Diogo Filipe Faria Duarte Fernandes

Nº46156

Uma Abordagem para Manutenção Preditiva baseada em Sistemas Multiagente e Machine Learning

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Doutor, FCT/UNL

Co-orientador: Ricardo Alexandre Fernandes da Silva Peres, Doutor, FCT/UNL

Júri:

Presidente: Doutor Bruno João Nogueira Guerreiro, Professor Auxiliar da FCT/UNL

Arguente: Doutor André Dionísio Bettencourt da Silva Rocha, Professor Auxiliar convidado da FCT/UNL

Vogal: Doutor Ricardo Alexandre Fernandes da Silva Peres, Professor Auxiliar Convidado da FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Junho, 2020

Uma Abordagem para Manutenção Preditiva baseada em Sistemas Multiagente e Machine Learning

Dedicatória

Para o meu avô Luís e restante Família

Agradecimentos

Nesta secção gostaria de deixar umas palavras de especial apreço a todos aqueles que contribuíram para a realização desta dissertação.

Em primeiro lugar, ao meu orientador, Professor Doutor José Barata, pela oportunidade de me facultar um tema na minha área de interesse, tornando assim a realização da dissertação mais interessante.

Ao meu co-orientador, Doutor Ricardo Peres, gostaria de agradecer por todo o apoio e dedicação durante este processo. Por ter estado sempre disponível quando eu precisei e pelos conselhos fundamentais para a realização desta dissertação.

Gostaria de agradecer aos meus amigos Diogo Oliveira, Paulo Dias, por todas as horas que passámos a realizar as nossas dissertações e os momentos que tivemos, sem eles tornava-se tudo mais difícil.

A todos os que me acompanharam durante os 4 anos na equipa de futebol da faculdade, Nuno Barreira, Rafael Coelho, Guilherme Rosa, José Soares, Manuel Barbas um muito obrigado pelos momentos que passámos, contribuíram para tornar o meu percurso académico melhor.

Um agradecimento especial à minha namorada, Bárbara Galrinho por todo o apoio e paciência nesta fase mais importante da minha vida de estudante. Obrigado por teres estado sempre ao meu lado, ainda mais na reta final deste trabalho. Sem ti ao meu lado isto era bastante mais difícil.

Por último, gostaria de agradecer a toda a minha família. Obrigado por todo o apoio incondicional, por todas os ensinamentos que me deram, por tudo o que me proporcionaram, sem vocês não estaria aqui.

A todos os meus amigos que não mencionei aqui, muito obrigado pelo privilégio de vos ter como amigos.

A todos vós,

O meu sincero Obrigado!

Resumo

Nos dias de hoje onde a tecnologia está em constante evolução, surge a necessidade de os sistemas industriais se adaptarem às alterações do mercado, de modo a fazer frente às necessidades de um mercado cada vez mais competitivo e global.

No seguimento da evolução da tecnologia e dos desenvolvimentos recentes que resultaram numa maior disponibilidade dos sistemas de aquisição de dados e redes de computadores, a natureza competitiva da indústria atual força os sistemas de manufatura a avançar para a implementação de metodologias de alta tecnologia, possibilitando a estes sistemas uma maior flexibilidade e robustez ao lidar com situações inesperadas.

A Manutenção Preditiva é a primeira grande quebra de paradigma na manutenção, intensificando-se cada vez mais, quanto maior o desenvolvimento tecnológico. Esta solução, é considerada como a mais fiável sob o ponto de vista de produção, na medida que monitoriza o equipamento e/ou sistema, agindo quando necessário.

No entanto, toda a tecnologia inerente a este tipo de manutenção ainda precisa de intervenção humana, pois, embora os equipamentos/sistemas já possam detetar as falhas, estes não se auto-reparam, necessitando da intervenção do homem para efetuar o reparo.

O trabalho desenvolvido na presente dissertação de mestrado consiste no estudo, de como integrar *Machine Learning* em sistemas ciber-físicos baseados em agentes e utilizando serviços.

Palavras-chave: Manutenção Preditiva, Sistemas Multiagente, Sistemas Ciber-Físicos, Indústria 4.0, Análise de Dados

Abstract

In present days, where we find technology in constant evolution, the need for Industrial systems to adapt to the market changes has aroused, in order to meet the needs of a challenging and more and more competitive and global market.

Following the evolution of technology and the recent developments that turned out into a major availability of data acquisition systems and network computers, the competitive nature of the present Industry is compelling the manufacturing systems to implement high technology procedures, by making way for a bigger flexibility and robustness when dealing with unexpected situations.

Predictive maintenance is the great break in the maintenance paradigm, getting more and more intense, as technological development is increasing. This solution can be considered as the most liable as far as production is concerned, seeing that it motorizes either the equipment and/or the system, acting when necessary.

However, all technology concerning this type of maintenance still needs human intervention, because although equipments/systems are able to detect failures, they cannot repair themselves, they still need human supervising for achieving it.

The work developed in the present dissertation of Master Thesis consists of studying how to integrate *Machine Learning* into cyber-physical systems based on agents and using services.

Keywords: Predictive Maintenance, Multi-Agent Systems, Cyber-Physical Systems, Industry 4.0, Data Analytics

Índice

Índice de figuras

Índice de tabelas

Acrónimos

CbM	*Condition-based Predictive Maintenance*
CC	Componente Computacional
CC	*Cloud Computing*
CF	Componente Físico
CbS	*Cloud-based Services*
CPPS	*Cyber-Physical Production System*
CPS	*Cyber-Physical System*
DA	*Deployment Agent*
EC	*Edge Computing*
FIPA	*Foundation for Intelligent Physical Agents*
IoT	*Internet of Things*
IoS	*Internet of Services*
JADE	*JAVA Agent Development Framework*
KM	*Knowledge Management*
MA	Manutenção Agendada
MAE	*Mean Absolute Error*
MAS	*Multi-Agent System*
ML	*Machine Learning*
MP	Manutenção Preditiva
RA	*Resource Agent*

RF	*Random Forest*
RFID	*Radio-Frequency Identification*
RUL	*Remaining Useful Life*
SA	*Subsystem Agent*
SbM	*Statistical-based Predictive Maintenance*
SM	Sem Manutenção

1

1 Introdução

1.1 Contextualização do Problema

Nas últimas décadas verificaram-se grandes mudanças na produção industrial, passando dos sistemas tradicionais de produção em massa para sistemas mais flexíveis e dinâmicos.

Nos dias de hoje, graças a uma maior globalização dos mercados e na procura de corresponder às exigências dos consumidores, as empresas investem recorrentemente em formas de melhorar e otimizar as suas atividades. Numa sociedade cada vez mais exigente e competitiva, métodos para melhoria da qualidade do produto e para redução dos custos, são fatores cada vez mais importantes.

Temos como exemplo a indústria automóvel, onde estudos realizados mostram que são gastos em média, mais de 22000$ em despesas associadas a tempo de inatividade *ATS* (*Advanced Technology Services*, 2006). Como consequência, é vital manter as máquinas de produção a funcionar normalmente. Por todos estes motivos, novas normas foram surgindo de forma a tornarem os sistemas de manufatura adequados à atualidade, sendo a agilidade, a flexibilidade e a modularidade, as grandes particularidades em que se basearam estes paradigmas.

A aplicação do ramo de informática na indústria, tem trazido grandes vantagens, o que, para além de possibilitar a modernização em processos de gestão e comunicação, permitem também implementar sistemas mais complexos, surgindo novos métodos com o intuito de prolongar o tempo de vida do equipamento, reduzindo assim os custos associados à sua manutenção.

Com o aumento de sistemas conectados, é oferecido também novas oportunidades e desafios e, dado que os mesmos se encontram conectados, possibilitam a transmissão de milhares de dados em tempo real para a nuvem, necessitando de serem analisados posteriormente. Desta maneira, serão fornecidos *insights* valiosos para as novas tomadas de decisão.

De modo a combater o tempo de inatividade muitas empresas estão a optar pela análise de dados preditivos. Para além de reduzir os custos, pode oferecer informações sobre padrões de desgaste, reduzindo assim os reparos. Esta análise, possibilita que o tempo de inatividade, caso a manutenção se realize antes do equipamento perder desempenho, passe a ser uma solução do passado.

