. Os desenvolvedores não precisam minimizar esse fato; reduzir qualquer

tempo e esforço necessários para desenvolver sistemas baseados no conhecimento, por menor que seja a automação, pode produzir sistemas de grande valor prático.

Outro ponto bastante relevante é a discussão levantada por Deparkes (2017), onde é questionado qual a melhor abordagem: utilizar sistemas baseados em conhecimento com regras ou utilizar técnicas de aprendizagem de máquina. A conclusão, naturalmente óbvia, é "depende". Os sistemas baseados em regras costumam oferecer soluções táticas mais rápidas. O requisito de contribuição de especialistas em negócios pode ajudar a uma maior adesão dos negócios e facilitar a explicação de como as decisões foram tomadas. Muitos projetos começam com um especialista ou sistema baseado em regras para explorar e entender o sistema.

Nesta dissertação, a questão a ser trabalhada apresenta-se mais compatível com os problemas resolvidos através da Engenharia do Conhecimento com Regras do que utilizando uma técnica de aprendizagem de máquina ou ainda do que uma combinação de várias dessas técnicas. A abordagem baseada em AM de máquina foi apresentada aqui porque havia a intenção de implementar um classificador com essa abordagem para comparar com nossa solução. Porém, não foi possível concretizar essa intenção devido a falta de um corpus etiquetado de dados para este problema (ver Capítulos 4 e 5). Assim, no próximo Capítulo, 3, será mais detalhado o conteúdo já apresentado sobre a Engenharia do Conhecimento e a utilização de Regras.

# 3 ENGENHARIA DO CONHECIMENTO E REGRAS

Uma breve noção de Engenharia de Conhecimento foi apresentada na Seção 2.5.1. Seu conceito se tornou muito importante nos últimos anos. A Engenharia do Conhecimento (EC) é um campo dentro da Inteligência Artificial que desenvolve Sistemas Baseados em Conhecimento (SBC).

Segundo Darai, Singh e Biswas (2010), também pode-se ver a engenharia do conhecimento como o aspecto da engenharia de sistemas que trata de requisitos incertos do processo, enfatizando a aquisição de conhecimento sobre um processo e representando esse conhecimento em um sistema baseado em conhecimento. Tais sistemas contêm grandes quantidades de conhecimento em forma de regras e mecanismos de raciocínio para fornecer soluções para problemas do mundo real.

Os primeiros anos da engenharia do conhecimento foram marcados por problemas. Os engenheiros de conhecimento descobriram que adquirir conhecimento de alta qualidade suficiente para construir um sistema robusto e útil era uma atividade muito demorada e muito cara (MILTON, 2007).

Em seu livro, *The Fifth Generation*, Feigenbaum e McCorduck (1983) introduziram uma das primeiras definições da engenharia do conhecimento e sua relevância para dominar o que eles denominavam como sendo a indústria do conhecimento, a revolução mais importante trazida pelos computadores. Por sua vez, a EC deve ser utilizada para criar sistemas que representem o conhecimento e/ou realizem atividades de especialistas. Esses últimos tendem a ser pessoas importantes e ocupadas e, por isso, é vital que os métodos utilizados minimizem o tempo que cada especialista gasta no trabalho participando de sessões de aquisição de conhecimento.

Como tal, a aquisição de conhecimento foi identificada como o gargalo na construção dos SBC. Isso levou a aquisição de conhecimento a se tornar um importante campo de pesquisa em engenharia do conhecimento. Portanto, o objetivo do pesquisador ou desenvolvedor de SBC/EC é buscar métodos e ferramentas que tornam a árdua tarefa de capturar e validar o conhecimento de um especialista mais eficiente e eficaz.

Uma das principais aplicações de SBC são os sistemas especialistas (SE) (JACKSON, 1998), projetados para aplicar conhecimento especializado na solução de problemas complexos e emular os processos de raciocínio de um profissional especialista. Aqui é importante ressaltar que nem todo SBC é considerado um SE, pois alguns SBC tratam de conhecimento não especi-

alista, de ampla divulgação, como por exemplo um sistema de classificação do reino animal (mamíferos, aves, peixes, etc). Exemplos típicos de SE são: sistemas de diagnóstico médico, previsão de bolsa de valores, detecção de falhas em sistemas elétricos ou eletromecânicos, exploração mineral, entre outros. Tais aplicações requerem conhecimento de um especialista. Por fim, é importante ressaltar que nem todo SE é um SBC. DE fato, SE podem ser implementados tanto como um SBC (com regras explícitas) quanto usando Aprendizagem de Máquina.

O planejamento, desenvolvimento e manutenção de SBC tem muito em comum com a engenharia de software. Esses sistemas são usados em muitos domínios da ciência da computação, como inteligência artificial, incluindo banco de dados, mineração de dados, sistema especialista, sistema de suporte a decisões e sistema de informações geográficas. Como base para a construção desses sistemas baseados em conhecimento, a EC também está relacionada à captura e representação do nosso entendimento, de como o raciocínio e a lógica humana funcionam. Por último, mas não menos importante, a crescente produção de dados e informação gera uma demanda cada vez maior por técnicas mais eficientes de criação e exploração de conhecimento.

Nas próximas seções serão detalhados alguns conceitos sobre o conhecimento. Em seguira, será dado um foco mais detalhado sobre a engenharia do conhecimento com base em regras por este ser parte fundamental do estudo. Após isto, será apresentada a ferramenta Drools, que é uma ferramenta de processamento de regras bastante popular, e que é bastante usada na construção de SBC. Por fim, serão realizadas as conclusões acerca dos tópicos discutidos neste capítulo.

## 3.1 CONHECIMENTO: DEFINIÇÕES E TIPOS

O conhecimento é definido de várias formas (CONHECIMENTO, 2009):

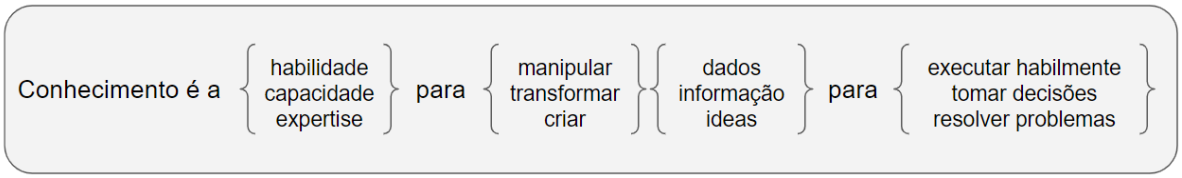
1. "Entendimento sobre algo; saber: conhecimento de leis;"
2. "Ação de entender por meio da inteligência, da razão ou da experiência;"
3. "Por Extensão, ação de dominar uma ciência, uma arte, um método, um procedimento, etc.: ele tinha grande conhecimento de história;"
4. "Historicamente: reunião das referências ou informações guardadas pela humanidade;"

5. "Filosoficamente: ação ou capacidade que faz com que o pensamento consiga apreender um objeto, através de meios cognitivos que se combinam (intuição, contemplação, analogia, etc.)."

Ainda, Milton (2007) também define conhecimento como algo equivalente à experiência ou à capacidade das pessoas de realizarem um trabalho eficaz e eficiente e de lidar com situações complexas.

Pode-se afirmar, então, que a aquisição de conhecimento envolve processos cognitivos complexos: percepção, aprendizado, comunicação, associação e raciocínio. E o termo "conhecimento" também é usado para significar a compreensão de um sujeito com a capacidade de usá-la para um propósito específico. Na Figura 5, é possível verificar algumas variações possíveis para a definição de conhecimento.

Figura 5 – Definições para o conhecimento




**Fonte:** Elaborada pelo autor (2020)

Existem várias maneiras de olhar para o conhecimento, mas destacam-se duas dimensões importantes que podem ser utilizadas para descrever o conhecimento (MILTON; CLARKE; SHADBOLT, 2006):

- procedimental vs. declarativo ou estático vs. Meta-conhecimento;
- básico ou explícito vs. profundo ou tácito;

### Conhecimento procedimental

Conhecimento procedimental, também chamado de conhecimento procedimental, é o conhecimento que uma pessoal nomearia se estivesse completando a frase: **Eu sei como ...**. Então, se for possível afirmar: ***Eu sei como* fazer um bolo**, esse tipo de conhecimento é procedimental, pois trata-se de processos, tarefas e atividades. É sobre as condições e ordem em que as tarefas são executadas.

### Conhecimento declarativo ou estático

Por sua vez, o conhecimento declarativo ou estático é o que alguém diria se estivesse completando a frase: **Eu sei que ...**. Então, se puder afirmar: ***Eu sei que* um bolo é mais calórico que um copo d'água**. Esse tipo de conhecimento é conceitual. Portanto, trata-se das maneiras como as coisas (que chamamos de 'conceitos') estão relacionadas entre si e sobre suas propriedades. Uma forma importante de conhecimento conceitual, declarativo ou estático são as taxonomias, ou seja, representação de classes e pertencimento em classes.

### Meta-conhecimento

Meta-conhecimento é conhecimento sobre conhecimento. É o conhecimento sobre a operação de sistemas baseados em conhecimento, isto é, sobre suas capacidades de raciocínio. Também podem ser utilizados como as estratégias para resolução de conflitos no controle das inferências realizadas pelo SBC.

### Conhecimento tácito

Conhecimento tácito é o conhecimento que as pessoas carregam em suas mentes e é, portanto, difícil de acessar. O conhecimento tácito é considerado mais valioso porque fornece contexto para pessoas, lugares, ideias e experiências. A transferência efetiva de conhecimento tácito geralmente requer amplo contato e confiança pessoal. O conhecimento tácito não é facilmente compartilhado. O conhecimento tácito consiste frequentemente em hábitos e cultura que não reconhecemos por nós mesmos.

### Conhecimento explícito

Por outro lado, conhecimento fácil de comunicar é chamado conhecimento explícito. O processo de transformação do conhecimento tácito em conhecimento explícito é conhecido como codificação ou articulação. Conhecimento explícito é o conhecimento que foi ou pode ser articulado, codificado e armazenado em determinadas mídias. As formas mais comuns de conhecimento explícito são manuais, documentos e procedimentos. O conhecimento também pode ser audiovisual, as obras de arte e o design do produto podem ser vistos como outras

formas explícitas de conhecimento, onde as habilidades, motivos e conhecimentos humanos são externalizados.

Na próxima seção, conhecendo a definição e algumas possíveis classificações sobre conhecimento, serão vistos princípios da Engenharia do conhecimento. Em seguida, será mostrado um processo de aplicação dessa engenharia: como planejar, adquirir, formalizar, implementar e avaliar o conhecimento e a base de conhecimento construída. Após isto, será visto como representar o conhecimento através de regras e quando usar sistemas basados em regras.

## 3.2 ENGENHARIA DO CONHECIMENTO

Segundo Abel e Fiorini (2013), a engenharia do conhecimento (EC) foi definida por Feigenbaum e McCorduck (1983) como a seguir:

> EC é uma disciplina de engenharia que envolve integrar conhecimentos em sistemas de computadores, a fim de resolver problemas complexos que normalmente exigem um alto nível de conhecimento humano. O conhecimento, como visto anteriormente, é um componente vital do projeto de engenharia. Com a engenharia do conhecimento, pode-se realizar reduções significativas nos custos e no tempo de desenvolvimento se o conhecimento for capturado por especialistas e armazenado em uma base de conhecimento (FEIGENBAUM; MCCORDUCK, 1983).

O conteúdo desta base de conhecimento pode ser utilizado de várias maneiras:

1. visando disseminar conhecimento para outras pessoas em uma organização;
2. para reutilizar o conhecimento de diferentes maneiras para diferentes fins;
3. usar o conhecimento para desenvolver sistemas inteligentes que possam executar tarefas complexas de design.

Apesar de conhecer as vantagens de registrar, utilizar e reutilizar o conhecimento, também é unanimidade na comunidade de IA que é difícil realizar essas ações em relação ao conhecimento de especialistas (MILTON; CLARKE; SHADBOLT, 2006).

Inicialmente, os especialistas não são bons em recordar e explicar o "conhecimento tácito", que opera no nível subconsciente. Portanto é difícil, senão impossível extrair tudo o que sabem. Em segundo lugar, os especialistas têm experiências e opiniões diferentes, o que exige a resolução do conflito dessas opiniões para fornecer uma única imagem coerente do que se

deseja registrar. Em terceiro lugar, os especialistas desenvolvem conceitos específicos e atalhos mentais que não são fáceis de se transmitir, comunicar ou registrar.

Para lidar com essas dificuldades, o campo da Engenharia do Conhecimento nasceu há cerca de 30 anos e um novo profissional foi criado: o engenheiro do conhecimento. Desde então, os engenheiros do conhecimento desenvolveram vários princípios, métodos e ferramentas que melhoraram consideravelmente o processo de aquisição, uso e implementação de conhecimento.

Ainda na década de 1970, mesmo sem uma definição sobre o termo "Engenharia do Conhecimento", os profissionais de IA acreditaram que o segredo para criar programas de software eficazes era enchê-los com conhecimento. No entanto, eles sabiam muito pouco sobre as técnicas e métodos necessários para extrair conhecimento de fontes textuais, bancos de dados ou especialistas seres humanos. O campo de aquisição e obtenção de conhecimento nasceu como uma área distinta.

Durante a década de 1980, a disciplina conhecida como aquisição de conhecimento atraiu rapidamente a atenção não só apenas dos envolvidos na criação de sistemas computacionais. Toda a área da ciência da administração e psicologia organizacional necessitava de técnicas para construir estruturas de conhecimento (FARRINGTON-DARBY; WILSON, 2006). No final dos anos 80 e início dos 90, o campo da gestão do conhecimento ganhou destaque em vários livros. As empresas se tornaram cada vez mais interessadas em seus ativos de conhecimento: como elas poderiam representar o conhecimento armazenado em suas empresas, como poderiam protegê-lo e como poderiam atribuir valor a ele. Todo o campo da gestão do conhecimento acrescentou uma nova dimensão ao requisito de ferramentas e técnicas para modelar formalmente o conhecimento.

Mais recentemente, o extraordinário surgimento da maior construção de informações da história da humanidade, a *World Wide Web* e a internet das coisas (IoT), nos apresentou uma nova maneira de pensar sobre como podemos organizar e reger informações. O desafio é que uma grande quantidade de informações deve ser adquirida, recuperada, indexada, classificada, estruturada e gerenciada.

Portanto, nos últimos anos, vimos pelo menos três fatores principais, todos apontando para técnicas de aquisição de conhecimento. Até o momento, não houve um tratamento único dos métodos práticos para construir sistemas, no sentido mais geral, que incorporam conhecimento, seja para fins de construção de sistemas de suporte à decisão, organização de um programa corporativo de gerenciamento de conhecimento, criação de *intra-nets* sustentadas ou modelagem do conteúdo da Web.

Nas próximas seções, serão apresentados brevemente alguns princípios importantes sobre a engenharia do conhecimento que precisam estar sempre considerados durante todo esse processo.

### 3.2.1 Princípios da engenharia do conhecimento

Antes de entrar mais a fundo nos processos da engenharia, Darai, Singh e Biswas (2010) citam importantes conceitos precedentes ao entendimento do conhecimento, trazendo-o como algo subjetivo e que por estar, neste ponto de vista, distante da exatidão das engenharias, também como algo mais relevante a ser constantemente considerado durante todo o processo.

**Princípio 1**: *Existem diferentes tipos de conhecimento*.

Os filósofos vêm estudando sobre conhecimento há milênios e parte de seu esforço tem sido a identificação de diferentes tipos de conhecimento. Como mencionado na seção anterior, há o conhecimento declarativo ou estático, conhecimento procedimental ou dinâmico, mas também abstrato, específico. Além dos tipos de conhecimento introduzidos pelo campo da lógica: conceitos, atributos e valores. Esses, por sua vez, podem ser vinculados para formar uma classe importante de regras de conhecimento.

**Princípio 2**: *Existem diferentes tipos de especialistas e de experiência*.

Smith, Harré e Langenhove (1995) comentam que, no campo da psicologia, as pessoas variam em quão bem elas lembram, processam e articulam seus conhecimentos. Além disso, uma mesma pessoa pode variar na medida em que faz isso. Reconhecer isto é importante, pois como consequência disso os engenheiros de conhecimento criaram vários métodos para superar esses problemas, como agregar e validar conhecimento entre fontes diversas.

**Princípio 3**: *Existem diferentes maneiras de representar o conhecimento*.

O campo da inteligência artificial pode ainda não ter produzido máquinas totalmente inteligentes, mas uma de suas principais realizações é a produção de vários formalismos para representar o conhecimento. Por sua vez, oferecer diferentes formalismos para representar conhecimento é uma parte vital da IA, uma vez que a facilidade de resolver um problema é fortemente influenciada pela maneira como a problemática é conceitualizada e representada.

**Princípio 4**: *Existem diferentes maneiras de usar o conhecimento*.

O conteúdo da base de conhecimento pode ser usado de várias maneiras: (i) disseminar conhecimento para outras pessoas em uma organização; (ii) reutilizar o conhecimento de diferentes maneiras para diferentes fins; (iii) usar o conhecimento para desenvolver sistemas

inteligentes que possam executar tarefas complexas, assim como as pessoas usam o conhecimento de maneiras diferentes, dependendo da tarefa que estão executando. Do ponto de vista do conhecimento, é útil poder classificar tarefas.

Essa classificação é baseada no tipo de problema que está sendo resolvido. Uma classificação, adaptada e refinada das ideias da psicologia, é uma hierarquia de tarefas de conhecimento intensivo, com base no tipo de problema que está sendo resolvido. Para isto, os engenheiros do conhecimento criam modelos que definem os tipos de conhecimento utilizados nas entradas e saídas da tarefa, bem como a forma como esse conhecimento é transformado para cada uma das tarefas. Usando esses modelos, o conhecimento envolvido em uma tarefa pode ser definido em termos da maneira como é usado para satisfazer uma ou conjunto de metas. Isso pode não apenas aumentar a eficiência da análise de tarefas e modelagem de processos, mas também permite identificar como o mesmo conhecimento é usado de maneira diferente, dependendo do contexto em que é utilizado (DARAI; SINGH; BISWAS, 2010).

**Princípio 5**: *Use métodos estruturados*.

Este princípio é baseado nos quatro primeiros. Existem diferentes tipos de conhecimento e especialistas, diferentes maneiras de representar e utilizar o conhecimento. Portanto, é necessário uma maneira de relacionar essas categorias de conhecimento, especialistas, representações e tarefas para realizar uma atividade orientada ao conhecimento. Para isso, são necessários métodos bem definidos: métodos estruturados de modo que as técnicas e ferramentas corretas sejam usadas, dependendo da situação e, principalmente, dos objetivos envolvidos. Para isso, precisamos de uma ampla variedade de técnicas e ferramentas de software.

Nas próximas seções, serão mostradas como a engenharia do conhecimento trabalha com o processo de modelagem do conhecimento, além de técnicas e ferramentas utilizadas para isto.

### 3.2.2 Engenharia do Conhecimento como processo de modelagem

Atualmente, existe um consenso de que o processo de construção de um Sistemas Baseados em Conhecimento (SBC) pode ser visto como uma atividade de modelagem. Construir um SBC significa criar um modelo computacional cujo objetivo é obter recursos para solução de problemas como faria um especialista a um especialista no domínio da aplicação. Não se destina a criar um modelo cognitivo adequado, mas um novo modelo que ofereça resultados semelhantes ao cognitivo na solução de problemas na área de interesse. Esse novo conhecimento

não é diretamente acessível, mas deve ser construído de forma estruturada. A modelagem do processo de construção de um SBC tem as seguintes consequências:

1. Como todo modelo, esse modelo é apenas uma aproximação da realidade (ODEGARD, 1964; BAILER-JONES, 2009). Em princípio, o processo de modelagem é infinito, porque é uma atividade incessante com o objetivo de aproximar o comportamento pretendido.

2. O processo de modelagem é cíclico (GUIDA; TASSO, 1995). Ou seja, novas observações podem levar a um refinamento, modificação ou conclusão do modelo já construído.

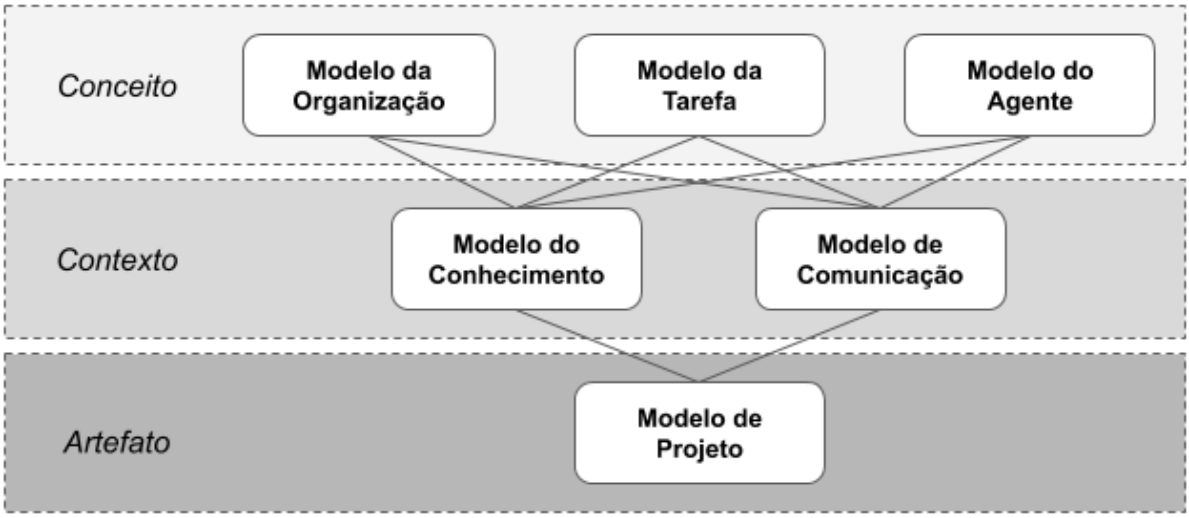
3. O processo de modelagem depende das interpretações subjetivas do engenheiro do conhecimento. Portanto, esse processo pode ser falho e uma avaliação do modelo em relação à realidade é indispensável para a criação de um modelo adequado.

Há vários fatores importantes que devemos ter em mente ao executar um projeto de engenharia de conhecimento em uma organização:

- O produto final deve ser útil para os usuários finais;
- Para ser útil, o produto final deve oferecer conhecimento de alta qualidade, correto, completo, relevante e armazenado de maneira estruturada;
- O projeto deve ser executado de maneira eficiente, aproveitando ao máximo os recursos disponíveis;
- O projeto não deve atrapalhar indevidamente o funcionamento normal da organização, portanto não deve demandar muito tempo de especialistas.

Uma metodologia de modelagem da engenharia do conhecimento já bem consolidada é o CommonKads (SCHREIBER et al., 2001). De acordo com os autores, CommonKADS é uma estrutura metodológica que norteia a construção de SBC para a resolução de problemas que envolvem conhecimento. Eles também descreve o método CommonKADS como uma extensão de métodos já existentes (análise estruturada de sistemas, teoria da organização, orientação a objetos, processos de reengenharia e gerenciamento da qualidade). Essa metodologia está organizada em três níveis de modelo, que responde a três questões centrais: **por quê?**, **o quê?** e **como?**. Todas as questões norteadoras da metodologia estão direcionadas para construir o Contexto, o Conceito e o Artefato do projeto. Conforme mostrado na Figura 6 abaixo.

Figura 6 – Camadas e modelos do CommonKADS




**Fonte:** SCHREIBER et al. (2001, tradução livre)

Para este trabalho, consideramos o processo de modelagem da engenharia do conhecimento dividido, de acordo com Abel e Fiorini (2013), nas seguintes etapas: (i) planejamento e entendimento do problema, (ii) aquisição do conhecimento, (iii) formalização do conhecimento adquirido, (iv) implementação da base de conhecimento e (v) refinamento da base e do conhecimento. Nas próximas seções, serão detalhadas essas fases.

### 3.2.3 Planejamento e entendimento do problema

Nesta fase, é realizada a identificação do domínio da aplicação a ser desenvolvida, bem como a seleção da equipe que participará das demais fases do desenvolvimento do sistema e a seleção das ferramentas de desenvolvimento que serão utilizadas pela equipe.

O planejamento é a definição de um futuro desejado e dos meios para alcançá-lo. Se um planejamento detalhado de todo o projeto, logo no seu início, transmite uma sensação de controle para o gestor do empreendimento e partes interessadas, na prática, os planos costumam não resistir à complexidade da realidade. Assim, Schreiber et al. (2001) consideram que o planejamento de um projeto deve ser feito de forma adaptativa, em ciclos espirais, dirigidos por uma avaliação sistemática do risco. Além disso, defendem que o foco do planejamento seja nas entregas a serem feitas (produtos ou artefatos), mais do que em fases ou atividades.

Uma área de conhecimento é caracterizada por seus requisitos de conhecimentos específicos e pelos processos que a compõem, suas práticas, entradas, saídas, ferramentas e técnicas. Por

exemplo, é necessário ter conhecimentos de conceitos, métodos e ferramentas que permitam delimitar e gerenciar mudanças no Escopo (área de conhecimento) de um projeto. Também é preciso dominar e aplicar métodos e ferramentas que garantam a adequada Comunicação (área de conhecimento) entre as partes interessadas no projeto, e assim por diante. Os processos, métodos, técnicas e ferramentas pertencentes a tais áreas de conhecimento são de grande importância para a efetiva condução de projetos de modo geral, inclusive aqueles que visam o desenvolvimento de Sistemas de Conhecimento.

### 3.2.4 Aquisição do conhecimento

Sobre a aquisição do conhecimento, Turban et al. (2006), em seu livro Decision Support and Business Intelligence Systems (Suporte a Decisão e Sistemas Inteligentes de Negócios, tradução livre), mostram que esta etapa envolve a aquisição de conhecimento de pessoas especialistas, livros, documentos, sensores ou arquivos de computador. Por isto, essa fase também pode ser conhecida por elicitação de conhecimento. Este pode ser específico para o domínio do problema ou para os procedimentos de solução do problema a ser resolvido, pode ser também conhecimento geral (por exemplo, conhecimento sobre tecnologia) ou pode ser conhecimento sobre conhecimento (o meta-conhecimento).

Por sua vez, Byrd (1995) verificou formalmente que a aquisição de conhecimento é o gargalo no desenvolvimento de sistemas baseados em conhecimento, principalmente em sistemas especialistas. Muita pesquisa teórica e aplicada ainda é conduzida nessa área. Uma análise de mais de 90 aplicações de SE, suas técnicas e métodos de aquisição de conhecimento está disponível em (WAGNER; CHUNG; NAJDAWI, 2003). Esses mesmos autores, em trabalhos anteriores Wagner, Najdawi e Chung (2001), listaram algumas das principais formas para aquisição de conhecimento:

- Manual:
  - Técnicas de entrevistas não estruturadas;
  - Técnicas estruturadas de entrevistas;
  - Análise de protocolos/processos;
  - Técnicas de escala psicológica (COOKE; MCDONALD, 1987);
  - Análise de caso;

  - Análise de incidentes críticos;
  - Discussões com usuários;
  - Gráficos e modelos conceituais;
  - *Brainstorming*;
  - Prototipagem;
  - Escala multidimensional;
  - *Cluster* hierárquico de Johnson;
  - Revisão de desempenho;

- Semi-automático:
  - Classificação do cartão (SPENCER; WARFEL, 2004);
  - *Repertory Grid Analysis* (RGA) (HART, 1986);

- Automático:
  - *Knowledge discovery and data mining* - KDD (LEE; ONG; QUEK, 2001);
  - Aprendizagem indutiva / computação neural (ROIGER, 2017);
  - Algoritmos genéticos (ROIGER, 2017).

É evidente, portanto, a relevância desta fase que tem por objetivo principal a identificação do conhecimento a adquirir e registro do conhecimento em linguagem natural ou usando alguma notação gráfica não-formal.

### 3.2.5 Formalização do conhecimento adquirido

Também chamada de representação do conhecimento, a fase de formalização do conhecimento adquirido tem por objetivos principais facilitar a validação com especialistas e verificação da consistência do conhecimento obtido. Para isto, são selecionados os formalismos usados para representar o conhecimento. A fim de facilitar validação, é possível representar o conteúdo coletado em forma de tabelas, XML (HAROLD, 1998), JSON (JSON, 2000), UML (BOOCH; RUMBAUGH; JACOBSON, 2006), ontologias (SIMONS, 1999), entre outras. Já no caso de uma verificação de consistência utilizando recursos mais avançados e automatizados, é

possível formalizar o conhecimento utilizando métodos formais como, por exemplo, a lógica de primeira ordem (LPO) (AYORINDE; AKINKUNMI, 2013).

A formalização do conhecimento é uma etapa básica da construção de um modelo de conhecimento ou de uma ontologia de domínio. Nela, o engenheiro de conhecimento definirá quais partes do domínio deverão ser modeladas e quais permanecerão para etapas futuras ou apenas como documentação para consulta. Os conceitos selecionados serão então aproximados às primitivas (ou construtos) de representação que melhor expressam seu significado (GRUBER, 1995) para a construção desse modelo conceitual.

Modelos conceituais não têm por objetivo apenas a descrição abstrata do conhecimento declarativo. Eles também buscam a representação do conhecimento inferencial com modelos conceituais no nível do conhecimento(ABEL; FIORINI, 2013). Entre as mais promissoras estão as estruturas de inferência de Common KADS (SCHREIBER et al., 2001), os padrões cognitivos de Gardner (GARDNER et al., 1998) e os Métodos de Solução de Problemas (FERNÁNDEZ-LÓPEZ; GÓMEZ-PÉREZ, 2002). Essas três abordagens têm em comum a proposição de uma representação diagramática de uma sequência abstrata de passos de inferência sobre conceitos do domínio.

### 3.2.6 Implementação da base de conhecimento

A fase de implementação é responsável por materializar as duas últimas etapas, Aquisição do conhecimento e Formalização do conhecimento adquirido, considerando o que foi definido na etapa de planejamento. Além dessa materialização, é comum também incluir a criação de uma interface gráfica para permitir a utilização do sistema, bem como a realização de testes e validações da base de conhecimento criada. Desta maneira, esta etapa permitirá ao usuário final verificar a consistência e corretude do conhecimento trabalhado no processo de modelagem.

É bastante comum nesta etapa de implementação, se utilizar de regras de produção. A regra de produção é uma das formas mais populares para a representação de conhecimento, também sendo usada nos sistemas automatizados de suporte à decisão. Com elas, o conhecimento é representado da forma de condição/ação, onde a condição é iniciada por uma cláusula SE (do inglês, *IF*) e a ação, também chamada de consequência, é iniciada por uma cláusula ENTÃO (do inglês, *THEN*).

Logo, **SE** condição (ou premissa ou antecedente) ocorre, **ENTÃO** ação (resultado, con-

clusão ou consequente) deverá ocorrer. Por exemplo, em um domínio relativo a trânsito de automóveis:

- **SE** o agente percebe luz do freio do carro em frente acesa, **ENTÃO** ele deve frear o carro — é um regra de ação;
- **SE** veículo tem 4 rodas **E** tem um motor, **ENTÃO** veículo é um automóvel — é uma regra que gera um novo conhecimento ou fato;

Nas próximas seções, serão apresentados mais detalhes das regras de produção, quais os problemas mais comumente resolvidos com essa representação, algumas alternativas às regras de produção para a resolução de problemas reais.

### 3.2.7 Avaliação e Refinamento da base de conhecimento

A fase de avaliação e refinamento da base de conhecimento é importante para identificar qual o ponto que deve ser trabalhado a fim de obter o melhor resultado para o problema em questão. Construir SBC de boa qualidade é um objetivo perseguido pelos que desenvolvem SBCs. O uso de métodos ao longo do desenvolvimento pode ser visto como uma forma de buscar essa qualidade e avaliar se a meta está sendo atingida e refinar a base de conhecimento quando necessário. SBC, por sua característica de permitir respostas incorretas ou com determinado grau de incerteza e também pelo seu comportamento, muitas vezes em desacordo com as expectativas do usuário, geraram uma série de estudos sobre o controle da qualidade. Foram, então, definidos alguns procedimentos gerais com tarefas e métricas específicas para esses sistemas (WERNECK, 1995).

A necessidade de serem feitos estudos e experimentos específicos para verificação, validação e refinamentos de sistemas baseados em conhecimento possibilitou o surgimento de um novo campo de pesquisa. As várias estruturas propostas de verificação e validação e os diversos sistemas desenvolvidos com esse propósito são um exemplo de pesquisas na área. Entretanto, trata-se de uma área ainda imatura, pois os termos verificação, validação, avaliação e testes têm diversas definições muitas vezes conflitantes além de ignorarem a terminologia empregada na engenharia de software (HOPPE; MESEGUER, 1993).

Oliveira (1995) realizou a revisão bibliográfica da avaliação da qualidade de sistemas baseados em conhecimento. Abordou-se nesse trabalho questões relativas a verificação e validação

de SBCs. A partir deste estudo, foi possível concluir que apesar de terem acontecido avanços significativos nesta área, ainda existe muito trabalho a ser realizado. Percebeu-se que as avaliações eram realizadas em geral qualitativamente ou quantitativamente, mas foi registrada também a necessidade de uma conceitualização maior, mais abrangente e concisa em relação as características de qualidade que devem estar presentes tanto na base de conhecimento como no processo de desenvolvimento desses sistema.

Sobre refinamento, Ginsberg (1988), Knauf et al. (2002) e Carbonara e Sleeman (1999) apresentaram em seus trabalhos envolvendo refinamento automático para bases de conhecimento que permitem reduzir o esforço do trabalho do engenheiro do conhecimento. Esses trabalhos indicam também que para a avaliação do SBC será testado pelo especialista e pelo usuário final para possíveis correções.

Na próxima seção, será abordada representação do conhecimento através de regras.

## 3.3 IMPLEMENTAÇÃO DO SISTEMA BASEADO EM REGRAS

Uma regra é um princípio ou regulamento que rege a conduta, ação, procedimento, acordo, etc. É uma declaração que define ou restringe algum aspecto do negócio; uma regra de negócios tem como objetivo afirmar a estrutura de negócios, controlar ou influenciar o comportamento da empresa. A importância das regras de negócios reside em sua capacidade de separar o conhecimento de sua lógica de implementação e de ser alterado sem alterar o código-fonte.

Muitas aplicações de negócios precisam lidar com as mudanças dinâmicas da economia de mercado. Por exemplo, as aplicações para uso nos setores bancário e de seguros devem ser capazes de acomodar as inevitáveis mudanças do mercado que ninguém pode prever ou planejar durante o design.

Cada regra representa um conhecimento de forma individual, ou seja, cada regra representa uma parte de um conhecimento independente. Por isto, é de suma importância que haja cuidado na manutenção da consistência quando se agrupam várias regras.

Ainda, conforme mencionado na seção 3.2.6, a representação do conhecimento através de regras é uma das técnicas mais populares na área. Além disso, também foi visto que essas regras baseiam-se em cláusulas SE-ENTÃO para resolução de um problema.

Por sua vez, essas regras podem ser categorizadas em dois tipos: regras de conhecimento ou declarativas; e regras de inferência ou procedurais. As regras do primeiro tipo, declarativas, oferecem uma descrição de fatos ou relacionamentos sobre o domínio da aplicação. Essas

regras são armazenadas pelo engenheiro do conhecimento na base de conhecimento. Já as regras do segundo tipo, procedurais (ou de inferência), auxiliam na resolução do problema. Essas outras regras são armazenadas no motor de inferência.

Para entender a importância da utilização de regras, há um conceito relacionado bastante importante: a resiliência. O termo resiliência é novo, e tem sua origem no latim no verbo Relisire, ou seja, "saltar para trás", "voltar ao estado natural" (CARMELLO, 2008).

> Resiliência é a capacidade de se antecipar — e se ajustar — continuamente a profundas tendências seculares capazes de abalar de forma permanente a força geradora de lucros de um negócio. A capacidade de reinventar modelos de negócios e estratégias de forma dinâmica à medida que mudam as circunstâncias (CARMELLO, 2008).

> A resiliência é a capacidade de o indivíduo lidar com problemas, adaptar-se a mudanças, superar obstáculos ou resistir à pressão de situações adversas - choque, estresse, algum tipo de evento traumático, entre outros. Sem entrar em surto psicológico, emocional ou físico, por encontrar soluções estratégicas para enfrentar e superar as adversidades. Nas organizações, a resiliência se trata de uma tomada de decisão quando alguém se depara com um contexto entre a tensão do ambiente e a vontade de vencer. Essas decisões propiciam forças estratégicas na pessoa para enfrentar a adversidade (WIKIMEDIA, 2020).

Kumar, Patil e Wadhai (2011) ratificam que mudanças são as únicas certezas que se pode ter. E também apontam que essas mudanças certamente são verdade para a lógica comercial das aplicações de software. Portanto, um software com propriedades resilientes é um objetivo a ser perseguido. Em complemento a isto, as alterações nos componentes que implementam a lógica de negócios de um aplicação podem ser necessárias por vários motivos:

- Para corrigir defeitos de código encontrados durante o desenvolvimento ou após a implantação;
- Para acomodar condições especiais que a lógica de negócios deve levar em consideração, e que não foram mencionadas inicialmente pelo cliente;
- Para lidar com alterações nos objetivos de negócios de um cliente;
- Para estar em conformidade com o uso processos ágeis e/ou de desenvolvimento interativo;

Dados esses motivos, uma aplicação que suporta alterações na lógica de negócios sem grandes complicações é altamente desejável — ainda mais se o desenvolvedor que precisar

realizar as alterações na lógica não for a pessoa que escreveu o código inicial. Uma solução para isto é a separação entre conhecimento e código, mantendo-se as regras em uma base de conhecimento declarativo e o código executável na máquina de inferência que executa essas regras. O mecanismo de regras avalia e executa essas regras conforme definidas pelos especialistas sem afetar a forma como a aplicação é executada. A aplicação é, então, criada para lidar com essas regras, mas projetadas separadamente do mecanismo.

Observando pela ótica da Engenharia de Software, pode-se considerar como um dos grandes benefícios de um motor de inferência permitir externalizar a lógica de negócios representadas em um SBC e facilitar a manutenção do software que as utiliza. Um motor de inferência pode ser visto como um intérprete sofisticado de instruções se-então. As instruções se-então são as regras.

Uma forma comum de implementação de uma aplicação utilizando um motor de inferência é ter como entrada uma base de regras, chamada conjunto de execução de regras, e os dados de entrada a serem classificados. Neste cenário, as saídas podem incluir os dados de entrada originais com modificações, a criação de novos dados e possíveis efeitos colaterais no comportamento da aplicação.

Uma vez as regras de negócio externalizadas da aplicação, é possível alterá-las de modo muito rápido. Portanto, programas de computadores que tenham este tipo de necessidade, são indicados que utilizem motores de regras e regras para implementação da sua camada de negócio. Desta forma, até usuários finais poderão alterar as regras da aplicação.

### 3.3.1 Funcionamento de um Sistema Baseado em Regras

Vimos anteriormente como um conhecimento pode ser representado através de regras. Além de tornar a alteração das regras de negócios uma atividade fácil e rápida, um mecanismo de regras é uma ótima ferramenta para a tomada de decisões eficiente, porque permite tomar decisões com base em milhares de fatos de maneira rápida, confiável e repetida. Há um tópico relevante sobre o mecanismo de regra que é a compreensão do seu funcionamento e, para isto, é importante registrar algumas definições utilizadas.

A primeira definição já citada anteriormente é a de **regra** como representação do conhecimento na forma de condição-consequência, se-então. A segunda definição básica é de **sentença**. Informalmente, uma base de conhecimento é um conjunto de sentenças, podendo ser definida como a seguir.

> Informalmente, uma base de conhecimento é um conjunto de sentenças (aqui, "sentença" é utilizada como um termo técnico; está relacionada mas não é idêntica às sentenças em português e em outros idiomas ou linguagens naturais). Cada sentença é expressa em uma linguagem de representação de conhecimento e representa alguma asserção sobre o mundo. Às vezes, ilustramos uma sentença com o nome axioma, quando a sentença for dada sem ser derivada de outras sentenças (RUSSELL; NORVIG, 2014).

Outro importante conceito é o de **modelo**. Um modelo é o conhecimento de "como o mundo funciona" — seja ele implementado em circuitos *booleanos* simples ou em teorias científicas complexas.

Além desses conceitos, também será necessário o entendimento sobre **inferência lógica** ou ilação. Inferência é a operação intelectual mediante a qual se afirma a verdade de uma proposição em decorrência de sua ligação com outras proposições já reconhecidas como verdadeiras. Também, inferência é o raciocínio através do qual uma proposição é considerada verdadeira pela sua ligação com outras já tidas como verdadeiras; a proposição que se assume como sendo verdadeira. Consiste, portanto, em derivar conclusões a partir de premissas conhecidas ou decididamente verdadeiras.

Um mecanismo de inferência com regras pode funcionar através de algoritmos de encadeamento para a frente, ou progressivo, e encadeamento para trás, ou regressivo. Ambos os algoritmos são naturais, por isso, as etapas de inferência são óbvias e fáceis de os seres humanos seguirem. Por sua vez, essa inferência é a base para a programação lógica.

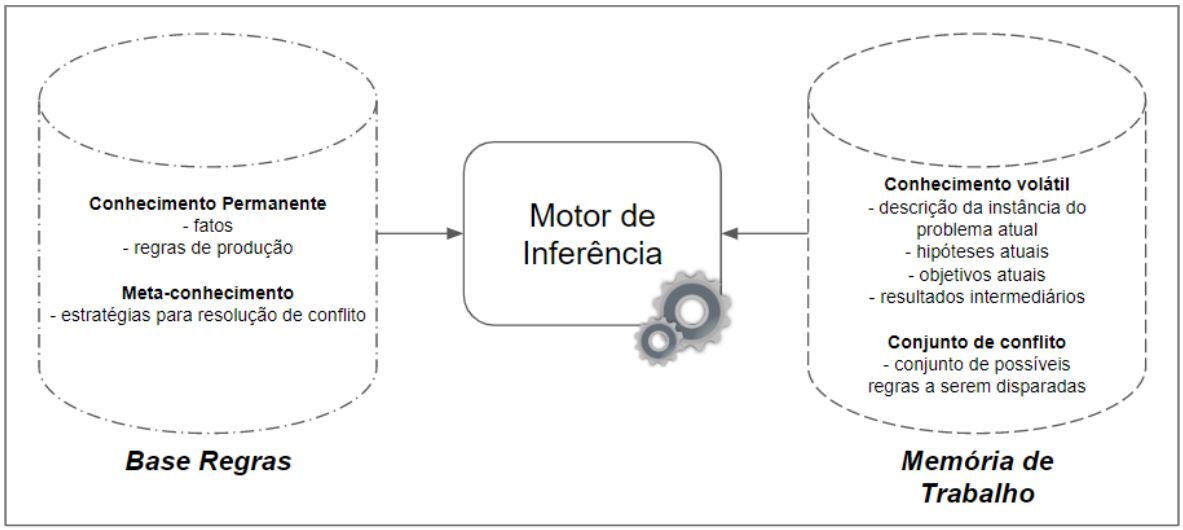
Antes de passar para o aprofundamento dos algoritmos de encadeamento, então, podemos ver na Figura 7 uma arquitetura genérica de um sistema de produção.

Nela, podemos ver a existência de duas bases. A primeira é base de regras em que o conhecimento inicial sobre o domínio foi registrado em forma de fatos, regras de produção e algumas estratégias para resolução de conflitos entre essas regras. A segunda base a memória de trabalho, onde fica registrado o conhecimento volátil com as instâncias atuais das variáveis, hipóteses, objetivos e resultados intermediários.

### Encadeamento progressivo

Heinzle et al. (1995) explicam que o funcionamento do encadeamento progressivo (do inglês, *forward chaining*) começa com a seleção de uma regra qualquer da base de conhecimentos. Simultaneamente, é criada a lista de verdades onde serão armazenadas todas as

Figura 7 – Arquitetura genérica de Sistemas de Produção




**Fonte:** Elaborada pelo autor (2020)

informações tidas como verdadeiras. No início do processo de inferência, alguns questionamentos serão obrigatórios já que a lista de verdades está ainda vazia.

As respostas a esses questionamentos são então colocadas na lista de verdades. No encadeamento progressivo, também chamado encadeamento dirigido por dados, o antecedente (*IF*) da regra é comparada com a descrição da situação atual, contida na memória de trabalho. As regras que satisfazem a esta descrição são executadas (disparadas), e os fatos contidos no consequente das regras serão incluídos na memória de trabalho (PY, 2009).

Após a utilização de uma regra, as conclusões desta regra também devem ser adicionadas à lista de verdades. Na sequencia, o sistema procura por uma regra que tenha entre as suas premissas a conclusão da regra avaliada. Este encadeamento que vai da conclusão de uma regra para as premissas de outra regra é chamado de encadeamento progressivo. Um sistema real deve considerar a existência de caminhos alternativos ao longo do processamento, inclusive com um mecanismo que possibilite retomo ou retrocesso, chamado de "*backtracking*", quando necessário.

### Encadeamento regressivo

Para o funcionamento do encadeamento regressivo (do inglês, *backward chaining*), Heinzle et al. (1995) afirma que consiste em partir da hipótese que se quer provar, procurando regras na base de regras cujo consequente satisfaz essa hipótese. Ou seja, ele usa as regras da base

para responder a perguntas ou provar se uma assertiva é verdadeira. Para isto, o encadeamento regressivo apenas processa as regras relevantes para a pergunta, sendo, em geral, mais rápido e eficiente do que o encadeamento progressivo para esta problemática. É, portanto, a forma mais amplamente utilizada de raciocínio automatizado.

No encadeamento regressivo, também chamado encadeamento dirigido por objetivos, o comportamento do sistema é controlado por uma lista de objetivos. Um objetivo pode ser satisfeito diretamente por um elemento da memória de trabalho, ou podem existir regras que permitam inferir algum dos objetivos correntes, isto é, que contenham uma descrição deste objetivo no seu antecedente.

Também é criada, neste método, a lista de verdades que conterá todas as informações tidas como verdadeiras. Esta lista, além das respostas fornecidas diretamente pelo usuário, engloba também as conclusões do próprio sistema.

### 3.3.2 Quando e onde utilizar sistemas baseados em regras

Nem todas as aplicações devem utilizar sistemas baseados em regras. Entretanto, se o código da lógica de negócios incluir várias instruções se-então, é interessante considerar usar um. A manutenção de uma lógica *booleana* complexa pode ser uma tarefa difícil, e um mecanismo de regras pode ajudá-lo a organizar essa lógica e tornar o processo mais fácil.

As alterações num programa são significativamente menos propensas a introduzir erros quando você pode expressar a lógica usando uma abordagem declarativa em vez de uma linguagem de programação imperativa. O problema é que as linguagens que dão suporte a este tipo de abordagem possuem mecanismos rudimentares, sistemas de tipo muito simples, dentre outras questões que terminam dificultando a construção do sistema.

Também devemos considerar um sistemas baseados em regras se as alterações no código puderem causar grandes perdas financeiras. Muitas organizações têm regras rigorosas sobre a implantação de código compilado em seus servidores. Por exemplo, se você precisar modificar a lógica em uma classe Java (ARNOLD et al., 2000), geralmente um longo e tedioso processo deve ocorrer antes que a alteração chegue ao ambiente de produção.

Mesmo uma simples alteração em uma linha de código pode custar milhares de reais a uma organização. Se for preciso seguir essas regras rígidas e fazer alterações frequentes no código de lógica de negócios, é bastante interessante considerar separar o conhecimento, a lógica de negócios, do código com a utilização de um mecanismo de regras. Mecanismos de regras são

usados em aplicações para substituir e gerenciar parte da lógica de negócios.

Eles são mais usados em aplicações em que a lógica de negócios é dinâmica demais para ser gerenciada no nível do código-fonte — ou seja, onde uma mudança em uma política de negócios precisa ser imediatamente refletida na aplicação. Logo, aplicações em domínios como seguradoras, serviços financeiros, governamentais, atendimento e cobrança de telecomunicações por clientes, o comércio eletrônico, e assim por diante, se beneficiam bastante do uso de mecanismos de regras.

O conhecimento sobre o cliente também pode ser um fator nessa decisão de escolher se deve-se utilizar de regras para representar a camada de negócios. Mesmo trabalhando com um conjunto simples de requisitos e permitindo uma implementação direta no código, se o cliente tiver que adicionar e alterar requisitos de lógica de negócios com frequência, durante o ciclo de desenvolvimento e mesmo após a implantação, será melhor se utilizar de um mecanismo de regras desde o início.

Além do que já foi mencionado anteriormente, também se pode listar como vantagens de adotar uma abordagem baseada em regras:

- Regras que representam políticas são facilmente comunicadas e compreendidas.
- As regras mantêm um nível mais alto de independência do que as linguagens de programação convencionais.
- As regras separam o conhecimento de sua lógica de implementação.
- As regras podem ser alteradas, adicionadas ou removidas sem alterar o código-fonte; portanto, não há necessidade de recompilar o código da aplicação.
- O custo de produção e manutenção diminui.

Os mecanismos de regras de negócios geralmente constituem componentes de software independentes, separados do restante da aplicação. Isso permite processar e modificar uma base de conhecimento independentemente dos outros módulos da aplicação. Esses sistemas podem ser adaptados ao problema específico do domínio, desenvolvendo uma base de conhecimento adequada.

No entanto, como já foi visto no início desta seção, o desenvolvimento da base de conhecimento não é uma tarefa trivial. Para construir uma base de conhecimento de alta qualidade, alguns mecanismos, como métodos eficientes de *design* ou verificação, devem ser utilizados.

Existem técnicas que permitem verificar a base de conhecimento de acordo com a demanda do usuário. No entanto, a correção de alguns erros na base de conhecimento pode introduzir outros erros. Portanto, para fornecer melhor eficiência de verificação, a base de conhecimento deve ser monitorada continuamente durante todo o projeto.

Na próxima seção, será mostrada uma ferramenta de processamento de regras, a ferramenta Drools, e o seu funcionamento com um pouco mais de detalhe. Também será mostrado quais benefícios da utilização do Drools e a justificativa da utilização do Drools neste trabalho.

## 3.4 DROOLS: UMA FERRAMENTA PARA PROCESSAMENTO DE REGRAS

A ferramenta Drools (BROWNE, 2009; BALI, 2009) é um mecanismo que oferece inferência baseada em regras, de código aberto, escrito na linguagem Java e que usa o algoritmo Rete (FORGY, 1989) para avaliar as regras. A implementação do Drools Rete é chamada ReteOO (SOTTARA; MELLO; PROCTOR, 2010), em que a Drools possui uma implementação aprimorada e otimizada do algoritmo Rete para sistemas orientados a objetos.