1.2 Perguntas de Investigação

Tendo em conta o problema anteriormente referido e os benefícios para os clientes em usar métodos para analisar os dados, algumas questões podem ser colocadas:

- De que formas se pode conferir a capacidade de análise preditiva de dados de um sistema de Manufatura, garantindo um impacto reduzido na sua empresa?

No sentido de responder a esta pergunta pertinente, elaborou-se uma proposta que se apresenta e se descreve nesta tese. A base da proposta consiste no estudo de modelos de *ML* no *Edge* e na *Cloud* e perceber que vantagens estas abordagens trazem quando colocadas em produção.

1.3 Motivação

Considerando os desafios apresentados na secção anterior, torna-se bastante explícito o papel da monitorização nos sistemas de manufatura. A presente dissertação foca-se no tópico de Manutenção Preditiva (MP) nesses sistemas, sendo o objetivo deste estudo, avaliar de que maneira é possível trazer vantagens aos fornecedores.

Deste modo, é necessário perceber como os desenvolvimentos da tecnologia contribuem para estes sistemas. Num momento em que a indústria 4.0 é uma realidade atual, todas as tecnologias que esta engloba, alicerçada em IoT, permitem uma mudança do paradigma atual de centralização da produção para uma maior descentralização da produção, graças à evolução das tecnologias. A aplicação de inteligência permite um sistema de redes, em que os objetos comunicam entre si, conferindo independência e autonomia ao processo de produção.

Num sistema em que todas as máquinas estão conectadas entre si, são geradas informações valiosíssimas para a evolução e desenvolvimento dos produtos. De seguida, por meio de soluções sofisticadas como ML e *Data Mining*, estes dados podem ser analisados com o uso de inteligência, tornando possível identificar eventos com mais agilidade e eficácia.

Em sistemas muito complexos, com muitos componentes interagindo entre si, e influenciando o tempo de vida um do outro, quando é que a manutenção deve ser realizada para que os componentes não sejam substituídos prematuramente, permitindo assim o funcionamento do sistema de maneira confiável?

MP é a proposta à pergunta acima citada, onde se constroem modelos que quantifiquem o risco de falha de um componente/equipamento em qualquer momento, usando as informações geradas pelas máquinas, para prever a altura exata de se realizar a manutenção. Por esse motivo, o objetivo principal deste estudo, seria organizar uma estrutura que fosse capaz de extrair e analisar dados, de maneira a prever possíveis bons resultados futuros.

1.4 Principais Contribuições

As principais contribuições propostas neste trabalho relacionam-se, na medida que integram tecnologias já existentes. Na solução proposta, foi possível cruzar conhecimentos e criar uma arquitetura que possibilite alcançar a flexibilidade e rapidez, indispensáveis hoje em dia no processo de manufatura.

A arquitetura desenvolvida tem como base o comportamento dos atuais sistemas de produção. De salientar, que o sistema multiagente, foi desenvolvido para tentar ser o mais genérico possível, de modo a certificar-se, que a substituição deste com outra arquitetura seja simples.

1.5 Organização do Documento

Este documento é composto por 7 capítulos, sendo eles a introdução, o estado da arte, a arquitetura, a implementação, testes e validações, conclusão e trabalho futuro e por último as referências.

O primeiro e presente capítulo, **Introdução,** pretende contextualizar o trabalho. Numa primeira parte é exposto uma breve descrição do problema e a pesquisa associada ao mesmo.

O segundo capítulo, **Estado da Arte**, é onde será realizada a abordagem das bases teóricas para o desenvolvimento do projeto. É neste capítulo que se encontra grande parte do trabalho referenciado por este documento, onde é possível encontrar conceitos como Indústria 4.0, Sistemas de Produção Ciber-físicos, Sistemas Multiagente e Manutenção Preditiva.

No terceiro capítulo, a **Arquitetura**, é apresentada em detalhe a arquitetura da solução, bem como o seu comportamento. É descrito aqui uma solução para o problema apresentado na Introdução.

No quarto capítulo, **Implementação**, é apresentado as entidades que compõem o sistema, assim como as interações entre elas, as tecnologias e os dispositivos utilizados. Resumidamente é a implementação da arquitetura redigida no capítulo anterior.

No quinto capítulo, **Validação e Testes**, é onde são apresentados e discutidos os resultados obtidos. Desde o comportamento da solução num sistema *Cloud computing* e num sistema *Edge computing*, sendo posteriormente comparados os dois cenários.

No capítulo seis, **Conclusão e Trabalho Futuro**, é onde são enumeradas as conclusões obtidas através da realização do projeto. Também é apresentado neste capítulo possíveis diretrizes para trabalhos futuros.

Por último, o capítulo das **Referências**, é onde se encontra as referências aos documentos que serviram de apoio para a realização deste projeto.

2 Estado da Arte

Neste segundo capítulo, o objetivo é apresentar uma revisão dos desenvolvimentos mais recentes relevantes para esta dissertação, de modo a enquadrar o desenvolvimento desta arquitetura na literatura.

Existem muitas abordagens no que diz respeito aos sistemas de manufatura. No século XIX, o trabalho era totalmente feito à mão, pelo que todos os produtos eram únicos, surgindo muitas vezes com defeitos, devido às variáveis em que os operários trabalhavam os produtos (Oliveira, 2003).

Com o passar do tempo, o mercado sofreu inúmeras mudanças, mudando da produção em massa, definida por Henry Ford no início do século XX, conhecida pela produção de produtos diversificados em larga escala e a custos relativamente baixos, para uma produção de alta qualidade (Ribeiro & Barata, 2011).

Com a globalização dos mercados, novas soluções foram exigidas. Assim sendo, foram necessárias alterações nos sistemas de produção industrial. Estes sistemas, têm agora requerimentos que permitem seguir estas mudanças, nomeadamente a redução do tempo de colocação dos produtos no mercado, uma maior flexibilidade na produção e a redução dos custos de produção (Schäfer, 2007).

Num mercado cada vez mais competitivo e de modo a corresponder às necessidades dos clientes, foram adotadas novas metodologias. Estas modificações nos sistemas de produção industrial trazem, cada vez mais, novos paradigmas, permitindo assim várias soluções para o mesmo problema, conduzindo assim a um melhor e mais rápido progresso.

2.1 Indústria 4.0

Antes de ser abordado o tema de Indústria 4.0, terá que ser explicado primeiro as suas origens, ou seja, as Revoluções anteriores. Pode se verificar na Figura 1, um esquema que fala das Revoluções Industriais, assim como o que marcou cada Revolução.

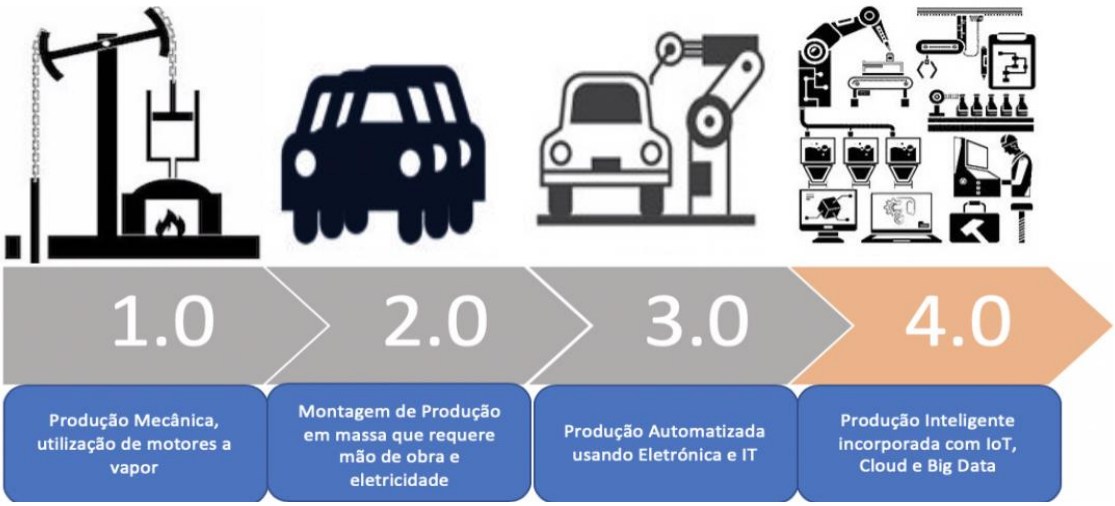

Figura 1 - História das Revoluções Industriais (adaptado de Pouspourika, 2019)

Tudo começou há mais de 200 anos em Inglaterra, com a transição de métodos de produção artesanais para a produção mecanizada. Esta mecanização estendeu-se desde o setor têxtil, à metalurgia, aos transportes e outros setores. A segunda Revolução é vista como um aperfeiçoamento e aprimoramento das tecnologias da primeira Revolução, mas acima de tudo é caracterizada pela separação entre conceção e execução, separando quem pensa (engenheiro) de quem executa (trabalhador). Na procura de lucros maiores e com a forma de produção em série de *Ford*, foi possível otimizar a produção, produzindo em massa, baixando assim o custo de cada unidade. A terceira Revolução Industrial destaca-se pelos descobrimentos nos ramos da Eletrónica e Informática, e consequentemente o seu uso no sistema de Produção Industrial, permitindo que muitas tarefas fossem realizadas de forma automática.