### 3.4.1 Por que utilizar o Drools?

O mecanismo Drools permite expressar regras de negócios de maneira declarativa. Podemos escrever regras usando uma linguagem nativa não-XML que é bastante fácil de aprender e entender. E podemos incorporar o código Java diretamente em um arquivo de regras de extensão ***.drl**, as regras DRL. O Drools também tem outras vantagens, tais como:

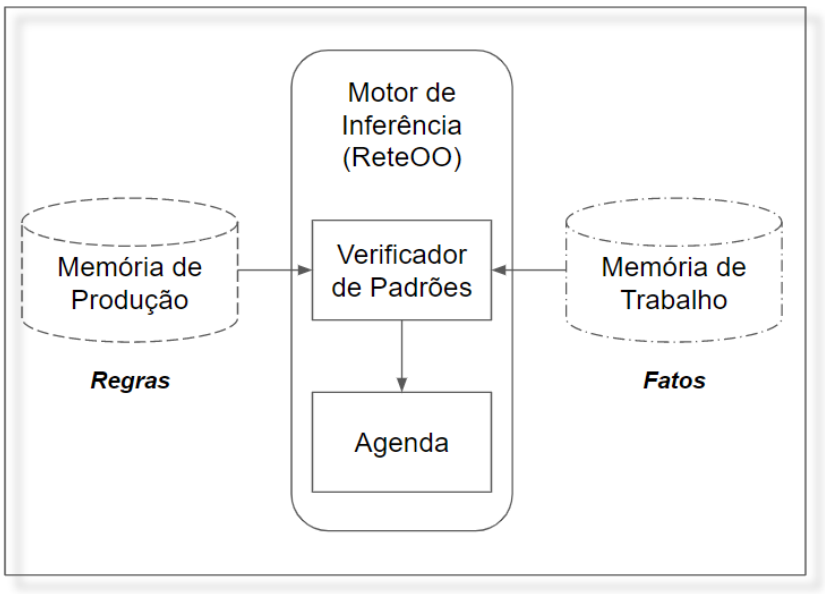
- Ser suportado por uma comunidade ativa;
- Ser fácil de usar;
- Ser rápido de executar;
- Ter ganhado popularidade entre desenvolvedores Java;
- Ser compatível com a API do *Java Rule Engine* - JSR 94;
- Ser software livre.

### 3.4.2 Como funciona o Drools?

O Drools usa a abordagem baseada em regras para implementar um sistema basado em conhecimento, por isso, é mais corretamente classificado como um sistema de regras de produção. Um sistema de regras de produção é Turing completo (LEWIS; PAPADIMITRIOU, 1998), com foco na representação do conhecimento para expressar lógica proposicional e de primeira ordem de maneira concisa, não ambígua e declarativa.

O cérebro de um sistema de regras de produção é um motor ou mecanismo de inferência (Figura 8). O motor de inferência combina fatos e dados com as regras de produção para inferir conclusões, e é capaz de escalar para um grande número de regras e fatos.

Figura 8 – Visualização de alto nível de um mecanismo de regras implementado pelo Drools




**Fonte:** JBOSS (2011, tradução livre)

Ainda na Figura 8, é possível observar que as regras são armazenadas na memória de produção e os fatos que o mecanismo infere corresponde à memória de trabalho. Os fatos são declarados na memória de trabalho, onde podem ser modificados ou deletados.

Um sistema com um grande número de regras e fatos pode resultar em muitas regras verdadeiras para a mesma afirmação de fato, essas regras são caracterizadas como **em conflito**. Por sua vez, a Agenda gerencia a ordem de execução dessas regras conflitantes usando uma estratégia de resolução de conflitos. O mecanismo de inferência de um sistema de regras de produção é capaz de manter a verdade. Isto é, todo fato inferido a partir de fatos verdadeiros também será verdadeiro — chamada preservação da verdade (JBOSS, 2011).

Tizzei et al. (2007) mostram que o Drools utiliza o algoritmo de encadeamento para frente em seu motor de inferência. Desta forma, o processo de inferência começa a partir da inserção dos fatos iniciais na memória de trabalho. Em seguida é iniciado o processo de verificação das regras contidas na base de conhecimento, cada regra contendo uma condição e uma ação a ser executada. As regras podem, durante a execução, inserir novos fatos na memória de trabalho, possibilitando o acionamento de outras regras.

## 3.5 CONSIDERAÇÕES FINAIS

Princípios, técnicas e ferramentas, desenvolvidas para a engenharia do conhecimento, podem ser utilizados para ajudar as instituições na gestão do conhecimento. A engenharia do conhecimento lida com o conhecimento nas últimas décadas, tanto do ponto de vista teórico quanto prático, e construiu princípios, técnicas e ferramentas que têm um bom histórico em uma ampla gama de aplicações. Há uma sobreposição substancial entre as preocupações e os problemas encontrados na engenharia do conhecimento, porque, em muitas áreas, está-se tentando fazer a mesma coisa: usar a tecnologia para **transmitir** o conhecimento de um especialista em um domínio para aqueles que não são especialistas.

Ainda neste capítulo, foram vistas as definições e tipos de conhecimento, alguns princípios da engenharia do conhecimento. Então, foi mostrado um processo para utilização da engenharia do conhecimento e suas etapas. Em seguida, foi mostrado como é realizada a representação do conhecimento através de regras, bem como, foi apresentada a Drools como ferramenta de processamento de regras.

No próximo capítulo, será mostrado um problema real de uma empresa real de teste de software. O domínio e o problema serão contextualizados. Após isto, uma solução será proposta utilizando a engenharia do conhecimento baseada em regras e utilizando a ferramenta de processamento de regras Drools.

# 4 DRL TC CLASSIFIER

Antes de contextualizar o problema real de uma empresa de teste de *software* para dispositivos móveis, faz-se necessário introduzir a importância do teste de software. Neste capitulo, será introduzido o conceito de teste de *software* e a importância da utilização dessa área na engenharia de *software*. Também será descrito o problema identificado nesta área e, em seguida, será apresentada uma solução proposta e como foi realizada sua validação. Por fim, as considerações finais sobre o trabalho descrito neste capítulo.

## 4.1 TESTE DE *SOFTWARE*

O teste de *software* é um conjunto de processos organizados para garantir que o sistema ou aplicação faça o que foi projetado para fazer e que não faça nada não intencional. O *software* deve ser previsível e consistente, sem oferecer surpresas aos usuários (MYERS; SANDLER; BADGETT, 2011).

«[rightmargin=0cm,leftmargin=4cm] [...] em um mundo ideal, gostaríamos de testar todas as permutações possíveis de um programa. Na maioria dos casos, no entanto, isso simplesmente não é possível, mesmo um programa aparentemente simples pode ter centenas ou milhares de combinações possíveis de entrada e saída. Criar casos de teste para todas essas possibilidades é impraticável. O teste completo de um aplicativo complexo levaria muito tempo e exigiria muitos recursos humanos para ser economicamente viável. (MYERS; SANDLER; BADGETT, 2011). (tradução livre) »

Em complemento, Parveen, Tilley e Gonzalez (2007) ainda afirmam que o teste de *software* é um processo para identificar a corretude, integridade e qualidade do *software* de computador desenvolvido. Ele abrange os conceitos de demonstração da validade do *software* em cada estágio do ciclo de vida de desenvolvimento e a autenticidade do sistema final em relação aos requisitos do cliente (PERRY, 2007).

O teste de software é uma área essencial da engenharia de *software*, mas frequentemente subutilizada. Uma grande variedade de técnicas de teste de *software* foi desenvolvida para identificar efetivamente erros, mas essas técnicas nem sempre são totalmente empregadas na prática (PARVEEN; TILLEY; GONZALEZ, 2007). Existem inúmeras razões para isso, incluindo a dificuldade em dominar a complexidade do gerenciamento de todos os casos de teste para projetos de grande escala.

O objetivo do teste é identificar com eficiência quaisquer desvios de alto impacto dos resultados esperados, para que esses desvios possam ser corrigidos ou evitados antes do lançamento do *software*. O valor do teste também abrange objetivos como a redução do número de defeitos e o aumento da produtividade do desenvolvimento por meio de *feedback* contínuo. Portanto, pode-se entender o teste de *software* como uma coleção de técnicas usadas no processo de verificação de um *software* para garantir que ele tenha um nível de qualidade aceitável para sua utilização.

Harrold (2000) menciona pesquisas que mostram que garantir a qualidade do *software* consome mais de 50% do custo do *software* e esse número é ainda mais alto para sistemas críticos de segurança. Apesar da grande e reconhecida relevância, em projetos de desenvolvimento de aplicativos, o teste geralmente não recebe recursos suficientes de tempo e prioridade até que o desenvolvimento seja concluído.

Com a pressão competitiva e o aumento do custo do tempo para lançar um produto, algumas organizações começaram a introduzir testes nos estágios iniciais do desenvolvimento de *software*, enquanto outras se esforçam para encontrar estratégias eficazes de teste. Porém, poucas organizações estabelecem critérios para medir a eficácia dos testes. Sem padrões de teste e uma estratégia de gerenciamento de teste adequada, a eficácia do teste não pode ser medida ou aprimorada.

O gerenciamento de casos de teste, por sua vez, envolve a organização sistemática de artefatos de teste, por exemplo, dados de rastreabilidade de requisitos, casos de teste e resultados esperados. Para ter sucesso, o gerenciamento de casos de teste requer um alto grau de disciplina por parte da equipe de testadores para acomodar o grande volume de artefatos.

Há também diversas medidas que podem ser tomadas para se obter testes melhores, mais ágeis, efetivos e baratos. Dentre estas, podemos citar: sistematizar as atividades de testes; definir os papéis e responsabilidades vinculadas ao teste; antecipar a preparação dos testes já durante as fases de construção do *software*; conhecer as técnicas relacionadas ao tema; testar características diferentes do *software* nas diferentes fases de testes; estimar adequadamente os esforços para realização dos testes; ter um controle efetivo das falhas encontradas; e automatizar os processos de testes, tanto planejamento quanto execução (COSTA et al., 2006).

Automatizar testes significa de forma simplória fazer uso de *softwares* que reproduzam os passos dos casos de teste. O uso desta prática pode reduzir o esforço necessário para a realização das atividades de teste em um projeto de *software*, ou seja, executar maiores quantidades de testes em tempo reduzido. Testes manuais que levariam horas para serem totalmente

executados poderiam levar minutos caso fossem automatizados (TUSCHLING, 2008).

Apesar dos testes manuais também encontrarem erros em uma aplicação, esse é um trabalho desgastante que pode consumir muito tempo. Automação de testes possibilita a execução de mais testes, com critérios mais explícitos, em menor tempo e com menor esforço de execução (COSTA et al., 2006). Além disso, é importante citar as inúmeras possibilidades que um *script* automático traz, como: facilidade de atender novos requisitos, aumentando a cobertura dos testes (extensibilidade), além de ser robusto quanto à quantidade de dados de entrada que podem ser processadas (confiabilidade). Outras vantagens que podem ser destacadas ao transformar suas rotinas de testes em *scripts* automáticos são: garantia de qualidade, reusabilidade e manutenibilidade (BERNARDO; KON, 2008), (COSTA et al., 2006).

Apesar da automação ter o potencial de trazer todos os benefícios citados, nem sempre ela é viável devido ao seu alto custo de esforço e tempo. Além disto, em alguns casos como, por exemplo, em casos de teste que mudam com frequência, o esforço de automação não compensa.

A partir daqui, trataremos de teste de *software* e casos de teste como conceitos fundamentais na etapa de desenvolvimento de *software* de qualidade. Na próxima seção será contextualizada a problemática tratada neste trabalho relativa a área de teste de *software* e a automação de casos de teste.

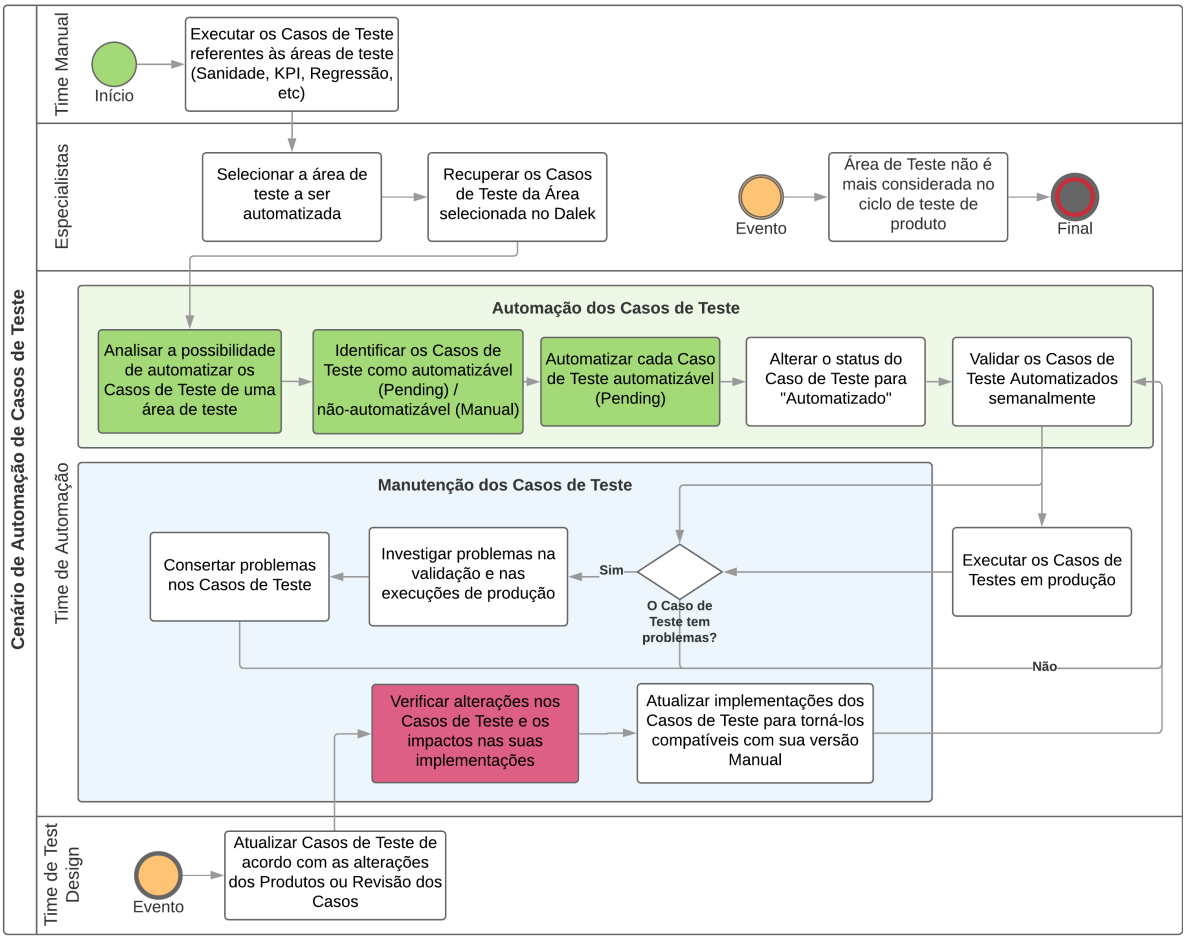
## 4.2 CONTEXTUALIZAÇÃO DO PROBLEMA

Como dito anteriormente, este trabalho de mestrado está relacionado diretamente a uma problemática na área de testes de *software* e à atividade de automação de casos de teste. Nesta seção, será descrito o cenário real e atual numa empresa de testes para dispositivos móveis. Este cenário que será detalhado pode ser verificado no processo para automação de casos de teste, na Figura 9.

Nesta empresa real, há equipes de execução de casos de teste manuais para diversos tipos de testes descritos na engenharia de testes. Dentre esses tipos, pode-se citar, por exemplo, testes de sanidade (ZYBIN et al., 2008), testes exploratórios (KANER, 2006), testes de *Key Performance Indicator* (KPI) (MARR; SCHIUMA; NEELY, 2004), testes de regressão (AIKEN; WEST; RENO, 1991), testes de aplicações de terceiros (COUNCILL, 1999; MA et al., 2001), testes de conformidade (GANDHI; KANG, 2002), entre outros.

Além dessas equipes, há outros times de execução de casos de teste automáticos. Esses

Figura 9 – Processo para automação de casos de teste



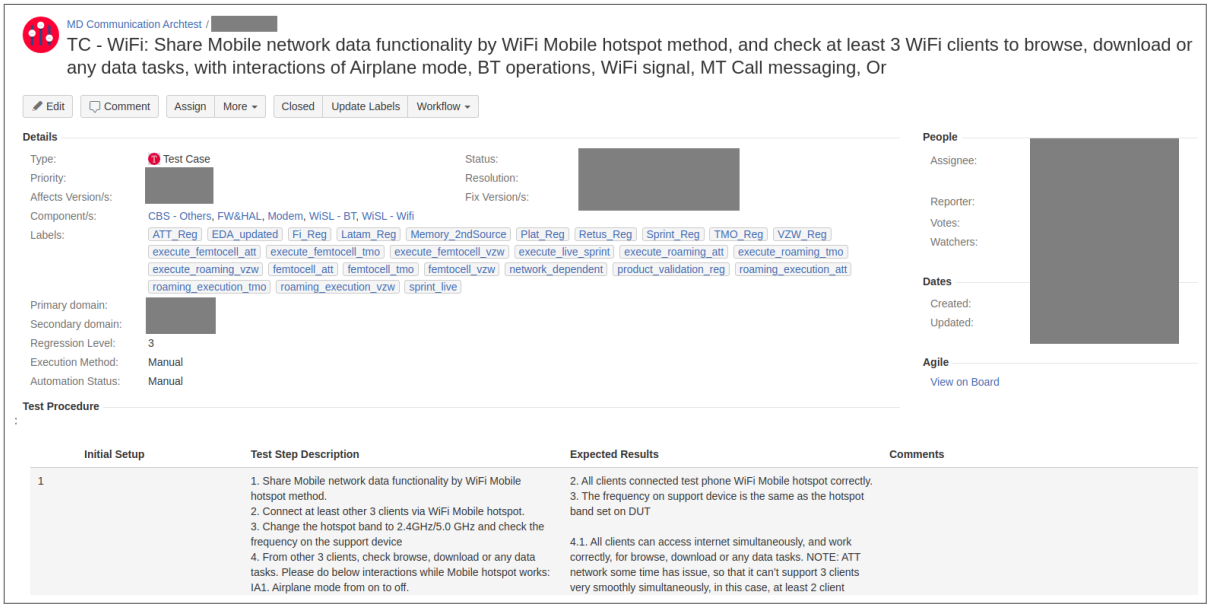

**Fonte:** Elaborada pelo autor (2020)

casos podem ser: exclusivamente automáticos, ou seja, não podem ser executados por um ser humano; ou casos automatizados que poderiam ser executados pelo time de manual, mas que uma vez transformado em *script* permite um controle melhor de qualidade, aumenta a produtividade da equipe de manual, permite o melhor aproveitamento dos recursos das equipes, etc.

Essas equipes, de execução de testes manuais e automáticos, estão divididas em áreas de testes correspondentes aos tipos de testes e essas divisões, também chamadas de sub-times, executam os casos de teste respectivos de forma manual. Esses casos de teste, executados pelos times correspondentes, são organizados em ciclos ou planos de teste. Tanto estes casos de teste quantos os planos de teste são gerenciados em um sistema fornecido pela Atlassian (2020): o Dalek. Esses casos estão armazenados na base da empresa de teste no formato mostrado na Figura 10.

Para cada tipo de teste e sua área, existe um planejamento no ciclo de vida do produto

Figura 10 – Caso de teste presente na base de testes




**Fonte:** Elaborada pelo autor (2020)

onde a quantidade de testes, o momento e frequência da execução dos ciclos de teste variam bastante. Por exemplo, há ciclos com dezenas de casos testes que são executados diariamente durante todo o ciclo de vida do produto.

Há também planos de testes que contam com centenas de casos testes e que são executados apenas uma ou duas vezes no ciclo de vida de um produto. Outro caso são os ciclos de testes que são executados algumas vezes mas apenas na fase final do ciclo de vida de um produto. Essa informação sobre a quantidade de casos de teste no ciclo, a fase e a frequência de execução dos ciclos é de grande importância na priorização dos casos de teste em relação a sua automação.

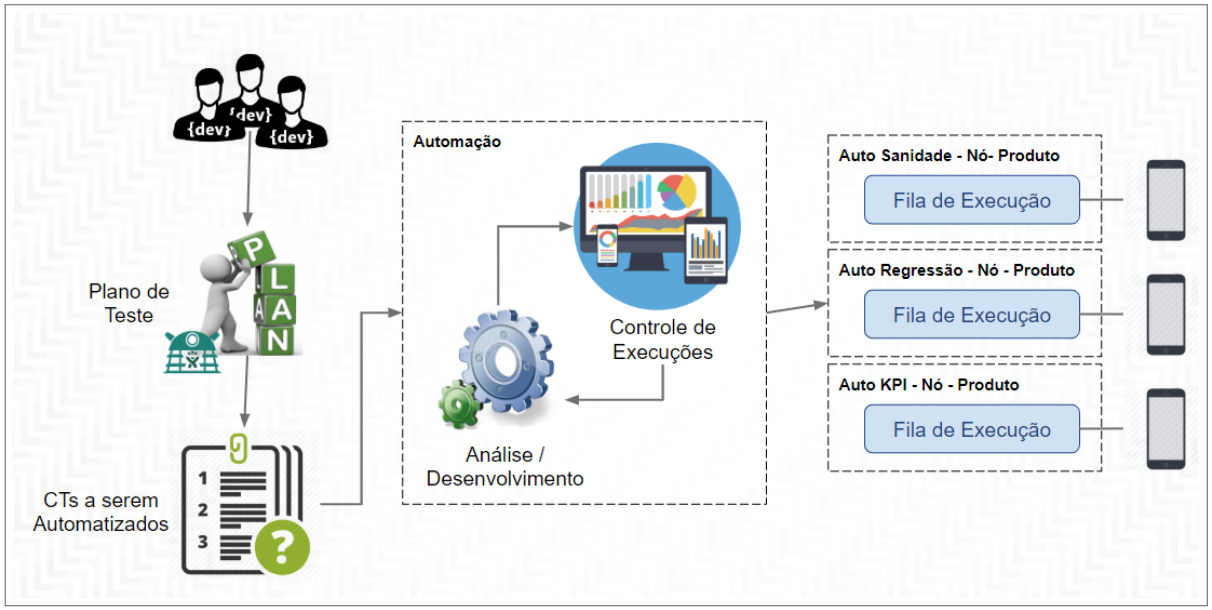
### 4.2.1 Automação de testes

A fim de aumentar a quantidade de testes executados e a qualidade do processo de teste, é selecionada uma área de testes para ser automatizada completa ou parcialmente. Para isto, é necessário recuperar todos os casos de teste relativos à área escolhida. Em seguida, através do conhecimento de especialistas, realiza-se uma avaliação inicial sobre a possibilidade de automação dos casos de teste, e em caso afirmativo, realiza-se uma avaliação individualizada de cada caso de teste. Se a avaliação inicial indicar que não é possível ou viável realizar a automação dos casos da área, ela é desconsiderada e uma nova área é selecionada para

análise.

Essa atividade de avaliação é registrada inicialmente em planilhas da área de teste sendo analisada para registrar os motivos que levaram um caso de teste a ser classificado como automatizável ou não automatizável. Após a avaliação detalhada dos casos de teste em relação a sua possibilidade de automatização, os testes são marcados na ferramenta Dalek como MANUAL, quando o caso de teste não é automatizável, ou como PENDING, quando essa possibilidade existe. Futuramente, depois de automatizado, o caso de teste é marcado como AUTOMATED. Entretanto, apenas a classificação fica registrada no sistema institucional e as justificativas detalhadas da análise, que foram registradas em planilhas, não são compartilhadas entre os sub-times de áreas distintas.

Figura 11 – Processo genérico de automação de casos de teste na empresa parceira para dispositivos móveis




**Fonte:** Elaborada pelo autor (2020)

Como pode ser visto na Figura 11, os casos de teste já automatizados são periodicamente executados em servidores (também chamados de nós), a fim de verificar se sua implementação é compatível com o comportamento descrito no caso de teste manual e com produto sendo testado. Além dessa verificação considerando o caso de teste automatizado, ainda é verificado, também periodicamente, se houve uma atualização na descrição ou no resultado esperado do caso de teste manual. Em caso positivo, o caso de teste é marcado como PENDING no Dalek até que seja verificada a correspondência do caso de teste escrito com sua versão automatizada. Só após isto, ou até de um possível retrabalho em sua automação, é que o caso de teste pode ser etiquetado como AUTOMATED.

## 4.3 PROBLEMA IDENTIFICADO

Dado o cenário apresentado, é possível identificar alguns pontos sensíveis no processo que são problemas reais nesta atividade de automação de casos de teste para dispositivos móveis. Por exemplo, há uma diversidade de casos de teste inerente às áreas de testes que exige que um tempo variado para execução desses testes. Isso deve ser um ponto considerado para a análise e automação dos casos de teste.

Outro ponto bastante importante é que como os testes são relativos às diversas áreas de testes, diversos times escrevem e mantêm esses casos no repositório. Esses times muitas vezes são de origem e culturas diferentes, e apesar de todos os CT serem escritos em inglês, a diversidade dos times traz dificuldades para se manter uma padronização na escrita desses casos de teste.

Como foi visto na seção anterior, a área de teste de *software* é uma das áreas da engenharia de software que garante a qualidade dos produtos desenvolvidos. É óbvio que para isto, o produto precisa estar desenvolvido parcial ou completamente. Com o mercado ávido por novos produtos, muitas vezes, o tempo para realização de testes sofre com os prazos enxutos e impactados pelos atrasos das etapas anteriores do desenvolvimento dos produtos.

Também é sabido que, para alguns tipos de testes, sua execução se torna cada vez menos efetiva para a identificação de falhas à medida que as versões do *software* evoluem e as correções de falhas são realizadas. Para isto, é importante manter uma visão sobre a efetividade dos casos de teste e atualizá-los de forma a variar as áreas exercitadas e os resultados esperados. Este ponto também impacta diretamente a classificação dos casos de teste, pois um novo caso de teste ou um caso de teste atualizado precisa ser verificado em relação à compatibilidade da sua versão automática.

Além dos problemas já mencionados, como os dispositivos móveis recebem atualizações periodicamente, é possível que os casos de teste criados para, ou compatíveis com, as versões de softwares não sejam mais válidos ou precisem de atualizações para suportar essas novas versões. Este ponto tem impacto diretamente na classificação dos casos de teste de forma similar ao que foi dito anteriormente.

Para este trabalho, focaremos na problemática de identificar se um caso de teste novo ou atualizado é passivo de automação considerando o *framework* institucional de automação de casos de teste, ambiente de execução dos casos de teste e o custo-benefício da automação de cada caso. Como foi explicado brevemente, para realização dessa análise, é preciso considerar

algumas questões:

1. Capacidade do framework para realizar todos os passos descritos no caso de teste;
2. Ambiente de execução exigido no caso de teste:
    - operadora de telefonia móvel;
    - acessório;
    - nível de estabilidade do ambiente;
    - unidades móveis para teste;
    - etc.
3. Custo de manutenção do caso de teste automatizado;
4. Versão do sistema operacional;
5. Custo-benefício de automatizar o caso de teste vs. executá-lo manualmente.

Considerando os itens na lista acima, é possível notar que há questões complexas a analisar para se determinar se um dado teste pode ser automatizado. Assim, esta análise deve ser realizada por um engenheiro experiente, um especialista, com o conhecimento suficiente para considerar todas esses cenários.

Dentre as considerações que o especialista precisa estar atento, uma importante é que os cenários também podem mudar. Por exemplo, é possível que alguma funcionalidade não suportada pelo *framework* tenha sido implementada por um time diferente. Também é possível, por exemplo, que uma operadora telefônica passe a dar suporte a um tipo de rede que não era dado anteriormente. Ou seja, para cada item da lista, é possível imaginar inúmeros cenários, mas é impossível prever todos.

Então, visando evitar esse custo de disponibilizar um especialista experiente com todo o conhecimento, treinamento, manutenção e compartilhamento dos registros desse conhecimento de forma mais estruturada, este trabalho se dedicou a resolver ou melhorar as dificuldades enfrentadas pelo time de automação de casos de teste, que precisa identificar quais casos de teste podem ou não ser automatizados.

## 4.4 SOLUÇÃO PROPOSTA

Nesta seção, será vista a solução proposta para problemática da classificação de testes em relação a sua possibilidade de automação, ou seja, verificar sua automatabilidade. Visando abarcar o conhecimento dos especialistas, e não simplesmente cobrir o que já é possível realizar pelo *framework* atual de automação de casos de teste, faz mais sentido registrar as características que impedem a automação de um caso de teste para dispositivos móveis. Assim, as regras propostas neste trabalho indicam se um caso de teste deve ser executado manualmente. Caso contrário, esse caso de teste é considerado como automatizável.

A necessidade de mapear as regras considerando o que não pode ser automatizado se torna mais evidente quando se considera a renovação e criação de *features* nos produtos sob teste versus a evolução do *framework* de automação de casos de teste. Ou seja, antes de serem implementadas as novas funcionalidades e mapeadas telas no *framework*, é necessário avaliar se os casos de teste relativos podem ser ou não automatizados. Caso contrário, as regras sobre possibilidade de automatização estariam sempre defasadas em relação às novas *features* e seriam criadas reativamente.

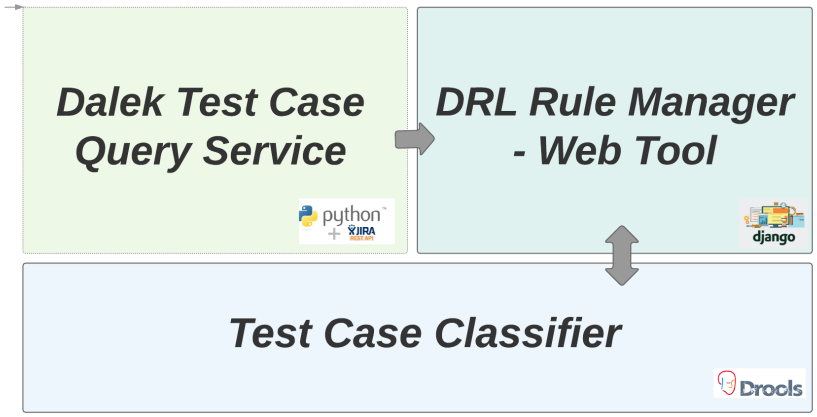
Nesse contexto, foi implementada uma solução utilizando regras DRL (Capítulo 3 - Seção 3.4) para classificação dos casos de teste. Na Figura 12, é possível verificar a existência de três módulos principais: o primeiro, um serviço de recuperação de casos de teste; o segundo, um serviço de classificação de casos de teste; e o terceiro, um sistema web que gerencia os outros dois serviços e permite a utilização e atualização deles. Nas próximas seções, serão detalhados como foi o processo de criação das regras e os módulos implementados.

### 4.4.1 Processo de criação das regras

Como foi dito anteriormente, no processo de classificação de testes em relação a possibilidade de sua automação, os especialistas analisam casos de teste para dispositivos móveis. Esses casos estão armazenados na base da empresa de teste no formato mostrado na Figura 10. Neste trabalho, o autor desta dissertação foi considerado tanto como engenheiro do conhecimento, que identificou e registrou as regras, quanto um dos especialistas da área de automação de testes.

Também foi visto que como resultado dessa análise, são geradas planilhas por área de testes ou sub-times onde podem ser encontradas as informações relevantes à classificação,

Figura 12 – Processo para classificação de casos de teste para dispositivos móveis em relação a sua possibilidade de automação




**Fonte:** Elaborada pelo autor (2020)

tais como, o identificador do caso de teste, a classe atribuída na classificação e a justificativa dessa decisão. Na Figura 13 é mostrado um exemplo de planilha criada neste tipo de análise manual.

Figura 13 – Exemplo de planilha utilizada para classificação manual de casos testes em relação à possibilidade automação

| DALEK ID | NAME | STATUS | COMMENTS |
|---|---|---|---|
| FAKE-ID-00036 | TC - [Latam] - [RCS] Attach many audio, video, image files - For Slot1 | Automatable | ECOXXXX is not present on BR Apps |
| FAKE-ID-00037 | TC - [Latam] - [RCS] Sharing Contacts/locations - For Slot2 | Not Automatable | Full storage interaction |
| FAKE-ID-00038 | TC - [Latam] - [RCS] Sharing Contacts/locations - For Slot1 | Not Automatable | Insert SIM Interaction |
| FAKE-ID-00039 | TC - [Latam] - [RCS] Chat in different network conditions - WiFi, Data - For Slot2 | Automatable | [Ready to Automation] Use TIM |
| FAKE-ID-00040 | TC - [Latam] - [RCS] Chat in different network conditions - WiFi, Data - For Slot1 | Automatable | [TC Ready for automation] Use TIM. Refactored based on FAKE-ID-104426 |
| FAKE-ID-00041 | TC - [Latam] - [RCS] Basic functionality check for Slot2 | Automatable | Insert/remove SD card |
| FAKE-ID-00042 | TC - [Latam] - [RCS] Basic functionality check for Slot1 | Automatable | [TC ready to automation] Go to refresh icon > Search new channels |
| FAKE-ID-00043 | TC - [Latam] - Make a VoLTE call while FDN enabled but FDN list empty | To Evaluate | Need to check audio quality |
| FAKE-ID-00044 | TC - Revoke easy prefix permissions and check the call redirecting app | To Evaluate | [READY FOR AUTOMATION] |
| FAKE-ID-00045 | TC - Disable Easy prefix during dual sim usage setup | To Evaluate | [Ready to Automation] Use TIM |
| FAKE-ID-00046 | TC - [Latam] - VoLTE - IMS registration check - DDS Slot 2 | To Evaluate | [TC Ready for automation] Use TIM. Refactored based on FAKE-ID-104426 |
| FAKE-ID-00047 | TC - Check Caller ID function while VoLTE interaction | To Evaluate | [Ready to Automation] Use TIM |
| FAKE-ID-00048 | TC - Time zone updating after phone leaves APM and connects to 4G network on SIM 1 and 3G on SIM 2 | Not Automatable | [Ready to Automation] Use TIM |
| FAKE-ID-00049 | TC - Time zone updating after phone leaves APM and connects to 4G network on both SIM cards | Not Automatable | Insert/remove SIM |
| FAKE-ID-00050 | TC - Check if voicemail name in call screen matches name in recent calls | To Evaluate | [TC Ready for automation] Use TIM. |

**Fonte:** Elaborada pelo autor (2020)

Essas planilhas serviram como base ao processo de aquisição do conhecimento (Seção 3.2.4). Para formalização e implementação da base de conhecimento (Seções 3.2.5 e 3.2.6), uma análise foi realizada sobre a informação da planilha e, com a ajuda de especialistas, foram

identificadas 6 áreas de teste de software: **Aplicações Core**, **GMS**, **Regressão**, **Regressão de GMS**, **Indicadores de Performance** e **Sanidade**. Foram criadas 6 regras de primeiro nível, uma para identificar cada uma dessas áreas. Por exemplo:

**SE** um caso de teste possui um marcador contendo a palavra "sanity",
**ENTÃO** esse caso de teste é de sanidade (área do caso de teste é igual a *SANITY*).

Porém, a identificação da área de teste não é suficiente para classificar o caso de teste entre manual ou automatizável. Assim, foram criadas regras de segundo nível (subgrupos) de inferência para cada área. Por exemplo:

**SE** a área do caso de teste é igual a *SANITY* **E** no caso de teste menciona "Amazon Account", "Amazon Widget"ou "Kindle",
**ENTÃO** o caso de teste não é automatizável (ou seja, é da classe **MANUAL**).

Cada área de teste pode ter uma quantidade diferente de regras de segundo nível e a ativação de apenas uma dessas regras de segundo-nível já define se um caso de teste pode ser automatizado ou não. Veja a Tabela 2.

Tabela 2 – Distribuição de subgrupos identificados para cada área de teste

| Área | Num. de subgrupos |
|---:|---|
| Aplicações Core | 16 |
| GMS | 32 |
| Regressão | 28 |
| Regressão de GMS | 25 |
| Indicadores de Performance | 11 |
| Sanidade | 26 |

**Fonte:** Elaborada pelo autor (2020)

A identificação das áreas e dos subgrupos foi um trabalho realizado pelos especialistas no processo de construção das regras. Esse processo teve várias iteração até chegar a um resultado da identificação de 6 áreas de testes e 50 subgrupos com resultados bastantes satisfatórios. Alguns dos subgrupos identificados podem fazer parte de mais de uma área de testes.

Como visto na Seção 3.2.7, o processo de criação da base de regras inclui validações intermediárias para verificar a qualidade das regras criadas. Quando o resultado for insatisfatório, a base de regra será modifica, podendo incluir, apagar ou modificar regras existentes. Esse

processo se repete até que o especialista considere o resultado da validação da base satisfatório (ver Capítulo 5).

Conforme foi mencionado na Seção 4.3, o especialista deve estar atento a mudanças no framework de automação e no ambiente de execução de testes. Essas mudanças podem degradar a qualidade do sistema e será necessário uma revisão das regras impactadas pelas mudanças. Esse esforço já é familiar ao time de especialistas, pois a cada mudança esse time reavalia a classificação dos casos de testes.

Nas seções seguintes, serão apresentados os módulos do DRL TC Classifier em relação às suas características, implementação e função dentro do sistema.

### 4.4.2 Módulo 1 - *Dalek Test Case Query Service*

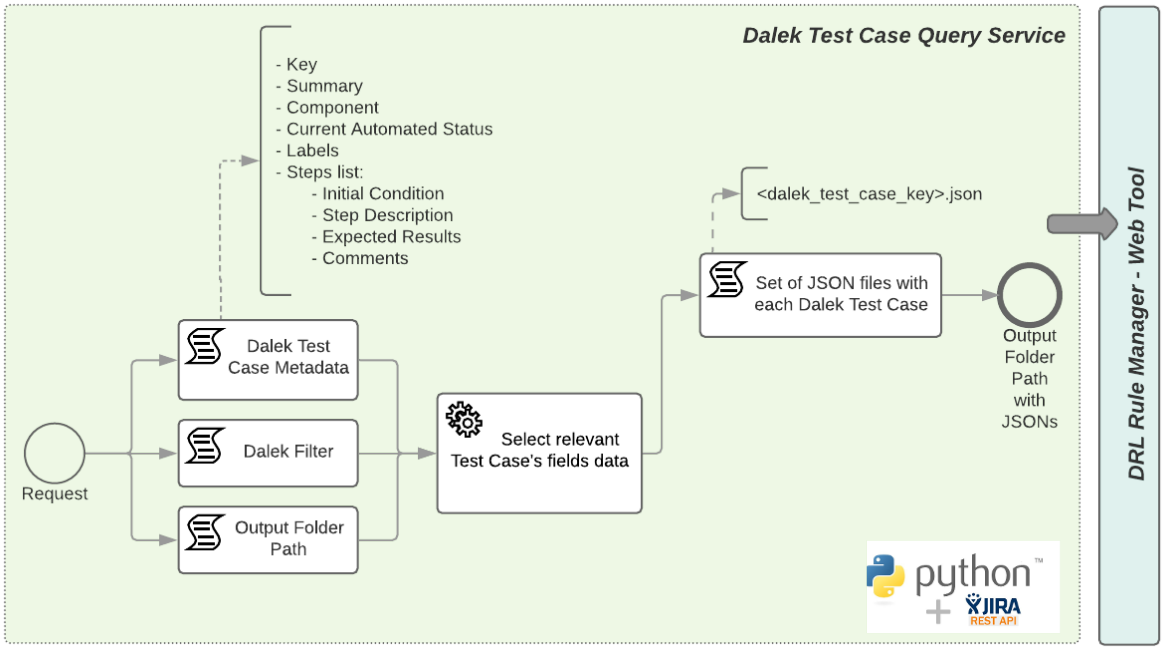
Este primeiro módulo é responsável por recuperar todos os casos de teste referente a uma área de teste ou alguma consulta de testes sobre a base de casos de teste. Ele corresponde a etapa que recupera e representa os documentos, apresentado na Figura 1, Seção 2.1.

Para recuperação, o módulo se conecta à base da empresa. Como a maioria dos sistemas desenvolvidos e utilizados na empresa parceira, bem como a existência de *scripts* que usam a API (SCHELLER; KÜHN, 2015) (Interface de Programação de Aplicações, do inglês, *Application Programming Interface*) de busca no Dalek, este módulo foi desenvolvido na linguagem de programação Python (ROSSUM; JR, 1995).

Na Figura 14 é mostrado que o início do serviço se dá com a entrada dos campos de um caso de teste relevantes para classificação: Dalek Test Case Metadata, o filtro referente a uma consulta no Dalek e o caminho da pasta onde esses casos de teste serão armazenados. O filtro referente a uma consulta no Dalek é uma consulta em JQL (RADIGAN, 2019) para consultar a base de casos de teste no Dalek que deverão ser considerados pelo serviço:

- ID;
- Sumário;
- Descrição;
- Tags ou marcadores (*labels*);
- Passos:

Figura 14 – Processo do módulo: *Dalek Test Case Query Service*




**Fonte:** Elaborada pelo autor (2020)

- Pré-Condição;
- Ação;
- Resultado esperado;
- Comentário.

Em relação à seleção de características apresentada na seção 2.4, como o tamanho e quantidade dos documentos era pequenos, nenhum dos métodos apresentado foi utilizado. Apenas os campos dos casos de teste indicados pelos especialistas foram considerados para classificação.

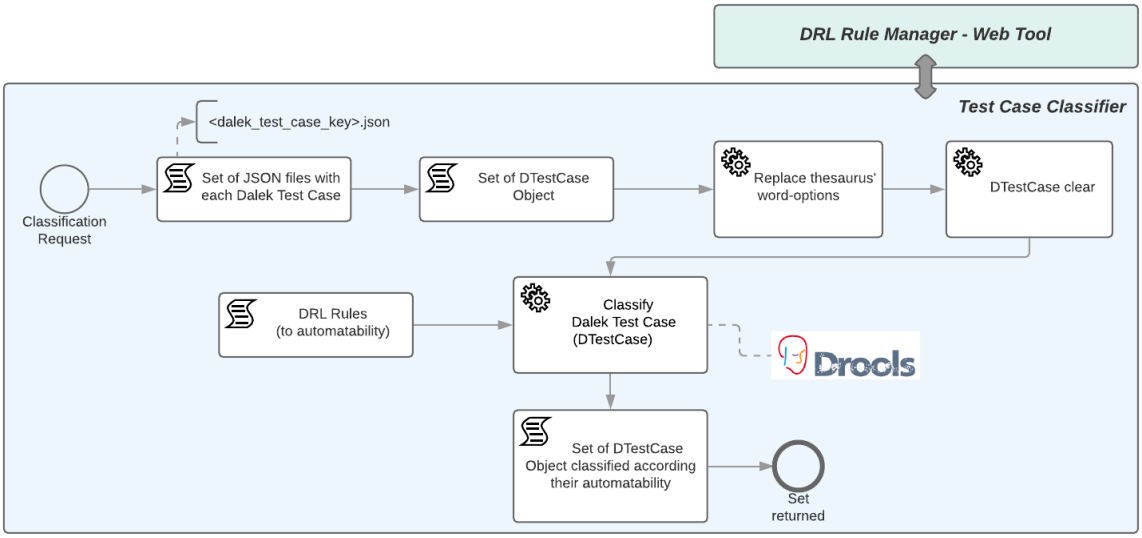
Com as três entradas mencionadas acima, a consulta é realizada e, para cada caso de teste retornado pela consulta, é gerada uma representação estruturada fácil de ser lida por um humano ou por uma máquina. Para a representação, o resultado dessa busca é estruturado num formato de arquivo JSON (*.json). No final da execução deste serviço os arquivos podem ser encontrados na pasta indicada e estão prontos para serem utilizados nos demais módulos desta solução. Alguns exemplos desses documentos podem ser encontrados no Apêndice B.

### 4.4.3 Módulo 2 - *Test Case Classifier Service*

Este módulo é o coração deste trabalho e também está implementado em forma de serviço. Inicialmente, ele realiza um pré-processamento nos casos de teste recuperados a partir do Dalek, que estão descritos segundo a representação estruturada criada pelo módulo anterior. A seguir, as regras DRL são executadas para cada caso de teste. Logo, ele corresponde à etapa que preprocessa e aplica o método de classificação nos documentos.

Ainda referenciando o processo na Figura 1, este módulo inclui a etapa de seleção de características, realizada tanto no momento em que o especialista indica os campos do caso de teste relevantes para a classificação, quando na criação das regras em que é indicado o que deve ser considerado na classificação.

Figura 15 – Processo do módulo: *Test Case Classifier Service*




**Fonte:** Elaborada pelo autor (2020)

Na Figura 15 é visto que o módulo de execução de regras recebe um conjunto de arquivos JSONs, transforma os campos desses casos de teste em objetos em memória e esses objetos em memória sofrem o processo de substituição de palavras utilizando um tesauro (CARTER, 1994; GREFENSTETTE, 2012). Não foi necessário eliminar *stopwords* nem realizar operação de *stemming*, pois os casos de teste resultam em documentos muito curtos.

Em relação à arquitetura genérica de um Sistema Baseado em conhecimento, Figura 2, Seção 2.5.1, este módulo é onde se encontra o "Motor de Inferência". Ele utiliza o Drools para o processamento das regras existentes. As características do Drools já foram listados no

Capítulo 3. E, entre elas, as mais relevantes para este trabalho são o encadeamento progressivo, fácil manutenibilidade, a possibilidade de representar as regras de negócio sem precisar alterar ou reimplementar novas ferramentas a cada mudança no ambiente, e ser mantida por uma comunidade ativa numa linguagem de programação altamente disseminada na equipe e no mercado.

Como o Drools foi desenvolvido principalmente para a linguagem de programação Java (ARNOLD et al., 2000), este módulo também é escrito em Java. Mesmo que o primeiro módulo tenha sido escrita na linguagem Python, não há qualquer problema uma vez que a comunicação entre as duas partes é assíncrona e se dá pelos casos de teste e regras descritas em arquivos JSONs.

Para este serviço, similar ao que foi feito no primeiro módulo da solução implementada, a representação dos casos de teste foi realizada considerando seus campos mais relevantes para a classificação em relação a sua automação. Uma vez carregados em memória novamente, agora na JVM (STÄRK; SCHMID; BÖRGER, 2012), é possível realizar as operações e tratamentos necessários para a classificação dos casos de teste.

Este módulo armazena e dispara as regras responsáveis por classificar os casos de teste entre manuais ou automatizáveis. Como também já foi mencionado, as regras indicam, por exclusão, a possibilidade de um caso teste ser automatizável. Logo, as regras determinam se um caso deve ser executado manualmente e o caso ser classificado como "MANUAL". As regras criadas podem ser encontradas no Apêndice A — Regras DRL. Um exemplo dessas regras, na Figura 16, indica que um caso de teste de sanidade, relativo ao brilho da tela do dispositivo móvel, não é automatizável.

Figura 16 – Exemplo de regra relativa a casos de teste de sanidade para testar brilho de tela

```
489 rule "BRIGHTNESS"
490     agenda-group "AutomationStatus"
491     no-loop true
492     lock-on-active true
493     salience 40
494     when
495         $dtc : DTestCase( testAreas contains "SANITY", tcString matches ".*brightness.*")
496     then
497         System.out.println("'brightness / ' rule");
498         $dtc.setAutoClassificationEnum(TestCaseAutoClassificationEnum.MANUAL);
499         $dtc.addRule("BRIGHTNESS");
500         update($dtc);
501 end
```

**Fonte:** Elaborada pelo autor (2020)

Como mostra a Figura 16, cada regra possui vários campos para o processamento do

Drools e para a classificação adequada de um caso de teste.

- Estrutura das Regras:
  - Nome da regra (*rule*) — identifica a regra;
  - Grupo da regra (*agenda-group*) — permitem que você coloque regras em grupos e coloque esses grupos em uma pilha.;
  - (*no-loop*) — quando a opção é selecionada, a regra não pode ser reativada (em *loop*) se uma consequência da regra disparar novamente uma condição previamente cumprida. Quando a condição não é selecionada, a regra pode ser repetida nessas circunstâncias;
  - (*lock-on-active*) — Aplicável apenas a regras dentro de grupos de fluxo de regra ou grupos de agenda. Quando a opção for selecionada, na próxima vez que o grupo de fluxo de regra para a regra se tornar ativo ou o grupo de agenda para a regra receber um foco, a regra não poderá ser ativada novamente até que o grupo de fluxo de regra não esteja mais ativo ou o grupo de agenda perca o foco.;
  - Prioridade da regra (*salience*) — Um inteiro que define a prioridade da regra. Regras com um valor de *salience* mais alto recebem prioridade mais alta quando solicitadas na fila de ativação;
  - Condição da regra (when) — a condição (**SE**) de uma regra;
  - Consequência da regra (then) — a consequência (**ENTÃO**) de uma regra.

Considerando a avaliação dos casos de teste em relação a automação e os dois níveis de regras criados (para identificar a área do caso de teste e para decidir se ele é automatizável), não há a necessidade de *loops* durante a execução das regras. Portanto os campos **no-loop** e **lock-on-active** são basicamente fixos neste sentido. Já a prioridade da regra apenas é considerada para organizar os dois níveis citados, sendo a identificação das regras o primeiro grupo avaliado, e a possibilidade de automação, o segundo grupo.

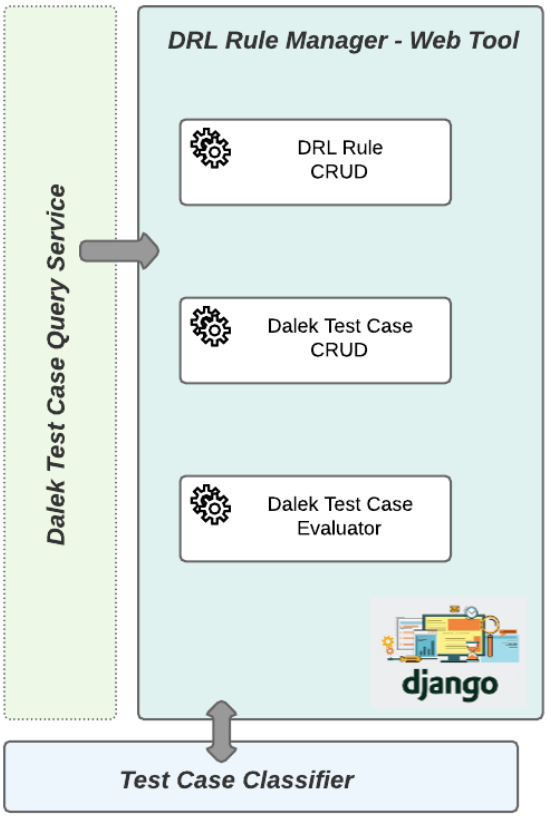
Esses campos serão utilizados nas telas do próximo módulo, Seção 4.4.4, para a criação, edição e gerenciamento de regras existentes.

### 4.4.4 Módulo 3 - *DRL Rule Manager - Web Tool*

Este módulo é a interface gráfica com funções (Figura 17) que reúnem a utilização dos Módulos 1 e 2, descritos anteriormente nas seções 4.4.2 e 4.4.3, respectivamente. Ainda, também visto anteriormente, na Figura 12, é evidenciada a relação com os demais módulos.

Considerando a arquitetura genérica de um Sistema Baseado em conhecimento, Figura 2 do Capítulo 3, este módulo é onde se encontra a "Interface Gráfica"e onde são armazenadas as regras, portanto, também a "Base de Conhecimento".