A Indústria 4.0, também conhecida como quarta Revolução Industrial, difere em relação às anteriores revoluções nas novas inovações tecnológicas usadas. Ao contrário das anteriores é esperada uma quarta revolução com um alcance mais amplio, não afetando só as indústrias, mas inclusive a Sociedade, razão pela qual é extremamente importante entender todo o seu potencial.

O termo Indústria 4.0 foi usado pela primeira vez na Feira de Hannover em 2011, atraindo logo interesse em todo o mundo. Segundo Kagermann, Wahlster & Helbig (2013) considera a Indústria 4.0 como a tendência atual em direção à Automação e Troca de Dados em Tecnologias de Manufactura. Esta Revolução está marcada pelo aparecimento de novas tecnologias, como a Robótica, a Inteligência Artificial, a Análise de Dados, *Internet of Things (IoT)*, *Internet of Services (IoS)*, *Cyber-Physical Systems* (CPS), entre outras.

Para além disso, as tecnologias relacionadas com esta indústria podem levar a produtos e serviços completamente novos. Os sensores e dispositivos portáteis, assim como a Análise de Dados e a Robótica, usados nesta revolução permitem melhorias em diversos meios, desde a criação de protótipos até à conectividade em produtos previamente desconectados.

Em Hermann, Pentek, & Otto (2015) é definido o conceito de Fábrica Inteligente, como uma fábrica constituída por *CPS* que através de *IoT* comunica com os humanos em tempo real a fim de realizar um determinado objetivo. Como todas as tecnologias, esta não será diferente e terá impactos. Como se pode verificar na Tabela 1 estes impactos serão visíveis em vários níveis.

Impactos	Consequências do Impacto
Ecossistema	Afetam os clientes, os fornecedores, entre outros. As tecnologias da Indústria 4.0 permitem interações entre cada ponto da rede
Organizações	Permitem que as organizações sejam mais proactivas, aumentando assim a produtividade
Individual (funcionários e clientes)	Para os funcionários representa uma mudança na forma de trabalho, para os clientes uma maior satisfação das suas necessidades

Tabela 1 - Impactos Indústria 4.0 (adaptado de Cotteleer & Sniderman, 2017)

Para além destes impactos, também surgirão impactos negativos, com os ataques cibernéticos na frente, pois quanto mais conectada a indústria está, mais esta fica sujeito à espionagem.

De uma forma geral, pode ser dita que a Indústria 4.0 facilita a visão e execução das Fábricas Inteligentes, conecta os recursos de rede junto com uma serie de outros recursos, permitindo que as organizações saibam de antemão assuntos que não conheciam antes. Estas Fábricas Inteligentes acima de tudo terão a capacidade e autonomia para prever falhas nos processos, e, por sua vez, adaptar-se às mudanças não desejadas.

Como refere Drath e Horch (2014) ao debruçar-se sobre a Indústria 4.0: *"Indústria 4.0 é um fenómeno, que virá inevitavelmente, quer queiramos ou não"* (p. 58).

2.2 Sistemas de Produção Ciber-Físicos

Os sistemas de produção modernos são sistemas bastante complexos, constituídos por inúmeros componentes individuais, cujo objetivo é trabalharem em conjunto, de modo a garantir o sucesso. Segundo Ribeiro (2017), o conceito de Sistemas de Produção Ciber-Físicos (*CPPS*) junta as funcionalidades e os benefícios de Sistemas Ciber-Físicos (*CPS*) aplicados na Indústria.

CPS são sistemas que compreendem a integração de computação com processos físicos, por outras palavras transfere o mundo físico num mundo virtual (Hermann et al., 2015). Contudo ao integrarmos os componentes físicos estamos a diminuir a fiabilidade do sistema, devido ao comportamento imprevisível do mundo físico, tornando assim os *CPS*, em sistemas dinâmicos, que se adaptam às alterações do mundo físico.

No que diz respeito à evolução dos sistemas ciber-físicos, estes podem ser implementados na indústria de várias formas. Uma de muitas formas, é constituída por 3 fases. Na primeira, são incluídas *tags Radio-Frequency IDentification* (*RFID*), permitindo assim identificar os objetos. Numa segunda fase, estes *CPS* são equipados com sensores e atuadores, enquanto que na terceira fase os *CPS* são capazes de analisar e armazenar dados, para além de serem compatíveis com a rede (Hermann et al., 2015).

Os *CPS* podem ser desenvolvidos para gerir *Big Data* e contribuir para a conectividade das máquinas, levando ao objetivo inicial de obter máquinas inteligentes e autónomas. Em Lee,

Bagheri, e Kao (2015) é apresentada uma comparação entre as fábricas de hoje em dia e a última geração de indústria, a Indústria 4.0, implementada com *CPPS*.

Devido ao aumento dos sistemas de aquisição de dados, consequentemente existe uma maior quantidade de dados disponíveis, necessitando assim de algoritmos que sejam capazes de os processar. *Machine Learning* (*ML*) é um conjunto de técnicas para construir modelos matemáticos que podem fazer inferências a partir de amostras de dados (conjunto de treino) (Ribeiro, Grolinger, & Capretz, 2015).

Nas ferramentas necessárias para a análise de dados, os algoritmos de *ML* podem ser usados para melhorar o desempenho e prever eventos inesperados, atuando nestes, tornando-se um aliado importante em tarefas de Manutenção.

2.3 Sistemas Multiagente

O paradigma de Sistemas Multiagente (*MAS's*) referenciados na literatura como *Multi-Agent System*, fornece uma nova abordagem para lidar com a complexidade dos sistemas de manufatura (Sabar, Montreuil, & Frayret, 2009). Não há uma definição consensual de agente inteligente, variando de investigador para investigador.

Segundo Oliveira,Fischer,e Stepankova (1999) *MAS* é definido como uma coleção de agentes autónomos que comunicam entre si, coordenando as suas atividades para alcançar um objetivo, ao mesmo tempo que cada agente procura objetivos individuais. Pode-se então concluir, que agente é o elemento principal destes *MAS's*.

Estes agentes têm diversas características, distinguindo-se assim de supervisores e/ou controladores que estejam dentro do mesmo ambiente, características estas que se podem observar na Figura 3.

De seguida, serão explicadas em pormenor as características da Figura 3:

- **Autonomia:** Os agentes têm as propriedades necessárias para desenvolver as suas responsabilidades de forma independente de terceiros.
- **Sociabilidade:** Os agentes têm a habilidade de interagir com outros agentes e entidades para cumprir um determinado objetivo.
- **Habilidade-Inferencial:** Os agentes têm a competência de operar nas provisões dos objetivos inteligentes, como a coleta de observações.
- **Reatividade:** Os agentes são capazes de reagir a mudanças no ambiente.
- **Proactividade:** Os agentes têm um comportamento orientado a objetivos, isto é, não são entidades puramente reativas, podendo tomar iniciativa durante uma operação.
- **Adaptabilidade:** Os agentes têm a capacidade de se adaptarem a alterações no ambiente.

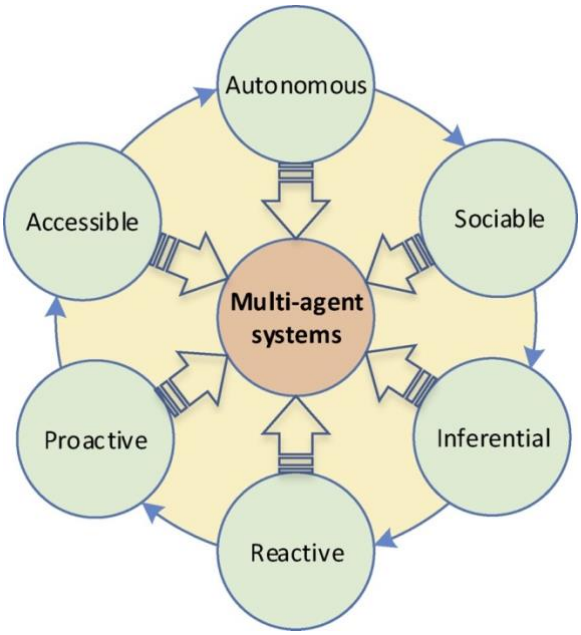

Figura 2 - Principais características dos agentes inteligentes (adaptado de Khan et al., 2019)

Para resolver problemas complexos de maneira distribuída, estes agentes interagem através de diferentes mecanismos, tais como: negociação, cooperação, coordenação ou passagem de mensagem simples (Wooldridge, 2002).

As soluções suportadas por *MAS´*s têm sido utilizadas maioritariamente na resolução de problemas de escalonamento na produção. Em Sabar et al. (2009) é desenvolvida uma solução recorrendo aos *MAS´*s para o problema de escalonamento de pessoal, no contexto de um centro de montagem de múltiplos produtos. Esta solução, baseia-se na cooperação de vários agentes que contêm as competências individuais e preferenciais dos trabalhadores. Com recurso a algoritmos matemáticos, cada agente é capaz de conhecer as atividades menos satisfatórias de cada trabalhador, evidenciando as mais satisfatórias, de modo a melhorar os seus planos de trabalhos iniciais.

Com a evolução da Indústria, surgiram também as linhas automatizadas e rapidamente foram desenvolvidos estudos para melhorar o rendimento das máquinas que se encontram nessas linhas. Um dos grandes exemplos da tecnologia envolta nesta evolução da Indústria, é a crescente utilização da informática.

Existem inúmeras soluções desta manufatura inteligente suportados por sistemas multiagente como (Guo & Zhang, 2010). Nesta solução é proposto um método eficaz para a manufatura inteligente, com base numa plataforma multiagente para resolver conflitos e responder a eventos imprevistos. Com soluções como esta, os sistemas vão ficando cada vez mais simples e a sua confiabilidade e robustez serão melhoradas, tornando estes sistemas cada vez mais flexíveis.

Em Peres, Rocha, Leitão & Barata (2018) e Antzoulatos et al. (2016) são apresentadas duas implementações de *MAS´*s, ambas desenvolvidas em *JADE*. Na primeira, é apresentada uma

framework, não só responsável pela aquisição de dados, mas também pela análise e avaliação destes, baseada em tempo real e históricos de um sistema de manufatura. Foi possível verificar-se que ao adotar o *MAS*, este confere flexibilidade e robustez adicionais ao *CPPS*, permitindo-o adaptar-se rapidamente às alterações. Por outro lado, na segunda proposta é usada uma framework para reconfigurar sistemas de montagem que mudam frequentemente. Esta framework e os recursos de manufatura são geridos por um *MAS*, onde os agentes comunicam uns com os outros através de comunicação (*FIPA*) referenciada na literatura como *Foundation for Intelligent Physical Agents* a fim de agregar as capacidades dos recursos para combiná-los com as especificações dos produtos.

2.4 Manutenção Preditiva

Manutenção Preditiva (MP) pode ser caracterizada por um conjunto de atividades que deteta alterações no equipamento, quer sejam falhas ou avarias, com o intuito de executar os corretos trabalhos de manutenção de forma a prolongar o tempo de vida do equipamento, uma vez que estas falhas ou avarias podem levar à interrupção da cadeia de produção (Lee, Cao, & Ng, 2017).

MP como o nome implica, é a manutenção de componentes, quer sejam pequenos ou grandes, de acordo com as previsões de quando vão falhar ou necessitar de manutenção. Existem vários fatores que influenciam estas previsões:

- ***Estado em tempo real do equipamento****:* Como é que está a funcionar/trabalhar?
- ***Histórico do equipamento:*** Como é que funcionou no passado?
- ***Dados para equipamentos semelhantes:*** Saber como outras partes foram executadas.
- ***Registos de Manutenção:*** Quando se encontrou em serviços de manutenção?
- ***Horários de Manutenção:*** O que recomenda o fabricante?

Este tipo de manutenção pode ser classificado em dois tipos dependendo dos métodos de detetar as falhas.

SbM referenciado na literatura como *Statistical-based predictive Maintenance* depende dos dados estatísticos tirados a partir de registos das paragens dos itens e dos componentes da fábrica, de forma a desenvolver modelos para prever falhas (Wang, 2016).

Por outro lado, Bousdekis, Magoutas, Apostolou e Mentzas (2015), argumenta que *CbM*, referenciado na literatura como *Condition-based Predictive Maintenance* incluí 2 fases. A fase de diagnóstico consiste em monitorizar o equipamento continuamente ou periodicamente para detetar falhas no sistema, enquanto que a fase de prognóstico precisa do cálculo ou estimativa de vida útil remanescente das máquinas e equipamento.

Os sistemas tradicionais como a manutenção reativa ou corretiva seguem a abordagem *run-to-failure*, abordagem na qual os trabalhos de substituição ou de reparo surgem após ocorrer a interrupção do equipamento. Este método tradicional, que tem sido aplicado na Indústria há décadas é considerado como a melhor política de manutenção para componentes não críticos com tempo de reparo curto no sistema. Contudo na maioria dos casos, as falhas no equipamento podem levar a atrasos inesperados na produção e à diminuição da mesma (Lee et al., 2017).

O grande problema destes métodos é o seu alto custo, pois esperavam que o material falhasse até se realizarem os respetivos trabalhos de manutenção, perdendo-se assim a produção.

Para adicionar a este problema, no seguimento das inspeções pessoais poder-se-iam substituir partes que não seriam necessárias, tornando-se essas inspeções mais caras. Por outro lado, seguindo os conselhos do fabricante não se pagavam os custos das inspeções, as quais podiam resultar na substituição de peças que ainda estivessem a funcionar e o problema podia continuar.

Uma solução para diminuir o custo e aumentar a produção é gerir continuamente as atividades associadas à manutenção e controlar a degradação do equipamento (MP), como se pode verificar na Figura 3.

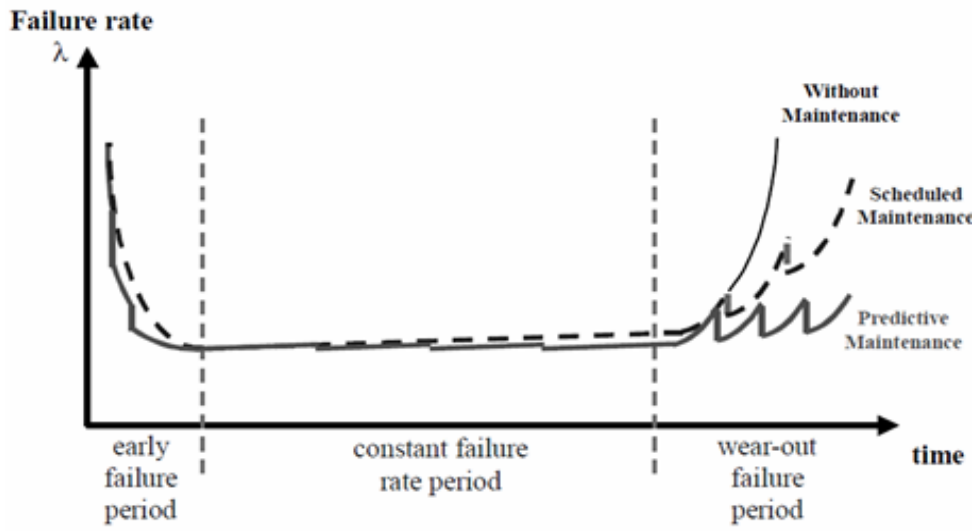

Figura 3 - Comparação da taxa de falhas em diferentes tipos de Manutenção (adaptado de Lee et al., 2017)

Como se pode observar na Figura 3, são contempladas três estratégias de manutenção: Sem Manutenção (SM), Manutenção Agendada (MA) e MP. São descritos também três períodos: período de falha precoce, período de taxa de falha constante e período de falha de desgaste.