Figura 17 – Funções do módulo: *Web Tool*

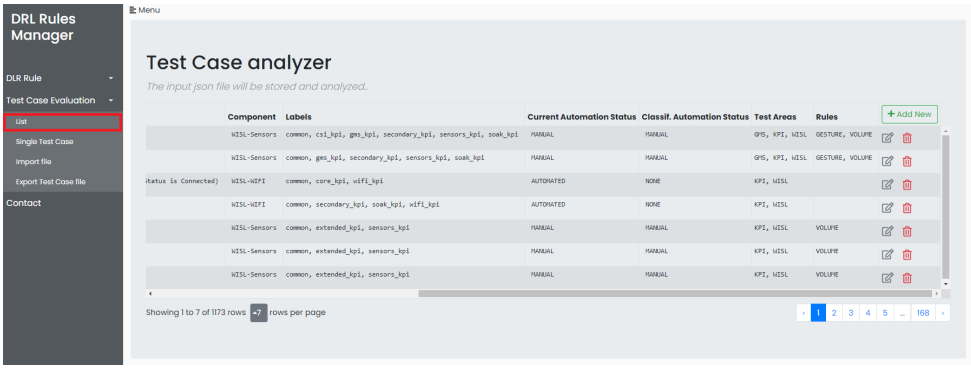



**Fonte:** Elaborada pelo autor (2020)

Na relação com primeiro módulo, a interface permitirá utilizar, importando, os casos de teste exportados no Módulo 1. Os casos, ao serem carregados, são enviados ao segundo módulo para avaliação e em seguida armazenados em um banco de dados, podendo ser consultados posteriormente. No sistema, há uma tela para listar os casos analisados pelo sistema (Figura 18); outra tela para inserir individualmente o caso de teste (Figura 19); e uma tela para importação dos casos de teste (Figura 20) em formato *.json similar ao que está representado

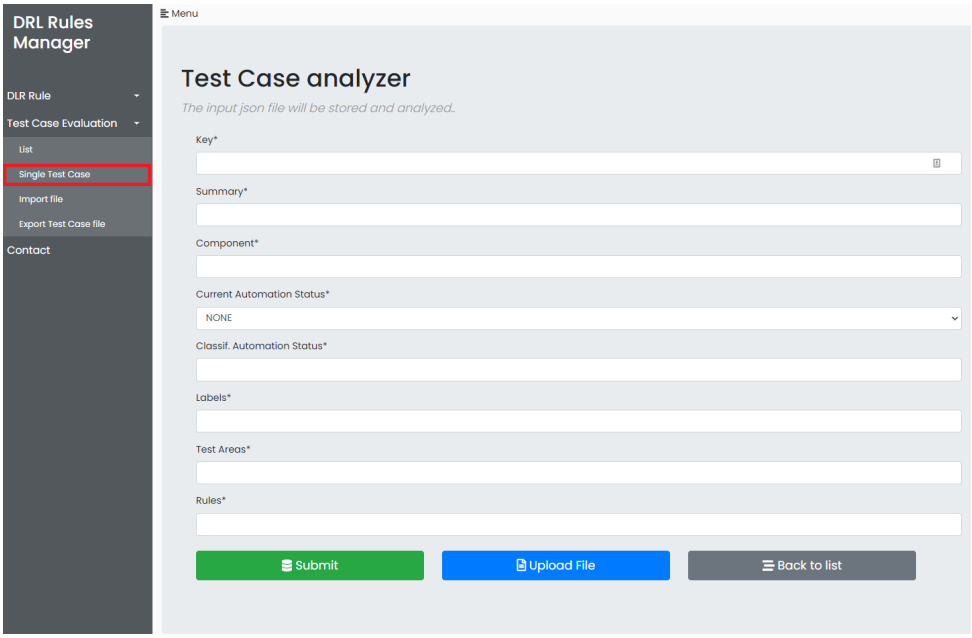
no Apêndice B.

Figura 18 – Lista dos casos de teste




**Fonte:** Elaborada pelo autor (2020)

Figura 19 – Inserção de casos de teste para análise

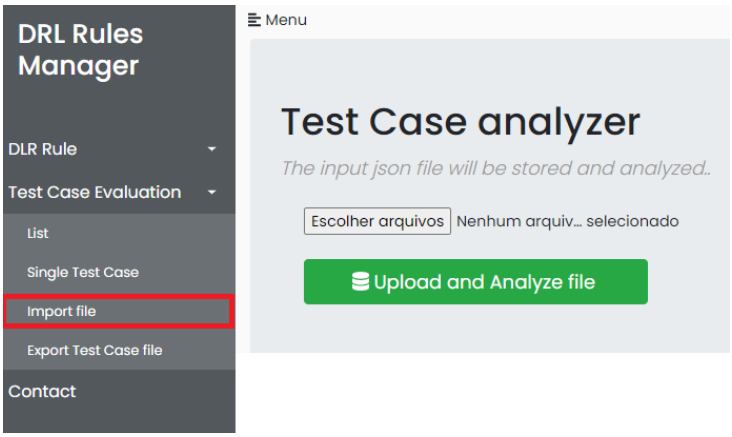



**Fonte:** Elaborada pelo autor (2020)

Na relação com o segundo módulo, além do envio de casos de teste para análise, este terceiro módulo é responsável por permitir ao gerente do sistema criar novas regras, atualizar ou deletar as regras existentes. Além disso, ele permite listar (Figura 21) todas as regras existentes no sistema, bem como os tipos das regras. As regras podem ser escritas em formato textual e seus campos são os mesmos listados na Seção 4.4.3, como pode ser visto na Figura 22.

Além de escrever ou editar as regras individualmente, é possível também importar as regras através a tela apresentada na Figura 23. Nela, é possível escolher arquivos com extensão *.drl.

Figura 20 – Importação de um ou mais casos de teste




**Fonte:** Elaborada pelo autor (2020)

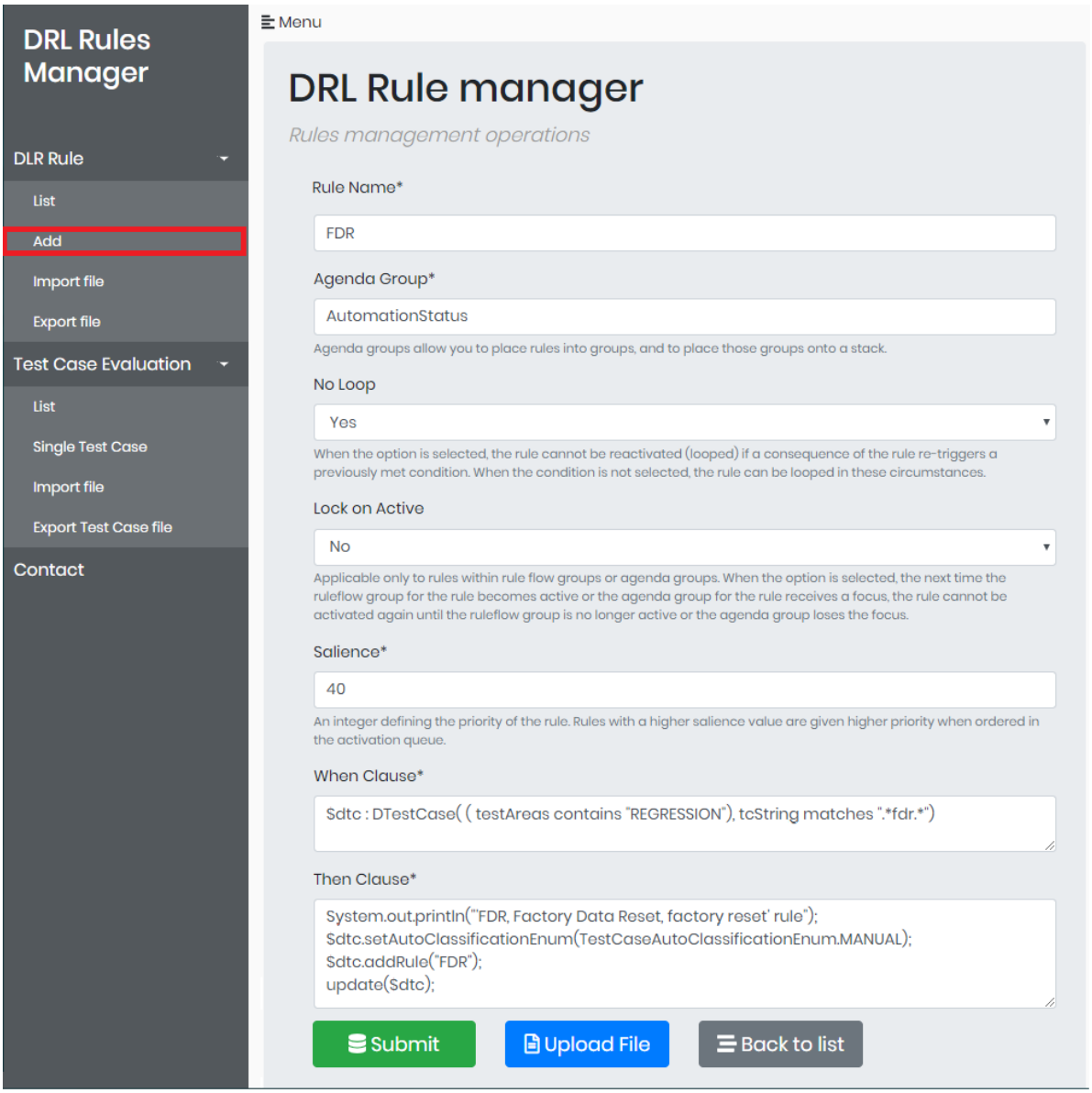
Figura 21 – Lista das regras DRL


**Fonte:** Elaborada pelo autor (2020)

Esses campos, no módulo de carregamento de regras, são suficientes para a transformação das regras em formato textual para regras em formato DRL ou *.drl. Este formato, por sua vez, é reconhecido pela ferramenta de inferência por regras, Drools, utilizada no Módulo 2.

Uma vez que todas as regras identificadas e registradas na planilha pelos especialistas são movidas para a Web Tool, a atualização, remoção das regras existentes e criação de novas regras podem ser realizadas na ferramenta. Seguindo o fluxo realizado atualmente, para uma nova classificação de outras áreas de teste ou de novos de casos nas áreas já avaliadas, os casos serão submetidos aos processo de classificação descrito nesta seção. Caso o resultado da classificação seja identificado como errado para algum teste, as regras relacionadas deverão ser refinadas ou até mesmo novas regras criadas. Os especialistas poderão identificar a corretude da classificação durante a automatização dos casos identificados como AUTOMATABLE ou

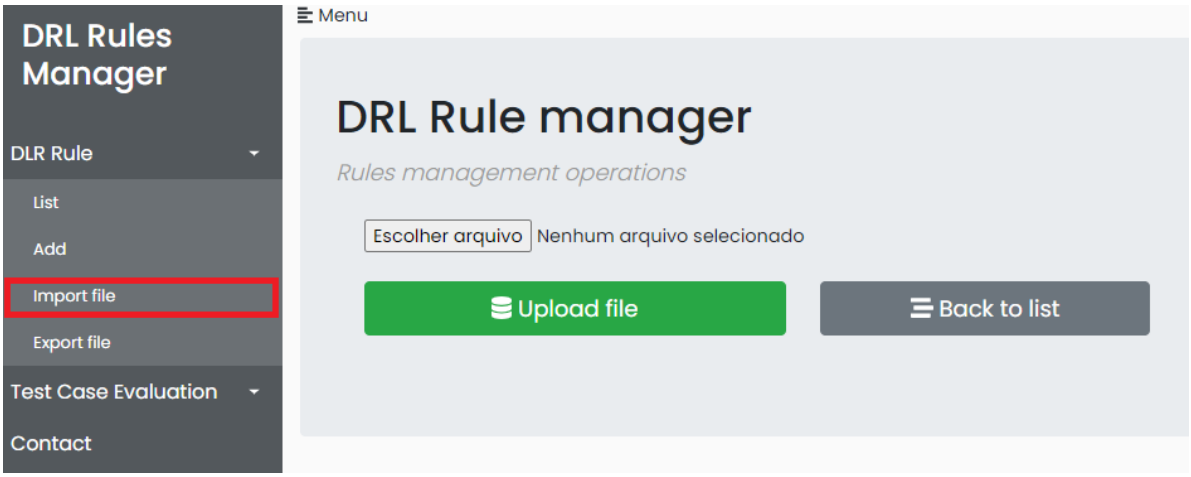
Figura 22 – Regra **FDR** sendo criada no DRL TC Classifier




**Fonte:** Elaborada pelo autor (2020)

numa revisão realizadas sobre os casos classificados como MANUAL.

Figura 23 – Importação das regras DRL — arquivos *.drl




**Fonte:** Elaborada pelo autor (2020)

## 4.5 CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentado o problema real de uma empresa de teste de software para dispositivos móveis. Foi introduzido o conceito de teste de software e a importância da sua utilização na engenharia de software. Também foram descritos alguns problemas identificados nesta área e, em seguida, foi apresentada uma proposta de solução para a problemática da classificação de casos de teste em relação à possibilidade sua automatização, o DRL TC Classifier.

Na apresentação do DRL TC Classifier, foi apresentado o processo realizado para a criação das regras. A partir deste processo, os 3 módulos criados para a solução e seus relacionamentos são executados: (1) a comunicação com a base de casos de teste existentes, (2) o módulo que processa as regras criadas para classificar os casos de teste e (3) o módulo que permite a criação, visualização e edição dessas regras.

No próximo capítulo, será apresentado o processo de validação do DRL TC Classifier como um sistema para classificação de documentos de texto e os resultados obtidos nesta validação.

# 5 AVALIAÇÃO DO CLASSIFICADOR

O sistema será avaliado com base nas métricas apresentadas na seção 2.6. A seguir, serão mostrados alguns exemplos de regras de classificação. Por fim, serão discutidos os resultados e contribuições alcançados.

## 5.1 METODOLOGIA DE AVALIAÇÃO

Inicialmente, especialistas analisaram manualmente um corpus de casos de teste aleatoriamente selecionados, classificando-os entre manuais ou automatizáveis. Esse corpus etiquetado foi usado para avaliar o desempenho do DRL TC classifier. No total, 662 casos de teste de diversas áreas de testes foram analisados, dentre elas: *Core Apps*, GMS, *Regression*, GMS Regression, *KPI*, *MME*, *Kernel* e *Sanity*. Dentre os casos de teste analisados, 331 foram considerados da classe **MANUAL** e 331 casos de teste foram considerados da classe **AUTOMATABLE**. Entretanto, não foi considerado um balanceamento entre as áreas de teste uma vez que na maioria desses existem poucos casos de teste. As regras criadas para identificar as áreas de teste e a possibilidade de automação dos casos de testes podem ser encontradas no Apêndice A - Regras DRL.

Os identificadores (IDs) dos casos de teste foram guardados numa planilha, indicando qual a área de teste, a classificação real do teste em relação a sua possibilidade de automação (classificação realizada pelos especialistas), a classificação do DRL TC Classifier e a razão da classificação do sistema. Conforme a avaliação dos especialistas, apenas os casos de teste classificados como MANUAL têm sua razão justificada. Com essa planilha formada, as justificativas foram agrupadas manualmente e deram origem a um conjunto de regras a serem processadas pelo motor de inferência. É possível visualizar a estrutura da matriz na Figura 24.

Para criação da matriz de confusão para avaliação do DRL TC Classifier, foram consideradas as seguintes possibilidades de classificação: MANUAL como sendo a classe positiva e *AUTOMATABLE* como classe negativa. Os casos de teste avaliados encontram-se marcados no campo *Automation Status* com um dos seguintes rótulos: *Automated*, *Pending* e Manual. Os casos marcados como *Automated* e *Pending* são considerados com *AUTOMATABLE*.

Figura 24 – Matriz de análise

| ID do Caso de Teste | Área do Caso de Teste | Classificação Especialista | Classificação DRL TC Classifier | Regras DRL Associadas ao Caso de Teste |
|---|---|---|---|---|
| TC-101 | REGRESSION | MANUAL | MANUAL | TWO_FOUR_GHZ |
| TC-102 | REGRESSION | MANUAL | MANUAL | BOTA, TWO_FOUR_GHZ |
| TC-103 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-105 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-106 | GMS_REG | MANUAL | MANUAL | GRAYSCALE |
| TC-107 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-108 | SANITY | MANUAL | MANUAL | FIVE_G, MOD |
| TC-109 | REGRESSION | MANUAL | MANUAL | CHARGER, ROAMING |
| TC-110 | REGRESSION | MANUAL | MANUAL | GESTURE, GOOGLE_VOICE |
| TC-111 | GMS_REG | MANUAL | MANUAL | GOOGLE_VOICE |
| TC-112 | SANITY | MANUAL | MANUAL | FIVE_G, MOD, QUALITY |
| TC-113 | SANITY | MANUAL | MANUAL | FIVE_G, MOD, QUALITY |
| TC-114 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-115 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-116 | GMS_REG | PENDING | MANUAL | DTV, FM_RADIO |
| TC-117 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-120 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-121 | GMS_REG | PENDING | AUTOMATABLE | |
| TC-124 | REGRESSION | MANUAL | AUTOMATABLE | |
| TC-125 | REGRESSION | MANUAL | AUTOMATABLE | |
| TC-126 | GMS_REG | MANUAL | MANUAL | WEARABLE |
| TC-127 | REGRESSION | PENDING | AUTOMATABLE | |
| TC-128 | REGRESSION | PENDING | MANUAL | QUALITY |
| TC-129 | REGRESSION | AUTOMATED | AUTOMATABLE | |

**Fonte:** Elaborada pelo autor (2020)

## 5.2 AVALIAÇÃO DO DRL TC CLASSIFIER

Para criação da matriz de confusão da avaliação do DRL TC Classifier, foram consideradas as seguintes possibilidades de classificação: **MANUAL** quando positivo e ***AUTOMATABLE*** quando negativo. A matriz representada na Tabela 3 foi gerada considerando a classificação dos especialistas, apresentada na Figura 24, e a classificação do DRL TC Classifier.

Aplicando as métricas apresentadas na seção 2.6 aos valores obtidos na Tabela 3, obtêm-se

Tabela 3 – Matriz de confusão da avaliação do DRL TC Classifier

| | Classe Real | |
|---|---|---|
| Classe Prevista | MANUAL | AUTOMATABLE |
| MANUAL | 300 | 74 |
| AUTOMATABLE | 31 | 257 |

**Fonte:** Elaborada pelo autor (2020)

os seguintes valores:

- **VP** = 300
- **VN** = 257
- **FP** = 74
- **FN** = 31
- **ACURÁCIA** = 84.1%
- **SENSIBILIDADE** = 90.6%
- **ESPECIFICIDADE** = 77.6%
- **EFICIÊNCIA** = 84.1%
- **VALOR PREDITIVO POSITIVO** = 80.2%
- **VALOR PREDITIVO NEGATIVO** = 89.2%
- **F1 SCORE** = 85.08%

As métricas mais relevantes para este trabalho são as **acurácia** (84.1%), **precisão** (80.2%), **sensibilidade** (90.6%) e **F1-Score** (85.08%) que são as métricas relacionadas à classe positiva (i.e., MANUAL). Note que, uma vez classificado como MANUAL, o caso de teste só será reavaliado se ele for alterado ou se ocorrer alguma mudança no framework de automação de testes. Já os casos de teste classificados como AUTOMATABLE serão revisitados pelo time de automação no momento da sua codificação, e uma correção dessa classificação pode ser feita caso necessário. Assim, é preferível classificar erroneamente um CT manual como sendo automatizável do que o inverso.

Nas próximas seções, 5.2.1, 5.2.2, 5.2.3 e 5.2.4, serão mostrados alguns exemplos de casos de teste analisados escolhidos aleatoriamente de cada quadrante da matriz de confusão.

### 5.2.1 VP - Verdadeiro Positivo

Como já mencionado, VP são casos de teste considerados pelo especialista como pertencentes à classe MANUAL que foram classificados corretamente pelo DRL TC Classifier. Esses

casos realmente obedecem às regras disparadas e não são passíveis de automação atualmente. As representações completas dos casos VP se encontram no Apêndice B.

- TC-001
  - Área de teste: *Regression*
  - Regras disparadas: *BT*, *CHARGER*, *HEADSET*, *POWER_KEY*, *ROAMING*, *TWO_FOUR_GHZ*, *VOLUME*
  - Justificativa: *todas as regras foram disparadas devidamente. O caso pede a utilização de acessórios específicos BT, utilização de tecla física, alteração de rede durante o caso de teste e verificação de alteração de volume.*
- TC-002
  - Área de teste: *Sanity*
  - Regras disparadas: *FDR*
  - Justificativa: *a regra foi disparada devidamente. O caso pede a realização de um factory data reset e essa operação poderá comprometer a conexão estabelecida entre o servidor de teste e o produto em teste, além de apagar os logs utilizados no report da execução.*
- TC-003
  - Área de teste: *GMS Reg*
  - Regras disparadas: *WEARABLE*
  - Justificativa: *a regra foi disparada devidamente. O caso pede a utilização de um acessório wearable não disponível*
- TC-004
  - Área de teste: *MME*, *KPI*
  - Regras disparadas: *LUMES*
  - Justificativa: *a regra foi disparada devidamente. O caso de teste pede uma determinada luminosidade na configuração do ambiente e o framework e instalações físicas não suporta essa configuração.*

### 5.2.2 FP - Falso Positivo

FP são casos de teste considerados pelo especialista como pertencentes à classe *AUTOMATABLE* que foram classificados como MANUAL de forma errada pelo DRL TC Classifier. As representações completas dos casos FP se encontram no Apêndice B.

- TC-005
  - Área de teste: *Regression*, *GMS*, *Core Apps*
  - Regras disparadas: *GESTURE*
  - Justificativa: *o caso apenas cita que a navegação é gestual: "In the android Q, The system gesture is fully gestural navigation". Essa citação fez com que a regra "GESTURE" fosse disparada. Como sugestão, o teste poderia ser reescrito para retirada da citação óbvia.*
- TC-006
  - Área de teste: *GMS Reg*
  - Regras disparadas: *DTV*, *FM_RADIO*
  - Justificativa: *o caso apenas pede a abertura das aplicações: "Open some apps (e.g: FM Radio/GoogleMaps/DTV)". Como o teste não indica a interação, apenas a sua abertura, o passo é passível de automação. Como sugestão, a regra poderia ser atualizada para verificar quando é pedido interações além da abertura da aplicação. Entretanto, poderia ser muito custoso mapear essas regras para todas as aplicações e, na opinião do especialista, não é necessária a atualização da regra.*
- TC-007
  - Área de teste: *Regression*, *GMS*, *Core Apps*
  - Regras disparadas: *BT*
  - Justificativa: *o caso indica operações muito básicas com o BT passíveis de automação. Como sugestão, a regra poderia ser atualizada ou poderiam ser criadas novas regras para identificar quais tipos de operações são passíveis de automação.*
- TC-008

  - Área de teste: *Sanity*
  - Regras disparadas: *QUALITY*
  - Justificativa: *o caso pede a realização de verificação de qualidade, mas como há outros testes de sanidade que já fazem isso, a verificação foi ignorada. Como sugestão, é possível atualizar o caso de teste para centralizar as verificações de qualidade apenas em casos executados manualmente.*

### 5.2.3 FN - Falso Negativo

FN são casos de teste considerados pelo especialista como pertencentes à classe MANUAL que foram classificados como *AUTOMATABLE* de forma errada pelo DRL TC Classifier. As representações completas dos casos FN se encontram no Apêndice B.

- TC-009
  - Área de teste: *Sanity*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *o caso não aponta nenhum motivo direto para não ser automatizado. Entretanto, o especialista indicou que a manutenção do teste seria muito custosa e beneficiaria uma versão de software muito específica.*
- TC-010
  - Área de teste: *GMS*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *o caso não aponta nenhum motivo direto para não ser automatizado. Entretanto, o especialista indicou que a utilização de "**secure hardware**" indicado no teste seria muito custoso em relação a execução de GMS como um todo. Como sugestão, seria possível a criação de uma regra para a área GMS e a utilização de "secure hardware".*
- TC-011
  - Área de teste: *KPI*
  - Regras disparadas: *nenhuma regra foi disparada*

  - Justificativa: *o caso aponta a necessidade de interações com objetos externos ao dispositivo: "Cover the sensors (Prox & ALS sensors) only with an object such as hand, paper, etc..". Como sugestão, uma nova regra poderia ser criada.*

- TC-012
  - Área de teste: *Regression*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *o caso não aponta nenhum motivo direto para não ser automatizado. O especialista não conseguiu identificar a razão do teste ser considerado MANUAL e alterará o "Automation Status" do teste para PENDING.*

### 5.2.4 VN - Verdadeiro Negativo

VN são casos de teste considerados pelo especialista como pertencentes à da classe *AUTOMATABLE* que foram classificados corretamente pelo DRL TC Classifier. Esses casos não disparam nenhuma das regras e realmente são passíveis de automação atualmente. As representações completas dos casos VN se encontram no Apêndice B.

- TC-013
  - Área de teste: *Sanity*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *não há nada presente nos campos dos testes que indique que ele não pode ser automatizável.*

- TC-014
  - Área de teste: *Regression*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *não há nada presente nos campos dos testes que indique que ele não pode ser automatizável.*

- TC-015
  - Área de teste: *GMS Reg*

  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *não há nada presente nos campos dos testes que indique que ele não pode ser automatizável.*

- TC-016
  - Área de teste: *Reg*
  - Regras disparadas: *nenhuma regra foi disparada*
  - Justificativa: *não há nada presente nos campos dos testes que indique que ele não pode ser automatizável.*

## 5.3 CONSIDERAÇÕES FINAIS

Neste capítulo, foi realizada a avaliação do sistema e foram apresentados os valores para as seguintes métricas: **acurácia**, **sensibilidade** (ou *recall*), **especificidade**, **eficiência**, **precisão** (ou valor preditivo positivo), **valor preditivo negativo** e ***F1 Score***. Foi identificado que as métricas mais relevantes, no caso do ***DRL TC Classifier***, são as **acurácia**, **precisão**, **sensibilidade** e **F1 Score**, com valores 84.1%, 80.2%, 90.6% e 85.08%, respectivamente. Este resultado foi considerado pelos especialistas como sendo de grande relevância no processo de classificação dos casos de teste em relação à possibilidade de sua automação.