Pode verificar-se através da figura que, com o passar do tempo ao longo dos distintos períodos, MP é a melhor solução. Durante o período de falha de desgaste, ou seja, o período mais dramático para os componentes/equipamento, **MP** é a que tem menor taxa de falha durante o mesmo espaço de tempo.

Como era esperado, a solução MA tem melhor rendimento (menor percentagem de erro no mesmo espaço de tempo) em relação à solução SM. Numa era marcada pela *Big Data*, a eficiência geral dos sistemas de manufatura pode ser melhorada pela implementação de MP (Lee et al., 2017). Devido ao que foi descrito anteriormente, a solução baseada em **MP** foi a escolhida para ser estudada, em detrimento das outras duas.

2.5 Análise Preditiva na Manufatura

Nos dias de hoje, em indústrias que já são bastante eficientes, os fabricantes enfrentam o desafio de melhorar a produtividade (Manyika et al., 2011). Estes procuram aumentar a produtividade, melhorando a eficiência no design e produção. Um método que promete proporcionar ganhos significativos na produção é a aplicação da análise de dados. No entanto, este uso ainda está longe de tirar o máximo proveito dos dados dos sistemas de manufatura.

A Análise Preditiva usa técnicas, como *ML*, *Data Mining*, entre outras, de modo a descobrir factos e fazer previsões sobre eventos futuros. Algumas destas aplicações de análise preditiva incluem deteção e previsão de falhas, análises para prever garantia e manutenção de produtos e muito mais.

Em Lechevalier, Narayanan e Rachuri (2014) são apresentados vários exemplos de aplicações de análise preditiva nos sistemas de manufatura. Exemplos desde aplicações no controlo de sistema de manufatura, aplicações no controlo de qualidade de manufatura, aplicações no diagnóstico de falhas de equipamento, entre outras. Em Shin, Woo e Rachuri (2014) é proposta uma arquitetura que visa responder às dificuldades para prever o desempenho nos sistemas de manufatura, nomeadamente no tempo que demora a recolher dados e construir modelos a partir destes dados.

2.5.1 Machine Learning

Nos dias de hoje, a Inteligência Artificial e *ML* tornaram-se peças importantes nos sistemas de produção, ajudando a reduzir o custo e tempo de produção. Devido ao crescimento de *ML*, os dados podem ser aproveitados de maneira eficaz resultando em melhorias significativas (MacGregor & Cinar, 2012).

Em Shang e You (2019) é apresentado um estudo de análise de dados e *ML* para o controlo ideal de processos industriais. Este diz que o controlo preditivo do modelo é uma abordagem bem estabelecida, baseado num modelo matemático conhecido para descrever comportamentos do sistema e planear sequências de controlo num futuro próximo. Também é descrito, que apesar do controlo preditivo do modelo poder ser ideal na prática, geralmente também existem incompatibilidade de modelos, ruídos aleatórios, entre outros. Nestes casos, uma abordagem promissora é integrar modelos com análise de dados e *ML*, que mostram grande potencial para lidar com o desconhecido (Yan & Wang, 2014).

Avanços recentes em Inteligência Artificial, especialmente em modelos não paramétricos de *ML*, ajudam a combater muitas das limitações mencionadas anteriormente e fornecem abordagens práticas para quebra de equipamentos. Entre eles, os modelos de *Random Survival Forest* são particularmente atraentes, pois podem lidar efetivamente com problemas de propensão e desvio de modelagem (Ishwaran, Kogalur, Blackstone & Lauer, 2008).

Em Bukkapatnam, Afrin, Dave e Kumara (2019) é apresentado um novo modelo, *Manufacturing System-Wide Balanced Random Forest* que combina os pontos fortes do *Random Survival Forest* e o balanceamento de dados, para aprender a evolução complexa de falhas e padrões no equipamento/máquina usando os dados da estação.

No estudo de (Radetzky, Rosebrock & Bracke, 2019) o objetivo é fornecer uma abordagem para a adaptação dos processos de manufatura que influenciam o resultado da superfície com base em medições e parâmetros estatísticos. Neste estudo, os dados foram analisados recorrendo a algoritmos de *ML*, divididos em 2 categorias, algoritmos de aprendizagem supervisionada e de não supervisionada.

Em Cavalcante, Frazzon, Forcellini e Ivanov (2019) é apresentado um estudo de manufatura digital, que adota uma abordagem híbrida (combinação de Regressão Logística com *K-Nearest Neighbors*) em combinação com modelos de simulação e integra-os no contexto de fornecedores.

2.5.2 Edge Computing

Edge Computing (EC) permite que os dados produzidos ou recolhidos por rede IoT sejam processados mais perto de onde são criados, em vez de terem de ser enviados através de longos segmentos de rede para centros de dados ou plataformas de *Cloud computing*.

Ao fazer esta computação mais próxima do extremo permite que os usuários beneficiem de serviços mais confiáveis, enquanto que as empresas aproveitam para analisar dados importantes em tempo real.

Um dos melhores exemplos de *EC* é o uso das realidades aumentada e virtual, que costumam ser prejudicadas por alta latência e largura de banda insuficiente. Outros benefícios da *EC* incluem a habilidade de fazer grandes análises e agregações de dados localmente, possibilitando a tomada de decisões quase imediata. Para além disso, a EC reduz o risco de exposição de dados confidenciais, porque armazena localmente os dados (Red Hat, 2020). Por todas as razões acima mencionadas, EC é uma tecnologia que caminhara paralelamente ao crescimento da IoT.

No estudo de Zhao, Lin, Shen, Zhang e Huang (2020) é apresentado o conceito de *EC* profundamente integrado na tecnologia IoT. Neste estudo uma arquitetura para *Edge computing* em *IoT* é desenvolvida para descarregar a pressão computacional e realizar a respetiva tomada de decisão.

Em Hu, Miao, Wu, Hassan & Humar (2019) é apresentado uma arquitetura do sistema *iRobot Factory*. É discutido neste estudo, a interação eficiente de informações, a fusão de dados e a produção automática. Também é apresentado duas soluções, uma baseada em sistemas de manufatura cognitiva que trata da operação e manutenção e outra baseada em *EC* que trata da comunicação e interação.

2.5.3 Cloud Computing

Cloud Computing (*CC*) pode ser definida como um modelo de disponibilização e utilização de tecnologias de informação e comunicação, que permite o acesso remoto, através da Internet, a um leque de recursos de computação partilhados em formas de serviços.

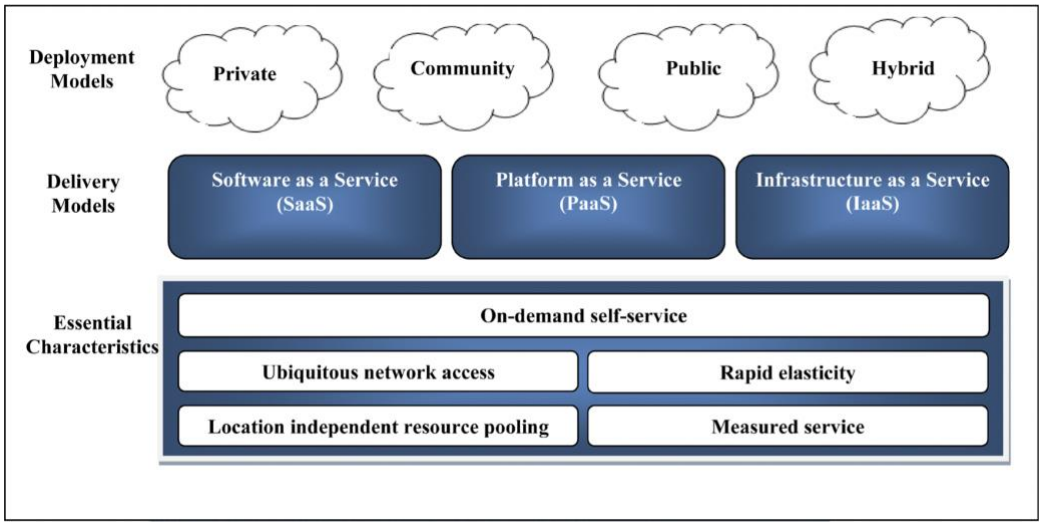

Figura 4 - Definição de *Cloud Computing* (adaptado de Mell & Grance, 2011)

Como se pode verificar na figura 4 a definição de *CC* é definida por quatro tipos de computação, três modelos de serviço e cinco características essenciais. Os quatros tipos de computação são *Cloud* Privada, *Cloud* Comunidade, *Cloud* Híbrida e *Cloud* Pública e são implementados dependendo do interesse do fornecedor em ter maior confiabilidade e segurança (Zhang, Cheng e Boutaba, 2010). Os modelos de serviço são oferecidos pela *Cloud* e podem ser agrupados em três categorias como se pode observar na imagem acima. Por último *CC* fornece cinco recursos importantes que são diferentes da computação do serviço tradicional.