Após a avaliação, foi possível verificar em casos de teste selecionados aleatoriamente que há cenários onde as regras existentes podem ser refinadas ou novas regras podem ser criadas; existem cenários onde a classificação manual foi realizada de forma errada; há cenários onde o texto que descreve o caso de teste deve ser melhorado para que sua classificação seja compatível com a interpretação do especialista; há cenários de exceções à regra que não devem de ser mapeados; e, para a grande maioria dos casos de teste, há cenários em que as regras se comportaram da maneira esperada.

Neste trabalho, nós avaliamos apenas o desempenho do classificador. Porém existem outras questões a serem examinadas e avaliadas:

- Usabilidade da interface;
- Tempo de aprendizagem na utilização do sistema e formato de escrita das regras do Drools;

- Esforço para utilizar o sistema;
- Como melhorar o desempenho do sistema;
- Generalização da solução, por exemplo, para outras áreas de teste;

Essas questões serão abordadas nos trabalhos futuros. No próximo capítulo, serão apresentadas as conclusões deste trabalho, bem como serão indicados trabalhos futuros que poderão estender naturalmente a pesquisa aqui reportada.

# 6 CONCLUSÃO

Este capítulo tem como objetivo apresentar algumas considerações finais sobre os principais tópicos abordados nesta dissertação, incluindo as contribuições alcançadas e indicações para trabalhos futuros. O problema abordado foi identificado em uma empresa real de testes de software para dispositivos móveis, sendo parte de um processo do time de automação de testes.

Inicialmente, foi realizado um estudo sobre as técnicas de classificação e, posterior e mais aprofundadamente, sobre a engenharia do conhecimento com a utilização de regras. Esse estudo possibilitou a criação de um Sistema Baseado em Conhecimento que permite o registro, exibição e manipulação das informações inicialmente retidas apenas por alguns especialistas da empresa.

Com base nesses estudos, foi construído um sistema protótipo, o DRL TC Classifier, para auxiliar a realização da classificação de casos de teste para dispositivos móveis em relação a sua possibilidade de automação. Nossa solução utiliza conceitos da engenharia do conhecimento baseada em regras.

## 6.1 PRINCIPAIS CONTRIBUIÇÕES

A principal contribuição do trabalho apresentado neste documento foi o desenvolvimento de um sistema para auxiliar a realização da classificação de casos de teste para dispositivos móveis em relação a sua possibilidade de automação.

Uma outra grande contribuição deste trabalho foi a construção da base de regras que armazena conhecimento antes de posse de apenas alguns especialistas no domínio de automação de testes. Agora esse conhecimento poderá ser compartilhado dentro das equipes de maneira mais maneira mais clara, precisa e uniforme.

Outra contribuição é a possibilidade de observar o '*log*' de execução do sistema, que poderá ser utilizado em estudos e análises posteriores, a fim de identificar quais regras são mais ativadas durante o processo de classificação. Essa informação certamente trará 'pistas' sobre as características mais frequentes nos CTs que levam à sua classificação como não automatizáveis. A partir desses resultados, espera-se montar um plano de ação mais efetivo para permitir a automação de cada vez mais casos de teste.

Além das contribuições mencionadas acima, também faz-se relevante ressaltar que técnicas

e métodos mais tradicionais na área de Inteligência Artificial ainda podem ser aplicados em problemas atuais e ainda resultar em performances bastante satisfatórias.

## 6.2 LIMITAÇÕES

Uma limitação deste trabalho é observada quando os casos de teste estão mal escritos, ou seja, quando o CT não segue a estrutura padrão da empresa. A falta de estruturação no texto dos CTs inviabiliza uma classificação mais refinada, por exemplo, por passo de teste. Quando um CT é considerado manual apenas por um passo do corpo do teste, uma possível ação seria quebrar esse CT em dois, sendo um autonomizável e outro manual. Porém, para que isso fosse possível, seria necessário identificarmos corretamente as 3 partes constituintes de um CT: pré-condições, passos e resultados esperados. A partir daí, seria necessário ainda construir o resultado esperado para o 1º CT criado a partir do CT original, e construir as pré-condições do 2º CT criado. Apesar de ser uma solução complexa, ainda nos parece promissora. Porém, na falta de uma estruturação correta do CT, ela se torna inviável.

A solução deste problema está fora do escopo deste trabalho. Entretanto, uma possível solução para essa limitação é estruturar os casos de teste utilizando uma linguagem natural controlada (CNL, do inglês *Controlled Natural Languages*) (SCHWITTER, 2010). Esta ideia já foi proposta no passado, entretanto, como não houve adesão de todos os times, a utilização de uma CNL não foi bem sucedida.

Uma segunda limitação bastante relevante é que, para alguns casos de teste específicos, há a decisão de negócio de não automatizá-los. Mesmo que o teste em questão seja bastante simples, para algumas situações não é relevante economicamente manter determinados casos automatizados. Este cenário é bastante complexo de ser representado via regras pois não é relativo diretamente a uma área de teste ou funcionalidade específica. Estes casos, exceções, foram deixados de fora da análise automática com as regras, e precisarão ser identificados isoladamente de forma manual.

Uma outra limitação para este trabalho foi a falta de conhecimento prévio na linguagem DRL (Drools Rule Language). Apesar de todo esforço e dedicação neste trabalho, é notório que ainda há possibilidades de um aproveitamento consideravelmente maior em relação a potência e representatividade da linguagem. Existe disponível uma documentação bastante vasta e completa (HAT, 2020a), e um plugin (HAT, 2020b) para a IDE de desenvolvimento também popular na comunidade: Eclipse (MURPHY; KERSTEN; FINDLATER, 2006). O plugin

ajuda bastante na formatação e escrita das regras, mas não permite acompanhar ou realizar depuração das condições das regras, apenas do consequente. De forma similar, também só é possível realizar *logs* nas cláusulas ***THEN***. Ainda considerando a utilização da IDE e o plugin do Drools, seria bastante relevante a existência de uma funcionalidade básica de auto-completar (KARAM, 2012) para ser utilizada durante a implementação das regras.

Considerando a limitação anterior, é possível pontuar também que os usuários do sistema deverão conhecer bem o formato de escrita das regras do Drools para utilizar a ferramenta. O usuário deverá entender minimamente como funciona um motor de regras (máquina de inferência). As regras que gerarem recursão devem ser tratadas pelo desenvolvedor. E, em casos de conflitos das regras, o desenvolvedor tem que escolher qual tratamento usar.

Outra limitação deste trabalho é a falta de conhecimento aprofundado em usabilidade para a criação de interfaces gráficas. Por isto, a interface gráfica implementada e descrita no módulo 4.4.4 ainda é bastante simplória e básica. É importante, ao se institucionalizar esta solução, ter um esforço no sentido de melhorar a usabilidade da mesma.

## 6.3 TRABALHOS FUTUROS

Como trabalhos futuros, indicamos as seguintes atividades:

- Criar regras mais genéricas para permitir que qualquer tipo de texto seja avaliado em relação a sua automação no *framework* de desenvolvimento de testes automáticos para dispositivos móveis.

- Reescrever de forma automática ou semi-automática os CTs de forma a permitir a classificação de forma mais granular. Como mencionado anteriormente, isso permitiria a separação dos passos dos casos de teste classificados como automático para aumentar a contribuição com automação de testes.

- Verificar a possibilidade de realizar também automaticamente a inferência de regras à medida que funcionalidades novas forem desenvolvidas no ambiente de desenvolvimento de testes automáticos, ou mesmo que um acessório necessário para automação/execução do caso de teste esteja disponível.

- Analisar e buscar técnicas complementares para os 11% dos casos que caíram na categoria indesejada. Apesar dos resultados serem animadores, é desejável melhorá-los o

máximo possível. Uma possibilidade seria validar uma combinação de técnicas de aprendizagem de máquina e engenharia do conhecimento ou realizar algum tipo de marcação específica para esses casos.

- Realizar um experimento controlado para avaliar mais detalhadamente quais fatores podem ter influenciado o resultado além da ação da própria ferramenta, por exemplo: grupos de testes selecionados, balanceamento desses grupos, dedicação dos especialistas, os times e áreas de teste, entre outros. Em um estudo mais completo, estes efeitos poderiam ser controlados de forma a capturar um valor estatisticamente mais associado a aplicação da ferramenta.

- Outro estudo bastante interessante seria comparar os resultados desta dissertação com abordagens de aprendizagem de máquina. Registrar quais seriam os resultados ao utilizar algumas das técnicas descritas anteriormente (KNN, SVM e Naïve Bayes). Checar se a diferença entre elas é significativa, bem como comparar o esforço para executá-las. Antes deste trabalho, outro pesquisador deu início ao estudo de uma solução utilizando abordagens de aprendizagem de máquina. Infelizmente, dado aos resultados insatisfatórios, o estudo foi abandonado sem maiores registros.

- Permitir *features* não habilitadas no IDE, como por exemplo, o auto-complemento de código.

- Avaliar e propor melhorias para as escolhas das tecnologias utilizadas (Django, Python, JIRA, etc), a arquitetura do sistema criado e a usabilidade do DRL TC Classifier.

- Validar o experimento utilizando outras linguagens de representação de regras de negócios visando comparar a representatividade e resultados alcançados.

- Considerar atualizar o gerenciador de regras no DRL TC Classifier, para que ele valide estaticamente as regras implementadas, tal qual é realizado no IDE proposto para o Drools.

# REFERÊNCIAS


Abbas, M.; Rauf, A.; Saadatmand, M.; Enoiu, E. P.; Sundmark, D. Keywords-based test categorization for extra-functional properties. In: *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. [S.l.: s.n.], 2020. p. 153–156.

ABEL, M.; FIORINI, S. Uma revisão da engenharia do conhecimento: Evolução, paradigmas e aplicações. *International Journal of Knowledge Engineering and Management (IJKEM)*, v. 2, p. 1–35, 01 2013.

AIKEN, L. S.; WEST, S. G.; RENO, R. R. *Multiple regression: Testing and interpreting interactions*. [S.l.]: Sage, 1991.

ARNOLD, K.; GOSLING, J.; HOLMES, D.; HOLMES, D. *The Java programming language*. [S.l.]: Addison-wesley Reading, 2000. v. 2.

ATLASSIAN. *Atlassian Products, c2020. Página inicial*. 2020. Disponível em: <https://www.atlassian.com/software/>. Acesso em: 30 mar. 2020.

AYORINDE, I.; AKINKUNMI, B. Application of first-order logic in knowledge based systems. *Afr J. of Comp & ICTs*, Vol 6, p. 45 – 52, 01 2013.

BAILER-JONES, D. *Scientific Models in Philosophy of Science*. University of Pittsburgh Press, 2009. ISBN 9780822971238. Disponível em: <https://books.google.com.br/books?id=avur50J7UecC>.

BALI, M. *Drools JBoss Rules 5.0 Developer's Guide*. [S.l.]: Packt Publishing, 2009. ISBN 1847195644.

BEIZER, B. *Software Testing Techniques (2nd Ed.)*. USA: Van Nostrand Reinhold Co., 1990. ISBN 0442206720.

BERNARDO, P. C.; KON, F. A importância dos testes automatizados. *Engenharia de Software Magazine*, v. 1, n. 3, p. 54–57, 2008.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: guia do usuário*. [S.l.]: Elsevier Brasil, 2006.

BROWNE, P. *JBoss Drools Business Rules: Capture, Automate, and Reuse Your Business Processes in a Clear English Language that Your Computer Can Understand*. Packt Publishing, 2009. (From technologies to solutions). ISBN 9781847196064. Disponível em: <https://books.google.com.br/books?id=xHtYmAEACAAJ>.

BYRD, T. A. Expert systems implementation. *Industrial Management & data systems*, MCB UP Ltd, 1995.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, p. 1–29, 2009.

CARBONARA, L.; SLEEMAN, D. Effective and efficient knowledge base refinement. *Machine Learning*, v. 37, p. 143–181, 11 1999.

CARMELLO, E. *Resiliencia: a transformacao como ferramenta para construir empresas de valor*. [S.l.]: Ed. Gente, 2008.

CARTER, J. Book reviews : Evaluation thesaurus (4th ed.), by michael scriven. newbury park: Sage publications, 1991, 391 pp. *Evaluation Practice*, v. 15, n. 1, p. 109–110, 1994. Disponível em: <https://doi.org/10.1177/109821409401500117>.

CHURCH, K. W.; HANKS, P. Word association norms, mutual information, and lexicography. *Computational linguistics*, MIT Press, v. 16, n. 1, p. 22–29, 1990.

CONHECIMENTO. *Conhecimento, c2009.* 2009. Disponível em: <https://www.dicio.com.br/conhecimento/>. Acesso em: 30 mar. 2020.

COOKE, N. M.; MCDONALD, J. E. The application of psychological scaling techniques to knowledge elicitation for knowledge-based systems. *International Journal of Man-Machine Studies*, Elsevier, v. 26, n. 4, p. 533–550, 1987.

COSTA, M. G. et al. Estratégia de automação em testes: requisitos, arquitetura e acompanhamento de sua implantação. [sn], 2006.

COUNCILL, W. T. Third-party testing and the quality of software components. *IEEE software*, IEEE, v. 16, n. 4, p. 55–57, 1999.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967.

DALAL, M.; ZAVERI, M. Automatic text classification: A technical review. *International Journal of Computer Applications*, v. 28, 08 2011.

DAMASCENO, M. Introdução a mineração de dados utilizando o weka. *shorturl.at/eyCJ4, Acesso: 30 de jun. de 2020*, 2020.

DARAI, D.; SINGH, S.; BISWAS, S. Knowledge engineering-an overview. In: *Knowledge Engineering-an overview*. [S.l.: s.n.], 2010.

DEPARKES, D. *Machine Learning vs Rules Systems*. 2017. Disponível em: <https://deparkes.co.uk/2017/11/24/machine-learning-vs-rules-systems/>.

DHAR, A.; DASH, N. S.; ROY, K. An innovative method of feature extraction for text classification using part classifier. In: MINZ, S.; KARMAKAR, S.; KHARB, L. (Ed.). *Information, Communication and Computing Technology*. Singapore: Springer Singapore, 2019. p. 131–138. ISBN 978-981-13-5992-7.

DUDA, R. O. Knowledge-based expert systems for the age to come. *Byte Mag.*, v. 238, 1981.

DUDA, R. O.; SHORTLIFFE, E. H. Expert systems research. *Science*, American Association for the Advancement of Science, v. 220, n. 4594, p. 261–268, 1983.

FARRINGTON-DARBY, T.; WILSON, J. R. The nature of expertise: A review. *Applied Ergonomics*, v. 37, n. 1, p. 17 – 32, 2006. ISSN 0003-6870. Special Issue: Fundamental Reviews. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0003687005001262>.

FEIGENBAUM, E.; MCCORDUCK, P. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. USA: Addison-Wesley Longman Publishing Co., Inc., 1983. ISBN 0201115190.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A. Overview and analysis of methodologies for building ontologies. *The knowledge engineering review*, Cambridge University Press, v. 17, n. 2, p. 129–156, 2002.

FEWSTER, M.; GRAHAM, D. *Software test automation*. [S.l.]: Addison-Wesley Reading, 1999.

FINIZOLA, A. B. S. *Dogfooding Analysis System: um sistema de análise de feedbacks de dogfooding para auxiliar as atividades de Testes Exploratórios*. 2019. Mestrado em Ciências da Computação.

FORGY, C. L. Rete: A fast algorithm for the many pattern/many object pattern match problem. In: *Readings in Artificial Intelligence and Databases*. [S.l.]: Elsevier, 1989. p. 547–559.

GANDHI, O. P.; KANG, G. Some present problems and a proposed experimental phantom for sar compliance testing of cellular telephones at 835 and 1900 mhz. *Physics in Medicine & Biology*, IOP Publishing, v. 47, n. 9, p. 1501, 2002.

GARDNER, K. M.; RUSH, A. R.; CRIST, M.; KONITZER, R.; TEEGARDEN, B. *Cognitive patterns: Problem-solving frameworks for object technology*. [S.l.]: Cambridge University Press, 1998. v. 14.

GINSBERG, A. *Automatic Refinement of Expert System Knowledge Bases*. Pitman, 1988. (Research notes in artificial intelligence). ISBN 9780273087946. Disponível em: <https://books.google.com.br/books?id=E3YZAQAAIAAJ>.

GREFENSTETTE, G. *Explorations in automatic thesaurus discovery*. [S.l.]: Springer Science & Business Media, 2012. v. 278.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, v. 43, n. 5, p. 907 – 928, 1995. ISSN 1071-5819. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1071581985710816>.

GUIDA, G.; TASSO, C. *Design and Development of Knowledge-Based Systems: From Life Cycle to Methodology*. USA: John Wiley & Sons, Inc., 1995. ISBN 0471928089.

HAO, L.; HAO, L. Automatic identification of stop-words in chinese text classification. In: IEEE. *2008 International conference on computer science and software engineering*. [S.l.], 2008. v. 1, p. 718–722.

HARMELEN, F. V.; LIFSCHITZ, V.; PORTER, B. *Handbook of knowledge representation*. [S.l.]: Elsevier, 2008.

HAROLD, E. R. *XML: Extensible Markup Language*. 1st. ed. USA: IDG Books Worldwide, Inc., 1998. ISBN 0764531999.

HARROLD, M. J. Testing: A roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2000. (ICSE '00), p. 61–72. ISBN 1581132530. Disponível em: <https://doi.org/10.1145/336512.336532>.

HART, A. *Knowledge acquisition for expert systems*. [S.l.], 1986.

HAT, I. R. *Drools Documentation*. 2020. Disponível em: <https://docs.jboss.org/drools/release/7.35.0.Final/drools-docs/html_single/>. Acesso em: 30 mar. 2020.

HAT, I. R. *Drools Downloads*. 2020. Disponível em: <https://www.drools.org/download/download.html>. Acesso em: 30 mar. 2020.

HEINZLE, R. et al. Protótipo de uma ferramenta para criação de sistemas especialistas baseados em regras de produção. 1995.

HOPPE, T.; MESEGUER, P. Vvt terminology: a proposal. *IEEE Expert*, IEEE, p. 48–55, 1993.

IKONOMAKIS, E.; KOTSIANTIS, S.; TAMPAKAS, V. Text classification: a recent overview. In: WORLD SCIENTIFIC AND ENGINEERING ACADEMY AND SOCIETY (WSEAS). *Proceedings of the 9th WSEAS International Conference on Computers*. [S.l.], 2005. p. 1–6. ISBN 960-8457-29-7.

JACKSON, P. *Introduction to Expert Systems*. 3rd. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1998. ISBN 0201876868.

JBOSS. *Drools Expert User Guide, c2011.* 2011. Disponível em: <https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/>. Acesso em: 30 mar. 2020.

JOHNSON, D. E.; OLES, F. J.; ZHANG, T.; GOETZ, T. A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, IBM, v. 41, n. 3, p. 428–437, 2002.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, MCB UP Ltd, 1972.

JONES, K. S. Idf term weighting and ir research lessons. *Journal of documentation*, Emerald Group Publishing Limited, 2004.

JSON. *Introducing JSON.* 2000. Disponível em: <https://www.json.org/json-en.html>. Acesso em: 02 fev. 2020.

KANER, C. Exploratory testing. In: *Quality assurance institute worldwide annual software testing conference*. [S.l.: s.n.], 2006.

KARAM, J. F. *Multi-directional auto-complete menu*. [S.l.]: Google Patents, 2012. US Patent 8,332,748.

KIM, S.-B.; HAN, K.-S.; RIM, H.-C.; MYAENG, S. H. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, IEEE, v. 18, n. 11, p. 1457–1466, 2006.

KNAUF, R.; PHILIPPOW, I.; GONZALEZ, A.; JANTKE, K.; SALECKER, D. System refinement in practice - using a formal method to modify real-life knowledge. In: *System Refinement in Practice - Using a Formal Method to Modify Real-Life Knowledge*. [S.l.: s.n.], 2002. p. 216–220.

KOOCHAKZADEH, N.; ALHAJJ, R. Social network analysis in software testing to categorize unit test cases based on coverage information. In: *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*. USA: IEEE Computer Society, 2011. (HPCC '11), p. 412–416. ISBN 9780769545387. Disponível em: <https://doi.org/10.1109/HPCC.2011.60>.

KORDE, V.; MAHENDER, C. N. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, v. 3, p. 85–99, 03 2012.

KUMAR, N.; PATIL, D. D.; WADHAI, V. M. Rule based programming with drools. *International Journal of Computer Science and Information Technologies*, Citeseer, v. 2, n. 3, p. 1121–1126, 2011.

LANGLEY, P.; SAGE, S. Induction of selective bayesian classifiers. In: MANTARAS, R. L. de; POOLE, D. (Ed.). *Uncertainty Proceedings 1994*. San Francisco (CA): Morgan Kaufmann, 1994. p. 399 – 406. ISBN 978-1-55860-332-5. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781558603325500559>.

LANGLEY, P.; SIMON, H. A. Applications of machine learning and rule induction. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 11, p. 54–64, nov. 1995. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/219717.219768>.

Larsen, H. L.; Yager, R. R. The use of fuzzy relational thesauri for classificatory problem solving in information retrieval and expert systems. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 1, p. 31–41, 1993.

LEE, H.; ONG, H.-l.; QUEK, L.-h. Exploiting visualization in knowledge discovery. *Proceedings of theFirst International Conference on Knowledge Discov-ery and Data Mining, Montreal, Quebec, 1995, 198–203*, 04 2001.

LEOPOLD, E.; KINDERMANN, J. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, Springer, v. 46, n. 1-3, p. 423–444, 2002.

LEWIS, H. R.; PAPADIMITRIOU, C. H. Elements of the theory of computation. *SIGACT News*, Association for Computing Machinery, New York, NY, USA, v. 29, n. 3, p. 62–78, set. 1998. ISSN 0163-5700. Disponível em: <https://doi.org/10.1145/300307.1040360>.

LI, K.; ZHANG, Z.; KOU, J. Breeding software test data with genetic-particle swarm mixed algorithm. *Journal of computers*, Academy Publisher, PO Box 40 Oulu 90571 Finland, v. 5, n. 2, p. 258–265, 2010.

Lima, C. A. C.; Santos, I. V.; Barros, F. A.; Mota, A. C. Spt: A text mining process to extract relevant areas from sw documents to exploratory tests. In: *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.: s.n.], 2018. p. 254–259.

LORENA, A.; CARVALHO, A. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada; Vol. 14, No 2 (2007); 43-67*, v. 14, 12 2007.

LU, P. J. *Knowledge-Based Systems*. Elsevier, 2020. Disponível em: <https://www.journals.elsevier.com/knowledge-based-systems>. Acesso em: 30 mar. 2020.

MA, Y.-S.; OH, S.-U.; BAE, D.-H.; KWON, Y.-R. Framework for third party testing of component software. In: IEEE. *Proceedings Eighth Asia-Pacific Software Engineering Conference*. [S.l.], 2001. p. 431–434.

MANNING, C. D.; MANNING, C. D.; SCHÜTZE, H. *Foundations of statistical natural language processing*. [S.l.]: MIT press, 1999.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to information retrieval*. [S.l.]: Cambridge university press, 2008.

MARR, B.; SCHIUMA, G.; NEELY, A. Intellectual capital–defining key performance indicators for organizational knowledge assets. *Business Process Management Journal*, Emerald Group Publishing Limited, 2004.

MICHIE, D.; SPIEGELHALTER, D. J.; TAYLOR, C. et al. Machine learning. *Neural and Statistical Classification*, Technometrics, v. 13, n. 1994, p. 1–298, 1994.

MILTON, N. *Knowledge Acquisition in Practice: A Step-by-Step Guide*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2007. ISBN 9781846288609.

MILTON, N.; CLARKE, D.; SHADBOLT, N. Knowledge engineering and psychology: Towards a closer relationship. *International Journal of Human-Computer Studies*, v. 64, n. 12, p. 1214 – 1229, 2006. ISSN 1071-5819. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1071581906001212>.

MITCHELL, T. M. *Machine Learning*. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

MUCCINI, H.; FRANCESCO, A.; ESPOSITO, P. Software testing of mobile applications: Challenges and future research directions. *2012 7th International Workshop on Automation of Software Test, AST 2012 - Proceedings*, 06 2012.

MURPHY, G. C.; KERSTEN, M.; FINDLATER, L. How are java software developers using the elipse ide? *IEEE software*, IEEE, v. 23, n. 4, p. 76–83, 2006.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing*. [S.l.]: John Wiley & Sons, 2011.

ODEGARD, D. Models and metaphors. *Philosophy*, Cambridge University Press, v. 39, n. 150, p. 349–356, 1964.

OLIVEIRA, K. M. d. *Avaliação da qualidade de sistemas especialistas*. 1995. 170 p. Disponível em: <https://www.cos.ufrj.br/uploadfile/1368036712.pdf>.

PARGAS, R. P.; HARROLD, M. J.; PECK, R. R. Test-data generation using genetic algorithms. *Software testing, verification and reliability*, Wiley Online Library, v. 9, n. 4, p. 263–282, 1999.

PARK, Y.; BYRD, R. J.; BOGURAEV, B. Automatic glossary extraction: beyond terminology identification. In: *COLING 2002: The 19th International Conference on Computational Linguistics*. [S.l.: s.n.], 2002.

PARVEEN, T.; TILLEY, S.; GONZALEZ, G. A case study in test management. In: . [S.l.: s.n.], 2007. p. 82–87.

PATEL, S. *Chapter 2: SVM (Support Vector Machine). Theory. Machine Learning 101*. 2017. Disponível em: <https://bit.ly/2pFAPFo>.

PERRY, W. E. *Effective methods for software testing: Includes complete guidelines, checklists, and templates*. [S.l.]: John Wiley & Sons, 2007.

PORTER, M. F. et al. An algorithm for suffix stripping. *Program*, Citeseer, v. 14, n. 3, p. 130–137, 1980.

PRANCKEVICIUS, T.; MARCINKEVICIUS, V. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, v. 5, 01 2017.

PY, M. X. Sistemas especialistas: uma introdução. *Porto Alegre: Instituto de informática, Universidade Federal do Rio Grande do Sul*, 2009.

RADIGAN, D. *JQL: The most flexible way to search Jira*. Atlassian, 2019. Disponível em: <https://www.atlassian.com/blog/jira-software/jql-the-most-flexible-way-to-search-jira-14>. Acesso em: 30 mar. 2020.

RODRIGUES, J. P. *Sistemas Inteligentes Híbridos para Classificação de Texto*. 2009. Mestrado em Ciência da Computação.

ROIGER, R. J. *Data mining: a tutorial-based primer*. [S.l.]: Chapman and Hall/CRC, 2017.

ROSSUM, G. V.; JR, F. L. D. *Python tutorial*. [S.l.]: Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

ROUSE, M. *What is knowledge-based systems (KBS)? - Definition from WhatIs.com*. TechTarget, 2018. Disponível em: <https://searchcio.techtarget.com/definition/knowledge-based-systems-KBS>.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach. Tradução por Regina Macedo.* 3rd. ed. [S.l.]: Prentice Hall Press, 2013. ISBN 0136042597.

RUSSELL, S.; NORVIG, P. *Inteligência artificial: Tradução da 3a Edição*. Elsevier Brasil, 2014. ISBN 9788535251418. Disponível em: <https://books.google.com.br/books?id=BsNeAwAAQBAJ>.

SAMMUT, C.; WEBB, G. I. *Encyclopedia of machine learning*. [S.l.]: Springer Science & Business Media, 2011.

SCHELLER, T.; KÜHN, E. Automated measurement of api usability: The api concepts framework. *Information and Software Technology*, Elsevier, v. 61, p. 145–162, 2015.

SCHREIBER, G.; AKKERMANS, H.; ANJEWIERDEN, A.; HOOG, R.; SHADBOLT, N.; VELDE, W.; WIELINGA, B. *Knowledge Engineering and Management - The CommonKADS Methodology*. [S.l.]: Mit Press, 2001. v. 24.

SCHWITTER, R. Controlled natural languages for knowledge representation. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. USA: Association for Computational Linguistics, 2010. (COLING '10), p. 1113–1121.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 34, n. 1, p. 1–47, 2002.

SHIMAGAKI, J.; KAMEI, Y.; UBAYASHI, N.; HINDLE, A. Automatic topic classification of test cases using text mining at an android smartphone vendor. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2018. (ESEM '18). ISBN 9781450358231. Disponível em: <https://doi.org/10.1145/3239235.3268927>.

SHORTLIFFE, E. H.; BUCHANAN, B. G. *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. [S.l.]: Addison-Wesley Publishing Company, 1985.

SIMONS, P. Parts : A study in ontology. *Revue de Métaphysique et de Morale*, Presses Universitaires de France, v. 2, p. 277–279, 1999.

SMITH, J.; HARRÉ, R.; LANGENHOVE, L. van. *Rethinking methods in psychology*. Sage Publications, 1995. (Rethinking Psychology Series). ISBN 9780803977327. Disponível em: <https://books.google.com.br/books?id=ZvYMAQAAMAAJ>.

SOTTARA, D.; MELLO, P.; PROCTOR, M. A configurable rete-oo engine for reasoning with different types of imperfect information. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 22, n. 11, p. 1535–1548, 2010.

SPENCER, D.; WARFEL, T. Card sorting: a definitive guide. *Boxes and arrows*, v. 2, p. 1–23, 2004.

STÄRK, R. F.; SCHMID, J.; BÖRGER, E. *Java and the Java virtual machine: definition, verification, validation*. [S.l.]: Springer Science & Business Media, 2012.

SUPHAPALA, P.; LEELANUNTAKUL, U.; NGAMSAOWAROJ, N.; SOPHATSATHIT, P. Test case classification using category-partition finite state machine. In: . [S.l.: s.n.], 1970.

TIZZEI, L. P. et al. Uma infra-estrutura de suporte a evolução para repositorios de componentes. [sn], 2007.

TURBAN, E.; ARONSON, J. E.; LIANG, T.-P.; SHARDA, R. *Decision Support and Business Intelligence Systems (8th Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN 0131986600.

TUSCHLING, O. Software test automation. *White paper*, 2008.

VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 2013.

VAPNIK, V. N. An overview of statistical learning theory. *IEEE transactions on neural networks*, IEEE, v. 10, n. 5, p. 988–999, 1999.

WAGNER, W.; CHUNG, Q.; NAJDAWI, M. The impact of problem domains and knowledge acquisition techniques: a content analysis of p/om expert system case studies. *Expert Systems with Applications*, v. 24, n. 1, p. 79 – 86, 2003. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417402000854>.

WAGNER, W.; NAJDAWI, M.; CHUNG, Q. B. Selection of knowledge acquisition techniques based upon the problem domain characteristics of production and operations management expert systems. *Expert Systems*, v. 18, p. 76 – 87, 05 2001.

WERNECK, V. M. B. *Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento*. Tese (Doutorado) — UFRJ, 1995. Disponível em: <https://www.cos.ufrj.br/index.php/pt-BR/publicacoes-pesquisa/details/15/1724>.

WIKIMEDIA. Resiliência (psicologia). In: _____. *Wikipédia: a enciclopédia livre*. Wikimedia, 2020. Disponível em: <https://pt.wikipedia.org/wiki/Resiliencia_(psicologia)>. Acesso em: 24 mai 2020.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, ACM New York, NY, USA, v. 31, n. 1, p. 76–77, 2002.

ZHANG, W.; YOSHIDA, T.; TANG, X. Text classification using multi-word features. In: *Text classification using multi-word features*. [S.l.: s.n.], 2007. p. 3519–3524.

ZHANG, W.; YOSHIDA, T.; TANG, X. Tfidf, lsi and multi-word in information retrieval and text categorization. In: *TFIDF, LSI and Multi-word in Information Retrieval and Text Categorization*. [S.l.: s.n.], 2008. p. 108 – 113.

ZYBIN, R.; KULIAMIN, V. V.; PONOMARENKO, A. V.; RUBANOV, V. V.; CHERNOV, E. Automation of broad sanity test generation. *Programming and Computer Software*, Springer, v. 34, n. 6, p. 351–363, 2008.

# APÊNDICE A – REGRAS DRL

```
package com.ematos.tcclassifier

import enums.RuleEnum
import enums.TestCaseAutoClassificationEnum
import com.ematos.tcclassifier.model.EvaluatedString
import com.ematos.tcclassifier.model.DTestCase

// Begin of DRL Rules for DTestCase and its test area
rule "Test Type Sanity"
  no-loop true
  lock-on-active true
    when
        dtc : DTestCase( labels matches ".*(?i)(sanity).*")
    then
        System.out.println("'Test Type Sanity' rule");
        dtc.addTestArea("SANITY");
        update(dtc);
end

rule "Test Type Regression"
  no-loop true
  lock-on-active true
    when
        dtc : DTestCase( labels matches ".*(?i)(regression).*" || labels matches "
            .*(?i)(_reg).*" )
    then
        System.out.println("'Test Type Regression' rule");
        dtc.addTestArea("REGRESSION");
        update(dtc);
end

rule "Test Type KPI"
  no-loop true
  lock-on-active true
    when
        dtc : DTestCase( key matches ".*(?i)(kpi).*" || labels matches ".*(?i)(kpi)
            .*")
    then
        System.out.println("'Test Type KPI' rule");
        dtc.addTestArea("KPI");
        update(dtc);
end

rule "Test Type CoreApps"
```

```
43    no-loop true
      lock-on-active true
45      when
            dtc : DTestCase( labels matches ".*(?i)(core_apps).*" ||  key matches ".*(?i
                )(coreapps).*" )
47      then
            System.out.println("'Test Type CoreApps' rule");
49          dtc.addTestArea("CORE_APPS");
            update(dtc);
51  end

53  rule "Test Type Framework"
      no-loop true
55    lock-on-active true
        when
57          dtc : DTestCase( labels matches ".*(?i)(framework).*" )
        then
59          System.out.println("'Test Type Framework' rule");
            dtc.addTestArea("FRAMEWORK");
61          update(dtc);
    end
63
    rule "Test Type GMS"
65    no-loop true
      lock-on-active true
67      when
            dtc : DTestCase( labels matches ".*(?i)(gms).*" )
69      then
            System.out.println("'Test Type GMS' rule");
71          dtc.addTestArea("GMS");
            update(dtc);
73  end

75  rule "Test Type GMS_Reg"
      no-loop true
77    lock-on-active true
        when
79          dtc : DTestCase( labels matches ".*(?i)(gms_reg).*" )
        then
81          System.out.println("'Test Type GMS_Reg' rule");
            dtc.addTestArea("GMS_REG");
83          dtc.removeTestArea("GMS");
            dtc.removeTestArea("REGRESSION");
85          update(dtc);
    end
87
    rule "Test Type MME"
```

```
89    no-loop true
      lock-on-active true
91      when
            dtc : DTestCase( key matches ".*(?i)(mme.*media).*" || labels matches ".*(
                MME).*" )
93      then
            System.out.println("'Test Type MME' rule");
95          dtc.addTestArea("MME");
            update(dtc);
97  end

99  rule "Test Type Kernel"
      no-loop true
101   lock-on-active true
      when
103         dtc : DTestCase( key matches ".*(?i)(kernel).*" || labels matches ".*(?i)(
                kernel).*" )
      then
105         System.out.println("'Test Type Kernel' rule");
            dtc.addTestArea("KERNEL");
107         update(dtc);
    end
109
    rule "Test Type WISL"
111   no-loop true
      lock-on-active true
113   when
            dtc : DTestCase( key matches ".*(?i)(wisl).*" || labels matches ".*(?i)(wisl
                ).*" )
115   then
            System.out.println("'Test Type WISL' rule");
117         dtc.addTestArea("WISL");
            update(dtc);
119 end


121
    // Begin of DRL Rules for DTestCase and its automatability
123
    rule "AMAZON_ACCOUNT"
125   no-loop true
      lock-on-active true
127   when
            dtc : DTestCase( testAreas contains "SANITY", rules not contains "
                AMAZON_ACCOUNT", tcString matches ".*((amazon account)|(amazon widget)|
                kindle).*")
129 //        &&
    //        , testAreas matches "SANITY"
```

```
131     then
            System.out.println("'amazon account | kindle' rule");
133         dtc.setAutoClassificationEnum(TestCaseAutoClassificationEnum.MANUAL);
            dtc.addRule("AMAZON_ACCOUNT");
135         update(dtc);
end
137
// Begin of DRL Rules for EvaluatedString
139
rule "QUALITY"
141   no-loop true
      lock-on-active true
143     when
            es : EvaluatedString( value matches ".*quality.*")
145     then
            System.out.println("'image/video quality, sound/audio quality, call quality,
                 verify/check' rule");
147         es.addEvaluation("QUALITY");
end
149
rule "FINGERPRINT"
151   no-loop true
      lock-on-active true
153     when
            es : EvaluatedString( value matches ".*fingerprint.*")
155     then
            System.out.println("'fingerprint, finger' rule");
157         es.addEvaluation("FINGERPRINT");
end
159
rule "FACIAL"
161   no-loop true
      lock-on-active true
163     when
            es : EvaluatedString( value matches ".* facial.*")
165     then
            System.out.println("'facial recognition' rule");
167         es.addEvaluation("FACIAL");
end
169
rule "GESTURE"
171   no-loop true
      lock-on-active true
173     when
            es : EvaluatedString( value matches ".*(rotate|chop|twist|shake|motovoice|
                moto voice|motokey|moto key|gesture|moto action|motoaction).*")
175     then
```

```
        System.out.println("'rotate, chop, twist, shake the phone, motovoice, moto
            voice, moto key, gesture' rule");
177         es.addEvaluation("GESTURE");
end
179
rule "MOVE_DEVICE"
181   no-loop true
  lock-on-active true
183     when
        es : EvaluatedString( value matches ".*(move device to|proximity|move the
            phone out of range|move the phone out of somewhere|move the phone out to
             range|move the phone out to somewhere).*")
185     then
        System.out.println("'move to, move the host/phone out of/to range/somewhere,
             proximity interaction(s)' rule");
187         es.addEvaluation("MOVE_DEVICE");
end
189
rule "BT"
191   no-loop true
  lock-on-active true
193     when
        es : EvaluatedString( value matches ".*(headset|bluetooth|bt ).*")
195     then
        System.out.println("'call over headset bluetooth/bt/btooth' rule");
197         es.addEvaluation("BT");
end
199
rule "USB_CONNECT"
201   no-loop true
  lock-on-active true
203     when
        es : EvaluatedString( value matches ".*(usb (connect|disconnect)|plug in the
             usb).*")
205     then
        System.out.println("'usb connect/disconnect' rule");
207         es.addEvaluation("USB_CONNECT");
end
209
rule "CHROMECAST"
211   no-loop true
  lock-on-active true
213     when
        es : EvaluatedString( value matches ".*(chromecast|chrome cast).*")
215     then
        System.out.println("'chromecast/chrome cast' rule");
217         es.addEvaluation("CHROMECAST");
```

```
end

rule "HEADSET"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*(headset|jbl).*")
    then
        System.out.println("'headset, jbl' rule");
        es.addEvaluation("HEADSET");
end

rule "DTV"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*(dtv|digital tv|tv program).*")
    then
        System.out.println("'dtv/digital tv/tv program' rule");
        es.addEvaluation("DTV");
end

rule "INSERT_REMOVE_SIM"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*((insert|remove|switch|take out the|
            tak out) sim).*")
    then
        System.out.println("'Insert/remove/switch SIM' rule");
        es.addEvaluation("INSERT_REMOVE_SIM");
end

rule "INSERT_REMOVE_SDCARD"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*((insert|remove) (sdcard|sd card|
            external storage)).*")
    then
        System.out.println("'Insert/remove SDCard, SD Card, external storage' rule")
            ;
        es.addEvaluation("INSERT_REMOVE_SDCARD");
end

rule "CHARGER"
  no-loop true
```

```
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*charger.*")
    then
        System.out.println("'Insert/remove/Plug charger, battery charger' rule");
        es.addEvaluation("CHARGER");
end

rule "FM_RADIO"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*(fm radio|fmradio).*")
    then
        System.out.println("'FM Radio' rule");
        es.addEvaluation("FM_RADIO");
end

rule "SI_MESSAGE"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*(si message|xml provisioning|sms_cin)
            .*")
    then
        System.out.println("'SI message, XML provisioning, SMS_CIN.exe' rule");
        es.addEvaluation("SI_MESSAGE");
end

rule "FDN"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*fdn.*")
    then
        System.out.println("'FDN' rule");
        es.addEvaluation("FDN");
end

rule "N_PHONES"
  no-loop true
  lock-on-active true
    when
        es : EvaluatedString( value matches ".*(7 phones|phone C|phone D|group
            message|all recipients).*")
    then
        System.out.println("'n phones required, phone A,B,C,D,E.., support device(s)
```

```
            ' rule");
307         es.addEvaluation("N_PHONES");
end
309
rule "ROAMING"
311   no-loop true
  lock-on-active true
313     when
            es : EvaluatedString( value matches ".*roaming.*")
315     then
            System.out.println("'roaming' rule");
317         es.addEvaluation("ROAMING");
end
319
rule "TTY"
321   no-loop true
  lock-on-active true
323     when
            es : EvaluatedString( value matches ".*tty.*")
325     then
            System.out.println("'TTY' rule");
327         es.addEvaluation("TTY");
end
329
rule "ANDROID_N_O_P"
331   no-loop true
  lock-on-active true
333     when
            es : EvaluatedString( value matches "(.*android [n|o|p|7|8|9] ).*")
335     then
            System.out.println("'Android versions' rule");
337         es.addEvaluation("ANDROID_N_O_P");
end
339
rule "DTMF"
341   no-loop true
  lock-on-active true
343     when
            es : EvaluatedString( value matches ".*dtmf.*")
345     then
            System.out.println("'DTMF' rule");
347         es.addEvaluation("DTMF");
end
349
rule "POWER_KEY"
351   no-loop true
  lock-on-active true
```

```
353     when
            es : EvaluatedString( value matches ".*(power key|powerkey).*")
355     then
            System.out.println("'power key / ' rule");
357         es.addEvaluation("POWER_KEY");
end
359
rule "BRIGHTNESS"
361   no-loop true
      lock-on-active true
363     when
            es : EvaluatedString( value matches ".*brightness.*")
365     then
            System.out.println("'brightness / ' rule");
367         es.addEvaluation("BRIGHTNESS");
end
369
rule "GOOGLE_VOICE"
371   no-loop true
      lock-on-active true
373     when
            es : EvaluatedString( value matches ".*(google voice|google command|google
                assistant|google voice search|always speak).*")
375     then
            System.out.println("'Google Voice / Google commands / Google Assistant
                trigger / Google Voice Search / Always speak' rule");
377         es.addEvaluation("GOOGLE_VOICE");
end
379
rule "MOD"
381   no-loop true
      lock-on-active true
383     when
            es : EvaluatedString( value matches ".* mod .*")
385     then
            System.out.println("'MOD interactions' rule");
387         es.addEvaluation("MOD");
end
389
rule "CALL_FORWARDING"
391   no-loop true
      lock-on-active true
393     when
            es : EvaluatedString( value matches ".*(call forwarding|forward call|call
                forward).*")
395     then
            System.out.println("'call forwarding' rule");
```

```
397         es.addEvaluation("CALL_FORWARDING");
end
399
rule "VOICEMAIL"
401   no-loop true
  lock-on-active true
403     when
          es : EvaluatedString( value matches ".*(voicemail|voice mail).*")
405     then
          System.out.println("'voicemail / voice mail' rule");
407       es.addEvaluation("VOICEMAIL");
end
409
rule "FULL_STORAGE"
411   no-loop true
  lock-on-active true
413     when
          es : EvaluatedString( value matches ".*full storage.*")
415     then
          System.out.println("'full storage' rule");
417       es.addEvaluation("FULL_STORAGE");
end
419
rule "FDR"
421   no-loop true
  lock-on-active true
423     when
          es : EvaluatedString( value matches ".*(fdr|factory data reset|factory reset
              |data reset).*")
425     then
          System.out.println("'FDR, Factory Data Reset, factory reset' rule");
427       es.addEvaluation("FDR");
end
429
rule "OTG"
431   no-loop true
  lock-on-active true
433     when
          es : EvaluatedString( value matches ".*otg.*")
435     then
          System.out.println("'OTG' rule");
437       es.addEvaluation("OTG");
end
439
rule "CLI"
441   no-loop true
  lock-on-active true
```

```
443     when
            es : EvaluatedString( value matches ".* cli .*")
445     then
            System.out.println("'CLI' rule");
447         es.addEvaluation("CLI");
end
449
rule "FLEXIBLE_DISPLAY"
451   no-loop true
      lock-on-active true
453     when
            es : EvaluatedString( value matches ".*(flexible (display|screen)).*")
455     then
            System.out.println("'Flexible screen/display' rule");
457         es.addEvaluation("FLEXIBLE_DISPLAY");
end
459
rule "LUMES"
461   no-loop true
      lock-on-active true
463     when
            es : EvaluatedString( value matches ".* lumens.*")
465     then
            System.out.println("'lumens' rule");
467         es.addEvaluation("LUMES");
end
469
rule "FIVE_G"
471   no-loop true
      lock-on-active true
473     when
            es : EvaluatedString( value matches ".* 5g.*")
475     then
            System.out.println("'5G' rule");
477         es.addEvaluation("FIVE_G");
end
479
rule "WEARABLE"
481   no-loop true
      lock-on-active true
483     when
            es : EvaluatedString( value matches ".*(((android|moto) watch) | (android
                wear)).*")
485     then
            System.out.println("'android|moto watch|android wear' rule");
487         es.addEvaluation("WEARABLE");
end
```

```
489
/*
491 rule "AMAZON_ACCOUNT"
  no-loop true
493   lock-on-active true
    when
495         es : EvaluatedString( value matches ".*((amazon account)|(amazon widget)|
              kindle).*")
    then
497         System.out.println("'amazon account | kindle' rule");
            es.addEvaluation("AMAZON_ACCOUNT");
499 end*/

501 rule "LONG_ALARM"
  no-loop true
503   lock-on-active true
    when
505         es : EvaluatedString( value matches ".*(mins alarm|wait for alarm).*")
    then
507         System.out.println("'mins alarm' rule");
            es.addEvaluation("LONG_ALARM");
509 end

511 rule "VOLUME"
  no-loop true
513   lock-on-active true
    when
515         es : EvaluatedString( value matches ".*((volume (up|down))|phone volume|
              volume adjustment|busy tone|can be heard|is on ear|talkback|can hear).*"
              )
    then
517         System.out.println("'Volume up/down|audio effect' rule");
            es.addEvaluation("VOLUME");
519 end

521 rule "MICROPHONE"
  no-loop true
523   lock-on-active true
    when
525         es : EvaluatedString( value matches ".*(microphone|mic ).*")
    then
527         System.out.println("'microfone, mic' rule");
            es.addEvaluation("MICROPHONE");
529 end

531 rule "BOTA"
  no-loop true
```

```
533   lock-on-active true
        when
535         es : EvaluatedString( value matches ".*bota.*")
        then
537         System.out.println("'bota' rule");
            es.addEvaluation("BOTA");
539 end

541 rule "TWO_FOUR_GHZ"
      no-loop true
543   lock-on-active true
        when
545         es : EvaluatedString( value matches ".*2.4ghz.*")
        then
547         System.out.println("'2.4ghz' rule");
            es.addEvaluation("TWO_FOUR_GHZ");
549 end

551 rule "THIRD_PARTY_APPS"
      no-loop true
553   lock-on-active true
        when
555         es : EvaluatedString( value matches ".*(prime photos|amazon music|google
                hindi|google pinyin|amazon video|google music player|quickoffice|quick
                office).*")
        then
557         System.out.println("'3rd Party Apps' rule");
            es.addEvaluation("THIRD_PARTY_APPS");
559 end

561 rule "SPECIAL_CHAR"
      no-loop true
563   lock-on-active true
        when
565         es : EvaluatedString( value matches ".*(japanese|chinese).*")
        then
567         System.out.println("'japanese, chinese caractere' rule");
            es.addEvaluation("SPECIAL_CHAR");
569 end

571 rule "MAPS_INTERACTION"
      no-loop true
573   lock-on-active true
        when
575         es : EvaluatedString( value matches ".*(zoom (in|out)|swipe the maps|map
                area available offline|offline maps).*")
        then
```

```
577         System.out.println("'Zoom in / Zoom out maps' rule");
            es.addEvaluation("MAPS_INTERACTION");
579 end

581 rule "NFC_INTERACTION"
      no-loop true
583   lock-on-active true
        when
585         es : EvaluatedString( value matches ".*(nfc interaction).*")
        then
587         System.out.println("'nfc interaction' rule");
            es.addEvaluation("NFC_INTERACTION");
589 end

591 rule "VR_VIDEO"
      no-loop true
593   lock-on-active true
        when
595         es : EvaluatedString( value matches ".*(vr video).*")
        then
597         System.out.println("'vr video' rule");
            es.addEvaluation("VR_VIDEO");
599 end

601 rule "GRAYSCALE"
      no-loop true
603   lock-on-active true
        when
605         es : EvaluatedString( value matches ".*(grayscale).*")
        then
607         System.out.println("'grayscale' rule");
            es.addEvaluation("GRAYSCALE");
609 end
    //  End of DRL Rules
```

Código Fonte 1 – Código das regras de négocio para classificação de casos de teste - DRL

# APÊNDICE B – CASOS DE TESTE EXEMPLOS EM FORMATO JSON

Neste apêndice, serão mostrados alguns exemplos de casos de teste em relação a sua classificação automática de acordo com o DRL TC Classifier. Aqui apresenta-se exemplos de casos verdadeiro positivos (VP), casos falso positivos (FP), casos falso negativos (FN) e casos verdadeiro negativos (VN). Os exemplos são apresentados no formato JSON.