2.6 Análise da Discussão da Literatura

Como foi possível verificar, os sistemas de manufatura evoluíram ao longo do tempo, partindo de sistemas centralizados, para sistemas distribuídos. De modo a colmatar as necessidades emergentes, surgiram vários paradigmas. Todos estes paradigmas cumprem com algumas características fundamentais, entre as quais a robustez e a adaptabilidade.

Para além do que foi descrito anteriormente, com esta evolução foi possível perceber uma grande mudança na forma de pensar, planear e agir na indústria. Hoje em dia, a maior parte dos sistemas apresenta tecnologias que permite maximizar a produção e reduzir os custos. Com estas tecnologias e com as que possam surgir, os mercados vão ficando cada vez mais exigentes, e a competitividade entre as empresas aumentará.

Como se pode observar neste capítulo, há cada vez mais tecnologia a emergir e é fundamental perceber a mesma e tirar o máximo proveito possível. É necessário encontrar uma forma de agregar as melhores tecnologias, de modo a obter os melhores resultados possíveis.

3

3 Metodologia e Arquitetura

Tal como referido no capítulo anterior, com a evolução das tecnologias surgem novos paradigmas. Com estas tecnologias surgirão necessidades, que serão colmatadas com soluções inovadoras. Atualmente existem no mercado inúmeras soluções que possibilitam a criação de sistemas, possibilitando a estes uma maior flexibilidade e robustez.

Posto isto, no presente capítulo é apresentada a arquitetura do sistema, as suas características e todos os constituintes que nele estão presentes.

3.1 Metodologia da Arquitetura

Neste subcapítulo é apresentada a metodologia adotada, abordando os passos gerais para a realização deste estudo, desde o desenho da arquitetura à implementação do sistema.

Numa primeira parte foi necessário fazer uma análise exploratória dos dados. Com isto, foi possível obter determinadas características dos dados e informações essenciais para a realização deste estudo. Como explicado anteriormente, foi a partir desta análise que se adquiriu um conjunto de informações indispensáveis, permitindo assim tomar decisões para criar as regras necessárias para o *runtime* dos agentes presentes no sistema.

Depois desta primeira análise exploratória dos dados, seguiu se a fase de seleção do modelo de *ML*. Para esta fase houve a necessidade de testar vários modelos, comparando-os entre si, de modo a selecionar o modelo adequado. Após esta seleção, começou-se a definir a arquitetura que será usada como suporte neste estudo. O primeiro passo foi construir o *MAS*, desde os seus agentes, até à interação destes com o resto da arquitetura. Esta arquitetura será apresentada no próximo subcapítulo.

De seguida, com os agentes já criados, procedeu-se à criação do método *Cloud* e do método *Edge*, incorporando os agentes criados anteriormente nestes. Estes são os dois métodos que foram escolhidos para servir de comparação neste estudo.

Por último, depois de já ter todos os constituintes e os métodos criados, procedeu-se à execução do sistema, e posterior comparação entre os dois métodos. Todos estes passos que foram descritos, serão abordados detalhadamente neste e no próximo capítulo.

3.2 Visão Geral da Arquitetura

De seguida serão apresentadas duas arquiteturas que fazem parte do sistema apresentado. Numa primeira instância será abordado uma representação de alto nível do sistema, seguindo-se a representação do sistema multiagente.

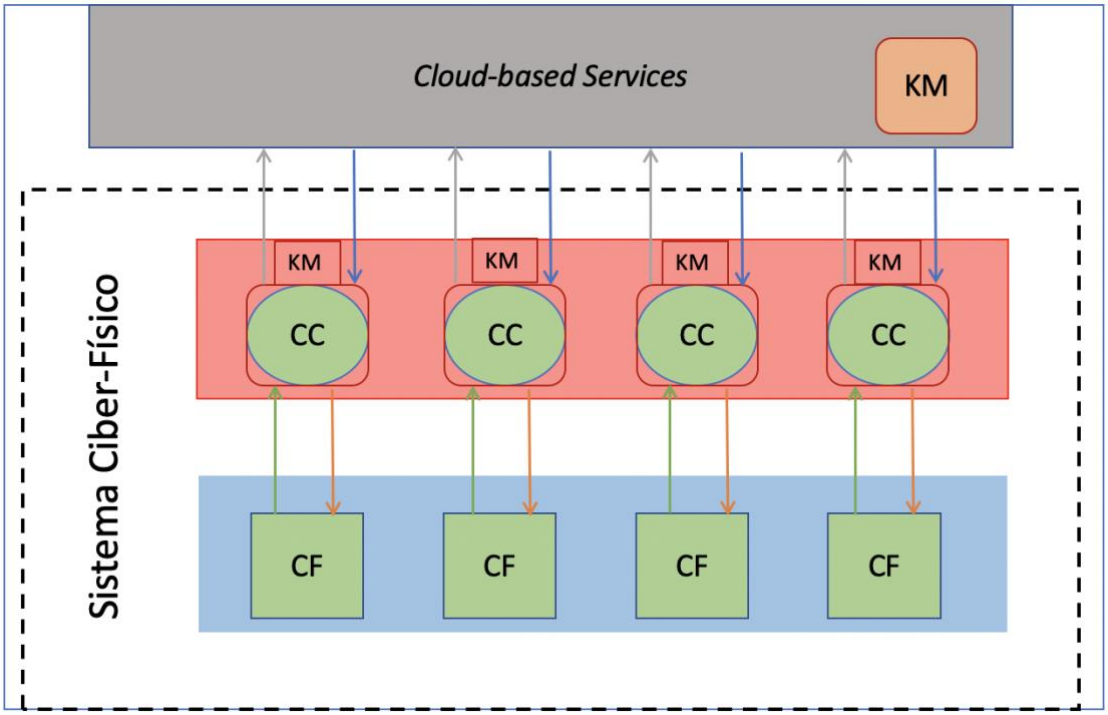

Figura 5 – Arquitetura de alto nível do sistema

Como se pode observar a partir da figura 5 é possível observar os elementos presentes na solução apresentada para este estudo, assim como as interações entre estes. Como é possível perceber esta arquitetura de alto nível está dividida em dois blocos. De um lado temos a *Cloud-based Services*, representado na literatura como *CbS* e no outro o *CPS*. Será explicado de seguida qual é a responsabilidade de cada um destes blocos e a interação um com o outro.

O bloco *CbS* é constituído pelo bloco *KM*, que irá ser alvo de estudo nas próximas seções e é o bloco responsável por responder aos pedidos provenientes do *CPS*. Um exemplo destes pedidos pode ser uma solicitação para que o modelo seja treinado novamente, enviando esse resultado de volta para o *CPS*.

O bloco *CPS* por outro lado é constituído por 2 componentes, o componente físico, representado na literatura como CF e o componente computacional, representado na literatura como CC. Por um lado, temos o CC, que é o responsável por receber os dados do CF e em certas instâncias enviar dados para o bloco *CbS*. Como se pode observar na figura acima este componente também tem o bloco *KM*. De realçar, que o CC também troca informação com o CF. Um exemplo desta troca é uma mensagem de alerta quando um resultado é inferior a um valor. Por outro lado, temos o CF que é o componente responsável por enviar os dados provenientes dos sistemas de produção para serem analisados. Para a implementação da parte ciber do CPS foi

desenhado um *MAS*. De seguida será apresentado o *MAS*, desde à sua arquitetura nesta subsecção assim como à sua implementação no próximo capítulo.

O modelo do sistema apresentado, como pode ser observado na figura 6, tem como base a arquitetura proposta em (Antzoulatos et al., 2016). Neste sistema, estão presentes três tipos de agentes: *Subsystem Agent* (*SA*), *Resource Agent* (*RA*) e *Deployment Agent* (*DA*), todos eles com funções diferentes uns dos outros (Barata, Camarinha-Matos, & Cândido, 2008).