## CASOS VERDADEIRO POSITIVOS (VP)


```
{   "key": "TC-001",
    "component": "CBS - Others",
    "summary": "TC - WiFi: Share Mobile network data functionality by WiFi Mobile
         hotspot method, and check at least 3 WiFi clients to browse, download or
         any data tasks, with interactions of Airplane mode, BT operations, WiFi
        signal, MT Call messaging, Or",
    "labels":["ATT_Reg", "EDA_updated", "Fi_Reg", "Latam_Reg", "Memory_2ndSource"
        , "Plat_Reg", "Retus_Reg", "Sprint_Reg", "TMO_Reg", "VZW_Reg", "
        execute_femtocell_att", "execute_femtocell_tmo", "execute_femtocell_vzw",
         "execute_live_sprint", "execute_roaming_att", "execute_roaming_tmo", "
        execute_roaming_vzw", "femtocell_att", "femtocell_tmo", "femtocell_vzw",
        "network_dependent", "roaming_execution_att", "roaming_execution_tmo", "
        roaming_execution_vzw", "sprint_live"],
    "steps": [
        {
            "comments": "",
            "expected_results": "2. All clients connected test phone WiFi Mobile
                hotspot correctly.\n3. The frequency on support device is the
                same as the hotspot band set on DUT\n\n4.1. All clients can
                access internet simultaneously, and work correctly, for browse,
                download or any data tasks.  NOTE: ATT network some time has
                issue, so that it can\u2019t support 3 clients very smoothly
                simultaneously, in this case, at least 2 client phone can access
                internet simultaneously. \n\n4.2. All in all, Phone works well,
                without ANR, FC, Panic, Tombstone and so on even for all
                interactions.\n\nIA1: Airplane mode from on to off. Test phone
                and MHS work without impact.\n\nIA2: Test phone and MHS work
                without impact, for Bluetooth co-existing operations. All clients
                 can access internet simultaneously.\n\nIA3. While WiFi signal
                from strong to weak and vice versa, test phone and MHS work
                without impact; All clients can access internet simultaneously.\n
                \nIA4. For MT Call, if operator network cannot support mobile
                data and voice simultaneously, at least test phone should not
                occur ANR, FC, Panic, Tombstone and so on. After call end, test
```

```
                phone MHS should restore correctly; and all clients can access
                internet simultaneously.\n\nIA5. For MT messaging, test phone and
                 MHS work without impact; All clients can access internet
                simultaneously.\n\nIA6. For Orientation change, test phone and
                MHS work without impact; All clients can access internet
                simultaneously.\n\nIA7. For Power key interactions, test phone
                and MHS work without impact; All clients can access internet
                simultaneously.\n\nIA8. For Volume up and down keys interactions,
                 test phone and MHS work without impact; All clients can access
                internet simultaneously.\n\nIA9. For Charger/USB cable plug and
                unplug, test phone and MHS work without impact; All clients can
                access internet simultaneously. ",
9           "initial_condition": "",
10          "step_description": "1. Share Mobile network data functionality by
                WiFi Mobile hotspot method.\n2. Connect at least other 3 clients
                via WiFi Mobile hotspot.\n3. Change the hotspot band to 2.4GHz
                /5.0 GHz and check the frequency on the support device\n4. From
                other 3 clients, check browse, download or any data tasks. Please
                 do below interactions while Mobile hotspot works:\nIA1. Airplane
                 mode from on to off.\n\nIA2. Bluetooth headset/carkit unpair/
                pair/connect/stream/control volume.\n\nIA3. WiFi signal from
                strong to weak, and vice versa.\n\nIA4. MO/MT Call.\n\nIA5. MO/MT
                 SMS and MMS.\n\nIA6. Orientation change.\n\nIA7. Power key.\n\
                nIA8. Volume up and down keys.\n\nIA9. Charger/USB cable. "},
11      {   "comments": "",
12          "expected_results": "Same as above ",
13          "initial_condition": "",
14          "step_description": "If applicable, Set DDS to SIM2 and repeat steps
                above to SIM2 "
15      }
16  ]
17 }
```



```
1  {   "key": "TC-002",
2      "component": "FT IDM/Basic functionality",
3      "summary": "TC - Device Checkin",
4      "labels": ["Sanity", "auto_cloud", "automated", "device_activation", "
           full_sanity", "smr_auto_sanity"],
5      "steps": [{
6              "comments": "http://www.epochconverter.com/\n\nEX \nhttp://dspipeline
                   .appspot.com/guide/passport?serialnumber=NADE210386\nhttp://
                   dspipeline.appspot.com/guide/passport?serialnumber=NADI210082 ",
7              "expected_results": "Expected Result:\nThe url should contain values
                   and the \"checkintimestamp\" should have a value within the range
                    from when the device booted after flashing. ",
8              "initial_condition": "Pre Condition: \nPerform FDR\nSetup Completed
                   with internet connection (wifi / mobile data) ",
```

```
            "step_description": "Test Steps:\nObtain the serial number from
                Settings->About Phone ->Status -> Serial Number\nAppend serial
                number to url and goto the mentioned url.\nCheck if it contains
                values.\nURL: http://dspipeline.appspot.com/guide/passport?
                serialnumber=<serial_number>\nEmail login mandatory to access
                above URL, use loging, don't use Gmail.\nVerify that the \"
                checkintimestamp\" has a value and it is within the range from
                when the device booted after flashing. (Convert the epoch
                checkintimestamp to human readable format before verifying) \neg.
                 \"checkintimestamp\":1461111284269.\nAfter converting the epoch
                time to Human readable format Wed, 20 Apr 2016 00:14:44.269 GMT.
                "
        }]
}
```

```
{   "key": "TC-003",
    "component": "GMS",
    "summary": "TC - AndroidWear: OTA notification check when connect Android
        watch",
    "labels": ["GMS_Reg", "Plat_Reg", "binary_leverage", "non_network_dependent"]
        ,
    "steps": [{
            "comments": "*After fix of IKSWP-9869 no OTA notification will be
                sent to watch. ",
            "expected_results": "1.Check the notification on phone\n2. No
                notification should appear on Watch*\n2.Can update system
                successfully, no crash or ANR happen.\n3.After successfully
                update to the new version, Android watch is still connect. ",
            "initial_condition": "Connect Android Watch with phone ",
            "step_description": "Receive a OTA notification from phone "
        }]
}
```

```
{   "key": "TC-004",
    "component": "Camera",
    "summary": "TC - KPI-0999 - Burst Mode (Time to capture a picture in Multi-
        shot mode)",
    "labels": ["2nd_memory_kpi", "camera_kpi", "common", "csi_kpi", "
        secondary_kpi", "soak_kpi"],
    "steps": [{
            "comments": "1. For this case, we measure the time difference \
                nbetween the time shown on PC stop watch when \ncaptured image
                from burst mode to the next image \ncapture in burst mode. Use
                the time found on capture\nimage of the stopwatch on PC to get
                delta.\n\n2. Each case repeates 5 times to get average data. \n\
                n3. After case completes, the application tested should \nbe
                swiped out of the recent apps menu ",
```

```
            "expected_results": "Start Point: \nImage captured in burst mode\n\
                nStop Point: \nNext image captured in burst mode ",
            "initial_condition": "1. Use a powerful external light in the
                background of the camera. About 1300 lumens\n\n2. Download and
                Print the following Chart https://drive.google.com/a/?usp=sharing
                \nNote: Image will be printed on long side of the paper\n\n3.
                Download and Print the following Color Checker Chart: https://
                drive.google.com/file/d/1mffiIO-1h19I6izEKbxR4PRvE9gxBlSy/view?
                usp=sharing\n\n4. Printed Charts Images above should be place
                about \n2 feet away from Device Under Test.\n\n5. Use Default
                Camera Settings.\n\n6.Place device in landscape position. ",
            "step_description": "To Measure Bust Mode please use the following
                setup\n\na. Launch online stop watch on PC and start time watch\
                nb. Place device in front of PC to capture stop watch\nStep Desc
                = 1. Launch Camera once and exit app \nusing Home Key.\n\n2.
                Place device in Landscape position.\n\n3. Relaunch Camera app\n\
                n4. Click and Press the Capture button to initiate Bust \nMode\n\
                n5. Measure the time between each image capture \nduring bust
                mode. Ignore the 1st and last image capture. Make sure to get 10
                values, so capture at least 12 image \nin burst mode. "
        }]
}
```

## CASOS FALSO POSITIVOS (FP)

```
{   "key": "TC-005",
    "component": "Call",
    "summary": "TC - Verify Instant Apps interaction with AoD and Lockscreen ",
    "labels": ["GMS_Reg", "Plat_Reg", "binary_leverage", "low_pri_auto", "
        non_network_dependent"],
    "steps": [
        {   "comments": "",
            "expected_results": "Instant app should run without any issue and
                could switch from/to other apps ",
            "initial_condition": "Device has a google account with Google Play
                Instant enabled.\nThe device has a screen lock ",
            "step_description": "Open some apps (e.g: FM Radio/GoogleMaps/DTV)
                and open a Instant App running in foreground "},
        {   "comments": "",
            "expected_results": "Notifications should be displayed properly ",
            "initial_condition": "",
            "step_description": "Lock the device and receive some notification (e
                .g: Gmail, Missed call, SMS) "},
        {   "comments": "",
            "expected_results": "No ANR/Tombstone/Crash observed ",
```

```
            "initial_condition": "",
            "step_description": "Open notifications and reply it, then select
                recent apps "},
        {   "comments": "",
            "expected_results": "Verify that it could be replied properly and
                Instant apps are present on recent apps\n[Android Q+ only]
                According to IKSWP-50375, Instant apps will only be present on
                Android Q onward ",
            "initial_condition": "",
            "step_description": "Change the lock screen to PIN/Pattern/Password
                and repeat steps above "
        }]
}
```

```
{   "key": "TC-006",
    "component": "Calling",
    "summary": "TC - Open notification curtain during MO call",
    "labels": ["AOSP_Dialer", "AOSP_Dialer_MS2", "Android_Q_updated", "CA_MS2", "
        CA_P_Update", "CA_general", "CA_network", "CoreApps_Reg", "GMS_Dialer", "
        GMS_Dialer_MS2", "auto_candidate", "auto_ide", "execute_local_network", "
        network_dependent", "network_leverage", "product_leverage"],
    "steps": [
        {   "comments": "",
            "expected_results": "The call is established ",
            "initial_condition": "- There is unread SMS on notification bar\n-
                There is Calendar notification on notification bar \n- There is
                unread Email on notification bar ",
            "step_description": "Make a call to remote party "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "In the android Q,The system gesture is fully
                gestural navigation,to go Back, swipe from either the left or
                right edge of the screen ",
            "step_description": "During the call, pull down the notification bar
                "},
        {   "comments": "",
            "expected_results": "Verify SMS is opened ",
            "initial_condition": "",
            "step_description": "Click on unread SMS on notification curtain "},
        {   "comments": "",
            "expected_results": "Verify phone returns to message screen ",
            "initial_condition": "",
            "step_description": "Click on BACK button "},
        {   "comments": "",
            "expected_results": "Verify phone goes to Homescreen and call
                continue on the background ",
            "initial_condition": "",
```

```
            "step_description": "Click on BACK button "},
        {   "comments": "",
            "expected_results": "Verify Calendar event is displayed ",
            "initial_condition": "",
            "step_description": "Pull down the notification bar and click on
                Calendar notification on notification curtain "},
        {   "comments": "",
            "expected_results": "Verify phone goes to Homescreen ",
            "initial_condition": "",
            "step_description": "Click on HOME button "},
        {   "comments": "",
            "expected_results": "Verify active call screen is displayed ",
            "initial_condition": "",
            "step_description": "Pull down the notification bar and click on
                current call on notification curtain "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "",
            "step_description": "Pull down the notification bar "},
        {   "comments": "",
            "expected_results": "Verify unread Email is displayed ",
            "initial_condition": "",
            "step_description": "Click on unread Email on notification curtain "}
                ,
        {   "comments": "",
            "expected_results": "Verify Email inbox screen is displayed ",
            "initial_condition": "",
            "step_description": "Click on BACK button "},
        {   "comments": "",
            "expected_results": "Verify phone goes to Homescreen and call
                continue on the background ",
            "initial_condition": "",
            "step_description": "Click on BACK button "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "",
            "step_description": "End the call "
        }]
}
```

```
{   "key": "TC-007",
    "component": "Contacts",
    "summary": "TC - Share My profile via Bluetooth",
    "labels": ["AOSP_Contacts", "Android_Q_updated", "CA_general", "CoreApps_Reg"
        , "GMS_Contacts", "auto_candidate", "auto_ide", "non_network_dependent"],
    "steps": [
        {   "comments": "",
```

```
            "expected_results": "",
            "initial_condition": "- On DUT, \"My profile\" is set up on Contacts
                app (Fill in all fields)\n- On Remote party, \"My profile\" is
                NOT set up on Contacts app\n- On DUT and Remote party, Bluetooth
                is enabled ",
            "step_description": "Go to Contacts app "},
        {   "comments": "",
            "expected_results": "Verify contact detail screen is displayed ",
            "initial_condition": "",
            "step_description": "Click on \"My Info\" contact "},
        {   "comments": "",
            "expected_results": "Verify pop-up apps list to be selected ",
            "initial_condition": "",
            "step_description": "Click on Options (3 dots on the right corner) ->
                 Share "},
        {   "comments": "",
            "expected_results": "The bluetooth\u00b4s list should appear to be
                selected ",
            "initial_condition": "",
            "step_description": "Select Bluetooth app "},
        {   "comments": "",
            "expected_results": "Verify both phones are connected\n\nFrom remote
                party, on status bar should appear a message with options to
                decline or to accept the incoming file ",
            "initial_condition": "",
            "step_description": "Choose a bluetooth device "},
        {   "comments": "",
            "expected_results": "On notification curtain, verify that the vcf
                file is received successfully ",
            "initial_condition": "",
            "step_description": "From remote party, click on ACCEPT option "},
        {   "comments": "",
            "expected_results": "It will appear the message \"Import contacts
                from vCard?\" with CANCEL and OK buttons ",
            "initial_condition": "",
            "step_description": "On notification curtain, click on the received
                file and then click on the vcf file "},
        {   "comments": "",
            "expected_results": "On notification curtain, verify that the vcf
                file was imported successfully ",
            "initial_condition": "",
            "step_description": "Click on OK button "},
        {   "comments": "",
            "expected_results": "Verify that the profile contact is imported with
                 all information to contacts list but it is not marked as \"My
                profile\"\n\n\"My profile\" keeps empty ",
            "initial_condition": "",
```

```
            "step_description": "From remote party, go to Contacts app and search
                for the contact imported "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "",
            "step_description": "Repeat the steps again but now from Remote Party
                to DUT "
        }]
}
```

```
{   "key": "TC-008",
    "component": "GMS",
    "summary": "TC - Hangouts - Call and messages",
    "labels": ["Sanity", "hangouts", "non_network_dependent", "platform_sanity",
       "sanity_common"],
    "steps": [
        {   "comments": "",
            "expected_results": "2. Verify a message is send to the contact\n\n4.
                Verify Hangouts video call connects succesfully, video and audio
                quality is good ",
            "initial_condition": "1.One test device and one support device. 2.
               Devices connect to WiFi, and both devices signin different Google
                accounts. ",
            "step_description": "1. On test device, launch Hangouts, start a New
               Hangouts session \n2. Select a contact that is from support
               device Google, account, send a message to the contact\n3.Make and
                receive video call  to support device Google account contact \
               nComments:  pls use wechat app if test in China "
        }]
}
```

## CASOS FALSO NEGATIVOS (FN)

```
{   "key": "TC-009",
    "component": "3rd Party App",
    "summary": "TC - Amazon apps - Launch all applications",
    "labels": ["amazon", "daily_sanity", "platform_sanity", "sanity", "
       sanity_amazon"],
    "steps": [
        {   "comments": "Please refer the correct order of the apps based on this
             cr: IKSWM-37912 ",
            "expected_results": "1. Apps can be launched successfully. No crash/
               ANR reported during and after launching of apps. ",
            "initial_condition": "",
```

```
            "step_description": "1. Launch each Amazon app in the preloaded at
                least once. (No need to sign in to each app). "
        }]
}
```

```
{   "key": "TC-010",
    "component": "GMS",
    "summary": "TC - Verify USB debugging: must be off by default",
    "labels": ["GMS3.1", "GMS4.0", "GMSCompliance", "non_network_dependent", "
        secure_hw"],
    "steps": [
        {   "comments": "Apply to the latest requirement if applicable ",
            "expected_results": "USB debugging must be off by default ",
            "initial_condition": "USE SECURE HW for this test, not using a Secure
                 HW will lead to incorrect results ",
            "step_description": "Verify \"Settings -> ... -> USB debugging\".\n\
                nIf \"Developer options\" is not displayed, go to \"Settings ->
                ... -> Build number\" and select it more than 5 times. \"
                Developer options\" should be displayed at \"Settings\". "
        }]
}
```

```
{   "key": "TC-011",
    "component": "AoD_KPI",
    "summary": "TC - KPI-0844 Phone removed from pocket/purse",
    "labels": ["2nd_display_kpi", "aod_kpi", "common", "secondary_kpi"],
    "steps": [
        {   "comments": "1. Each case repeates 5 times to get average data. \n\n2
            . After case completes, the application tested should \nbe swiped out
             of the recent apps manu.\n\n3. This case is only applicable for
            device supporting \nAOD (Always on Dipslay) ",
            "expected_results": "Start Point: \nRemove an object that covers Prox
                 & ALS sensors\n\nEnd Point: \nAs soon AOD breathing UI render
                first sign ",
            "initial_condition": "1  Screen Lock under Security Settings set to
                \"None\"\n\n2. DUT should be asleep with display off ",
            "step_description": "To simulate phone removed from pocket or purse
                and \nmake it easy to measure time follow below steps\n\n1. Cover
                 the sensors (Prox & ALS sensors) only with \nan object such as
                hand, paper, etc.. Make sure the \nsensor \nare fully convered,
                no light should be visible.\n\n3. Remove your hand or object from
                 Sensor, AOD \nbreathing initiated. "
        }]
}
```

```
{   "key": "TC-012",
    "component": "Dual Sim",
```

```
    "summary": "TC - VoWifi interactions with 2 SIM cards",
    "labels": ["Latam_Reg", "Plat_Reg"],
    "steps": [
        {   "comments": "",
            "expected_results": "DUT can camp on 4g/LTE and SIM1 can register IMS
                ",
            "initial_condition": "Insert 2 SIM cards with VoWifi enabled\nMobile
                Data is enabled to SIM1\n\nNote : TC is not applicable to DSDV ",
            "step_description": "Enable Wifi Calling for SIM1 "},
        {   "comments": "",
            "expected_results": "Voice Call (not VoWifi) is started ",
            "initial_condition": "",
            "step_description": "Make a MO call from SIM2 "},
        {   "comments": "",
            "expected_results": "VoWifi is started using SIM1 ",
            "initial_condition": "",
            "step_description": "Tap on Change SIM "},
        {   "comments": "",
            "expected_results": "VoWifi is finished ",
            "initial_condition": "",
            "step_description": "End call "},
        {   "comments": "",
            "expected_results": "DUT can camp on 4g/LTE and SIM2 can register IMS
                ",
            "initial_condition": "",
            "step_description": "Change Mobile Data to SIM2 "},
        {   "comments": "",
            "expected_results": " Voice Call (not VoWifi) is started ",
            "initial_condition": "",
            "step_description": "Make a MO call from SIM2 "},
        {   "comments": "",
            "expected_results": "Voice Call can be finished ",
            "initial_condition": "",
            "step_description": "End call "},
        {   "comments": "",
            "expected_results": "Volte Icon on status bar changes to Wifi Call
                Icon ",
            "initial_condition": "",
            "step_description": "Enable Wifi Calling for SIM2 "},
        {   "comments": "",
            "expected_results": "VoWifi is started using SIM2 ",
            "initial_condition": "",
            "step_description": "Make a MO call from SIM2 "},
        {   "comments": "",
            "expected_results": "Voice Call is made via SIM1 (not VoWifi) ",
            "initial_condition": "",
            "step_description": "Tap on Change SIM "},
```

```
        {   "comments": "",
            "expected_results": "Call is finished ",
            "initial_condition": "",
            "step_description": "End call "
        }]
}
```

## CASOS VERDADEIRO NEGATIVOS (VN)

```
{   "key": "TC-013",
    "component": "Messaging",
    "summary": "TC - Display - Peek and open notifications",
    "labels": ["aod", "daily_sanity", "display", "non_network_dependent", "
       platform_sanity", "refactor", "sanity", "sanity_common"],
    "steps": [
        {   "comments": "",
            "expected_results": "- Opening a group of notifications opens the app
                 as expected\n- Opening a conversation, from a group of
                notifications, opens the app directly in the message\n- Check the
                 correct behavior for a specific app ",
            "initial_condition": "There are app notifications in Display ",
            "step_description": "Open the applications using Display
                notifications\n(tap a notification and swipe up over the
                notifications to open the app) "},
        {   "comments": "",
            "expected_results": "- Received message/email can be replied as
                expected\n- With the feature OFF, the message can be replied on
                Display ",
            "initial_condition": "- [ Quick reply ON | Safe reply OFF ] setting
                in display\n- There are message notifications (e.g. email, SMS) "
                ,
            "step_description": "Reply to the message "},
        {   "comments": "",
            "expected_results": "- Received message/email can be replied as
                expected\n- With the feature OFF, unlocking the screen will be
                required ",
            "initial_condition": "- [ Quick reply OFF | Safe reply ON ] setting
                in display\n- There are message notifications (e.g. email, SMS) "
                ,
            "step_description": "Reply to the message "
        }]
}
```

```
{   "key": "TC-014",
    "component": "Browser",
```

```
    "summary": "TC - Ensure that browser is working when L2TP PSK VPN is
        connected via live cellular network.",
    "labels": ["ATT_Reg", "Android_O_updated", "CN_BR_Support", "CN_Basic_Request
        ", "CN_US_Support", "Cricket_Live", "Cricket_Reg", "Latam_Reg", "Plat_Reg
        ", "Retus_Reg", "Sprint_Reg", "TMO_Reg", "VZW_Reg", "
        execute_femtocell_att", "execute_femtocell_sprint", "
        execute_femtocell_tmo", "execute_femtocell_vzw", "execute_live_cricket",
        "femtocell_att", "femtocell_sprint", "femtocell_tmo", "femtocell_vzw", "
        network_dependent", "us-network-dependent"],
    "steps": [
        {   "comments": "To browse external sites it is necessary to add all
            these forwarding routes: 10.10.0.0/19 10.1.0.0/23 192.168.100.0/24 ",
            "expected_results": "Browser should work properly and site should get
                launched ",
            "initial_condition": "Device should be camped on live cellular
                network.\n\nNotes: see https://sites.google.com/a/enterprise-
                share/test-environment/vpn-environment?pli=1 for configuration.\n
                \n*In case of need of this settings*\n\nFor ICS build:\nb.
                Connection Name: XXX\nc. VPN server:144.188.54.174\nd. Pre Shared
                 Key Type: Text\ne. Pre Shaed key: test\ni. Username: user\nj:
                Password: user\n\nPress menu and save.\nSelect the L2tp PSK vpn
                just created to connect to it and enter below values:\nUser name:
                 test3\npassword: Password123 ",
            "step_description": "1.Create a L2TP PSK connection:\n- For N and
                older builds\nGo to Settings -> Wireless & network -> VPN
                settings -> Basic VPN -> Add VPN -> Add L2TP/IPSec PSK.\n- For O
                builds\nGo to Settings -> Network & Internet -> VPN -> Add VPN \n
                \n\nVPN set up For O builds:\na. VPN name: L2TP/IPSec PSK \nb.
                VPN type: L2TP/IPSec PSK \nc. Server Adress:144.188.130.240\nd.
                Enable L2TP secret: not chcked.\ne. IPSec pre-shared key: test\nf
                . Forwarding routes: 10.10.0.0/19 10.1.0.0/23 192.168.100.0/24\ng
                . User name: test3\nh. password: Password123 \nSave the VPN\n\n2.
                Connection to VPN\nand Browse any site like -www.espn.com (ICS
                build :Ater connected , verify by access http://10.10.12.3 (only
                accessable by this server) "
        }]
}
```

```
{   "key": "TC-015",
    "component": "GMS",
    "summary": "Google Duo: Multi Users: Receive duo call",
    "labels": ["GMS_Reg", "Plat_Reg", "binary_leverage", "low_pri_auto", "
        non_network_dependent", "non_vzw"],
    "steps": [
        {   "comments": "",
            "expected_results": "Device should not ring and notification appears
                when switch to the primary user ",
```

```
            "initial_condition": "Add multiple users ",
            "step_description": "Configure Google Duo on Secondary User "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "Turn on phone calls & SMS is disabled ",
            "step_description": "Switch to Primary user/Owner "},
        {   "comments": "",
            "expected_results": "Device should not ring ",
            "initial_condition": "",
            "step_description": "Receive a Duo call "},
        {   "comments": "",
            "expected_results": "Notification appears when switch to the
                Secondary user ",
            "initial_condition": "",
            "step_description": "Switch back to secondary user "
        }]
}
```

```
{   "key": "TC-016",
    "component": "Messaging",
    "summary": "TC - [RCS] Attach many audio, video, image files",
    "labels": ["Android_N", "Fi_Reg", "MPCS_Live", "MPCS_Reg", "Plat_Reg", "
        Retus_Reg", "Sprint_Reg", "TMO_Reg", "execute_femtocell_tmo", "
        execute_live_mpcs", "execute_live_sprint", "femtocell_tmo", "low_pri_auto
        ", "network_dependent", "rcs_aosp", "sprint_live"],
    "steps": [
        {   "comments": "",
            "expected_results": "No abnormal behavior observed. Messages are sent
                and received successfully ",
            "initial_condition": "DUT and Support device are RCS Capable ",
            "step_description": "Send and receive many audio/video/image files
                back to back "},
        {   "comments": "",
            "expected_results": "No abnormal behavior observed. Messages are sent
                and received successfully ",
            "initial_condition": "SIM Status is IMS Register ",
            "step_description": "Receive a Skype/Duo call during the sending and
                receiving of files "},
        {   "comments": "",
            "expected_results": "",
            "initial_condition": "DUT and recipient device are using cellular
                network (3G/HSPA/LTE) or Wi-Fi\nChat features Status \"Connected
                \" on Messages settings\n\"Messages\" and \"Carrier Services\"
                apps updated ",
            "step_description": "Repeat the steps in lanscape mode "},
        {   "comments": "",
            "expected_results": "",
```

```
            "initial_condition": "Multiple files are available for test ",
            "step_description": ""
        }]
}
```