É de extrema importância referir que o sistema suporta a presença de múltiplos agentes, sendo que na arquitetura apenas existe uma abstração de cada um deles. Isto é um pormenor do sistema, que permite a existência de paralelismo de ações, reunindo assim um dos principais objetivos deste trabalho, ter um sistema dinâmico e eficiente.

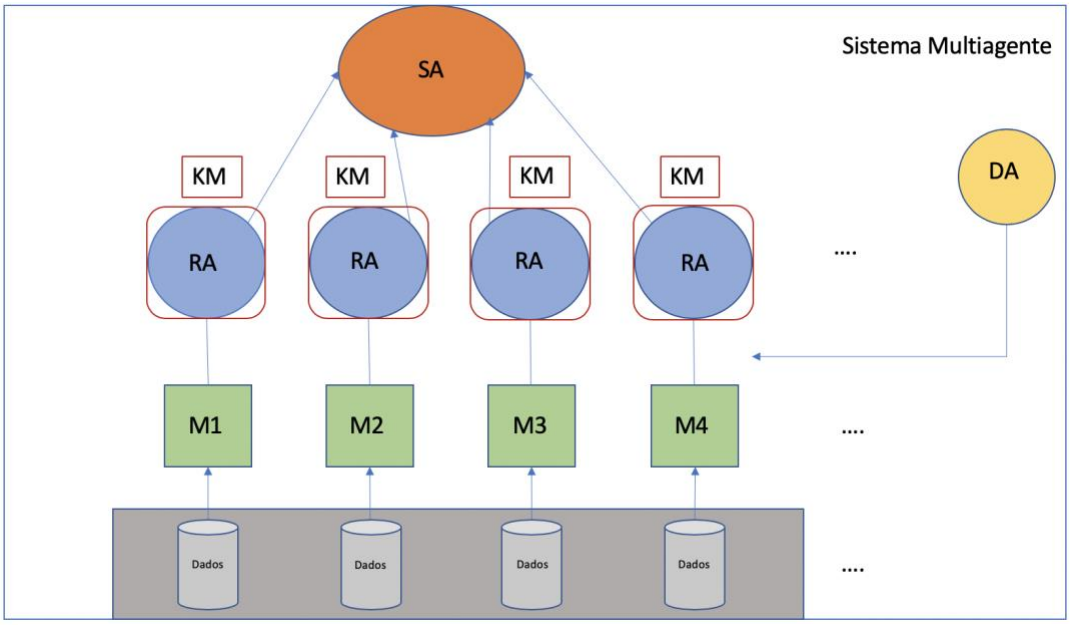

Figura 6 - Desenho geral do MAS (adaptado de Antzoulatos et al., 2016)

Este sistema multiagente pretende-se que seja capaz de implementar vários tipos de agentes, como dito anteriormente. Agentes estes, que disponibilizem vários comportamentos essenciais à comunicação e interoperação entre estes. Também é requerido que todos os agentes possam ser independentes entre si e que sejam autónomos, tentando satisfazer os pedidos que lhes chegam e as funções que os caracterizam da maneira mais rápida e possível.

Quando começa a execução do sistema, são lançados todos os agentes responsáveis por abstrair todas as entidades presentes no sistema. Cada um destes agentes tem um conjunto de comportamentos criados com recurso à plataforma *JADE* (*Java Agent Development Framework*), que serão abordados detalhadamente no próximo capítulo, assim como o bloco *Knowledge Management* (*KM*), responsável por tratar dos dados. É importante realçar que apesar destes agentes terem sido implementados usando a plataforma *JADE*, poderia ter sido usada outra tecnologia.

Segue-se de seguida a tabela 2, onde são apresentadas de um ponto de vista funcional o papel de cada agente, divididos em 3 grupos, agentes de suporte, agentes de execução e agentes de monitorização, de acordo com o seu papel no sistema.

Grupo	Classe	Função
Suporte	*Deployment Agent (DA)*	O *DA* possibilita ao utilizador lançar um número pretendido de Resource Agents automaticamente.
Execução	*Resource Agent (RA)*	O *RA* representa o agente mais importante do sistema. Este agente é o principal responsável por gerir o modelo que contém todas as informações pertencentes a todo o sistema, tendo, portanto, uma grande quantidade de informação a fluir através dele. Por outras palavras, é o que realiza todas os mecanismos necessários para recolher os resultados necessários neste trabalho.
Monitorização	*Subsystem Agent (SA)*	O *SA* é responsável por agregar informação dos *RA*, de modo a gerar novo conhecimento.

Tabela 2 - Agentes constituintes da arquitetura e funções

3.3 Arquitetura do Sistema

A arquitetura utiliza uma abordagem multiagente genérica, para que seja adaptável a diferentes tipos de sistemas de manufatura.

Para um funcionamento adequado desta arquitetura foram considerados 3 agentes já anteriormente referenciados, sendo estes o *Resource Agent* (*RA*), *o Subsystem Agent* (*SA*) e o *Deployment Agent* (*DA*).

Os diversos agentes podem ser representados por breves fluxogramas onde se pode observar o comportamento de cada um deles, bem como a ligação existente entre os agentes. Nas próximas subseções é apresentada uma visão dos principais agentes do sistema.

3.3.1 Deployment Agent

Como descrito na tabela 2, o *DA* é o agente responsável por lançar um determinado número de agentes no sistema. Esta operação ocorre no momento de início de execução do sistema, quando o *DA* é inicializado.

Quando o *DA* é inicializado, antes de serem lançados outros tipos de agentes no sistema, este agente fornece a habilidade, se desejado pelo fabricante, de automaticamente carregar informações. É possível visualizar este comportamento na figura 6.

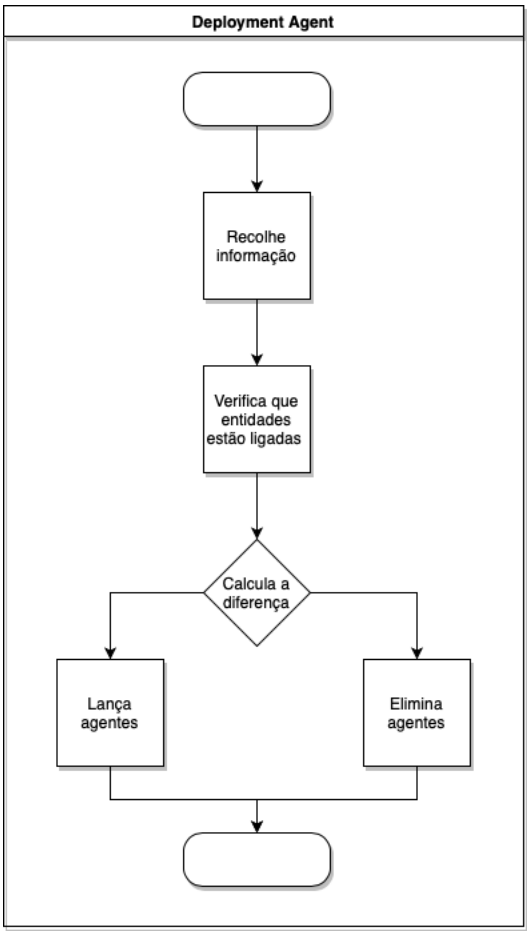

Figura 7 - Fluxograma do Deployment Agent

Como se pode observar na figura acima descrita, o *DA* é capaz de recolher informações provenientes de várias fontes, quer sejam de ficheiros, de *APIs* ou de *PLCs*. De seguida, tendo estas informações vai verificar que entidades estão em execução e calcula a diferença com as informações que recolheu das fontes. Após calcular a diferença, o *DA* lança novos agentes ou elimina consoante o valor da diferença. Este agente só irá ter efeito no início do programa, terminando depois de lançados os respetivos agentes.

3.3.2 Resource Agent

Como foi descrito na tabela anterior, este agente é o mais importante do sistema. É neste agente que está toda a inteligência do sistema. O número destes agentes que irá ser lançado, está no na fonte externa que o *DA* irá recolher. Como foi dito anteriormente, se o fabricante desejar, que o *DA* automaticamente descubra informações, o que foi descrito antes irá acontecer. Caso contrário o fabricante tem a opção de, no momento de início de execução do sistema lançar manualmente estes agentes. Este agente como se pode verificar na figura 7 irá realizar o método 1 (*Cloud*) ou o método 2 (*Edge*) consoante as características adquiridas por este. Este

comportamento de escolha do método irá ser abordado de seguida. De seguida, pode-se verificar o comportamento deste agente.

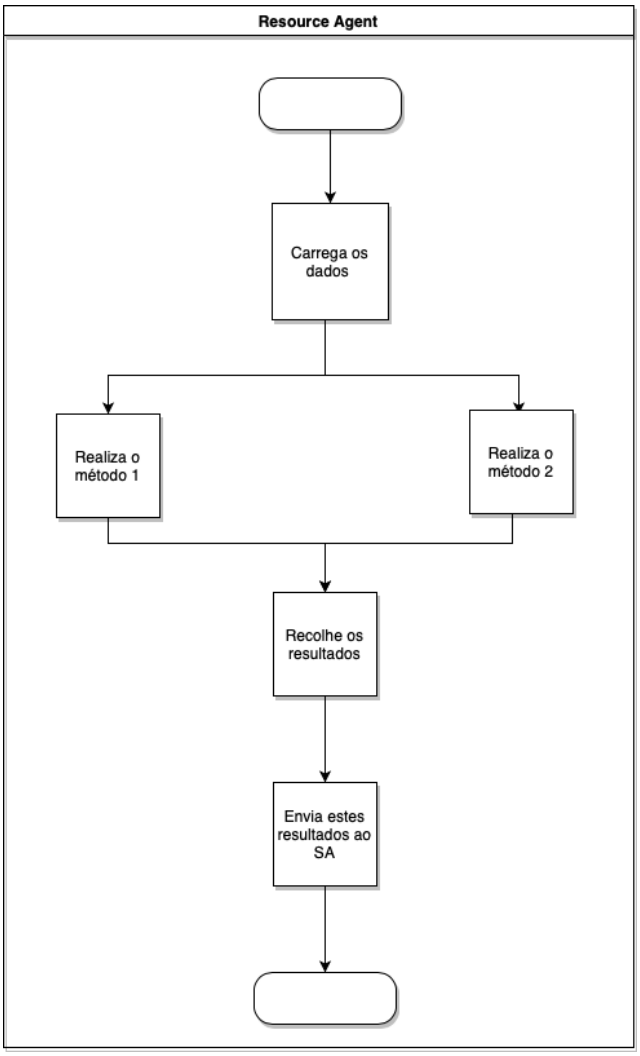

Figura 8 - Fluxograma do Resource Agent

Numa primeira fase o *RA* vai carregar os dados a partir de um recurso físico. Estes dados, antes de chegarem a esta fase irão ser alvo de processamento, de modo a chegarem prontos para serem utilizados. Este processamento, ocorre no bloco *KM*, falado anteriormente e será abordado detalhadamente no próximo capítulo.

Como foi dito anteriormente a escolha do método resulta das características adquiridas por o *RA*. Estes métodos serão abordados neste capítulo, de uma forma breve e no próximo capítulo serão explicados detalhadamente.

Depois de escolhido o método para prosseguir, estes dados serão processados e os resultados serão obtidos. Estes resultados estão continuamente a serem obtidos em tempo real, pois este agente realiza o fluxograma da figura anterior continuamente, parando unicamente por um tempo determinado quando é obtido um resultado inferior a um valor definido previamente,

quando processa todos os dados carregados ou quando o fabricante por algum motivo cancela a execução do sistema. Estes comportamentos irão ser explicados no próximo capítulo.

À medida que os resultados vão sendo obtidos, estes serão enviados para o *SA*. Como foi dito anteriormente, este agente irá realizar o processo da figura 7, enquanto não aconteça algum dos motivos explicados antes.

3.3.3 Subsystem Agent

Como descrito na tabela 2, o SA é responsável por recolher os resultados obtidos dos *RAs*. Será possível verificar o seu comportamento na figura 8.

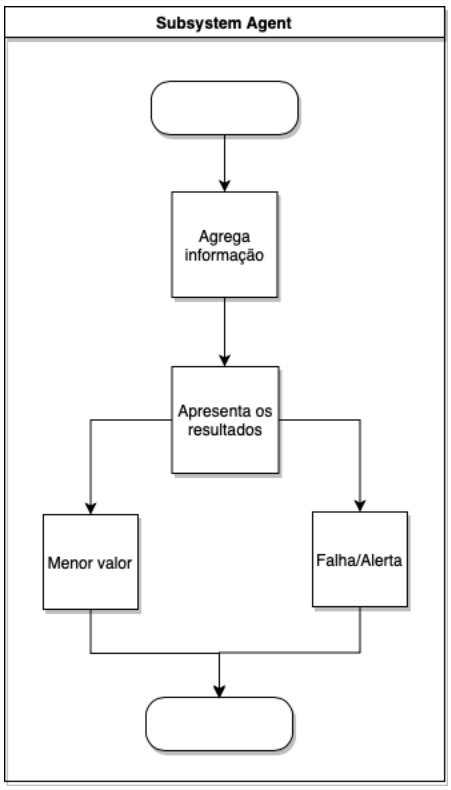

Figura 9 - Fluxograma do Subsystem Agent

O *Subsystem Agent* será inicializado ao mesmo tempo que o *RA*, pois mal o *RA* começa a execução, este começa logo a processar dados e a enviá-los para o *SA*. Quando este recebe os valores obtidos do *RA* agrega a informação e faculta algumas informações acerca destes valores.

Foram consideradas, como explicado anteriormente no *RA*, algumas características nos resultados para alertar o utilizador. O *SA* tem que disponibilizar sempre ao utilizador o menor valor dos resultados, e mostrar-lhe uma mensagem de alerta ou de falha quando este valor for menor que um determinado valor, definido antes da execução.

Este agente somente irá terminar quando o utilizador terminar ou cancelar a execução do sistema.

3.3.4 Escolha do método

Como explicado anteriormente, esta escolha ocorre depois do *RA* ter conhecimento de algumas características do sistema. Para o presente estudo foram tidos em conta alguns requisitos temporais, de modo a formular processamento de regras que irão influenciar a seleção.

Nos próximos capítulos explicar-se-á como estas características influenciam a escolha do método. De seguida, serão abordados os dois métodos de uma forma breve.

3.3.4.1 Cloud

Ao se usar, um *server* na *Cloud* é possível tornar o ambiente de trabalho de uma empresa mais integrado e fácil de gerir. Este método, como se pode observar na figura 9 é responsável por enviar os dados do *MAS* para um *server* na *Cloud*.

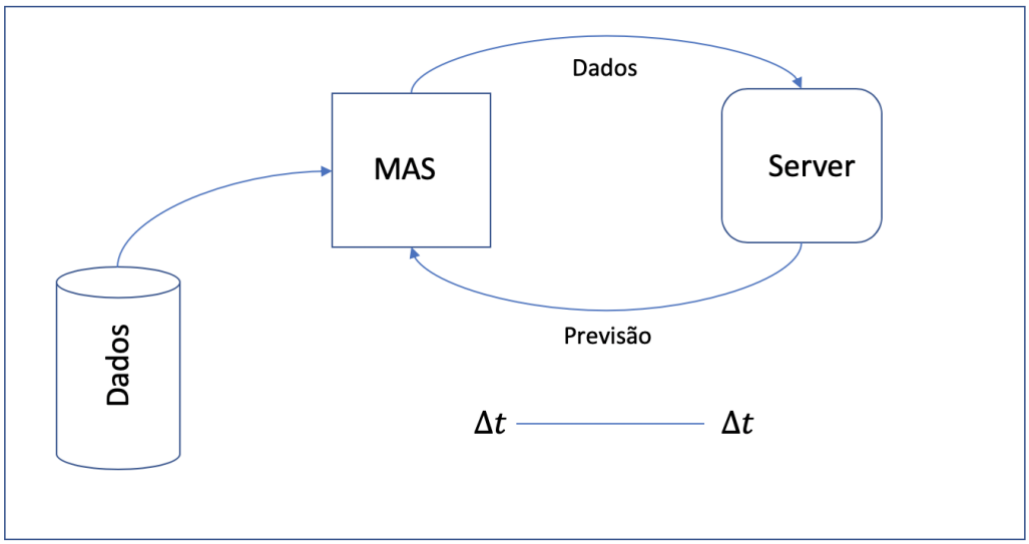

Figura 10 - Funcionamento do método 1

Por ser um *server* armazenado na *Cloud*, ou seja, que não existe "fisicamente", o sistema é executado num ambiente virtual, o que torna mais fácil a modificação de recursos. De seguida, é explicado na figura 10, o comportamento deste método.

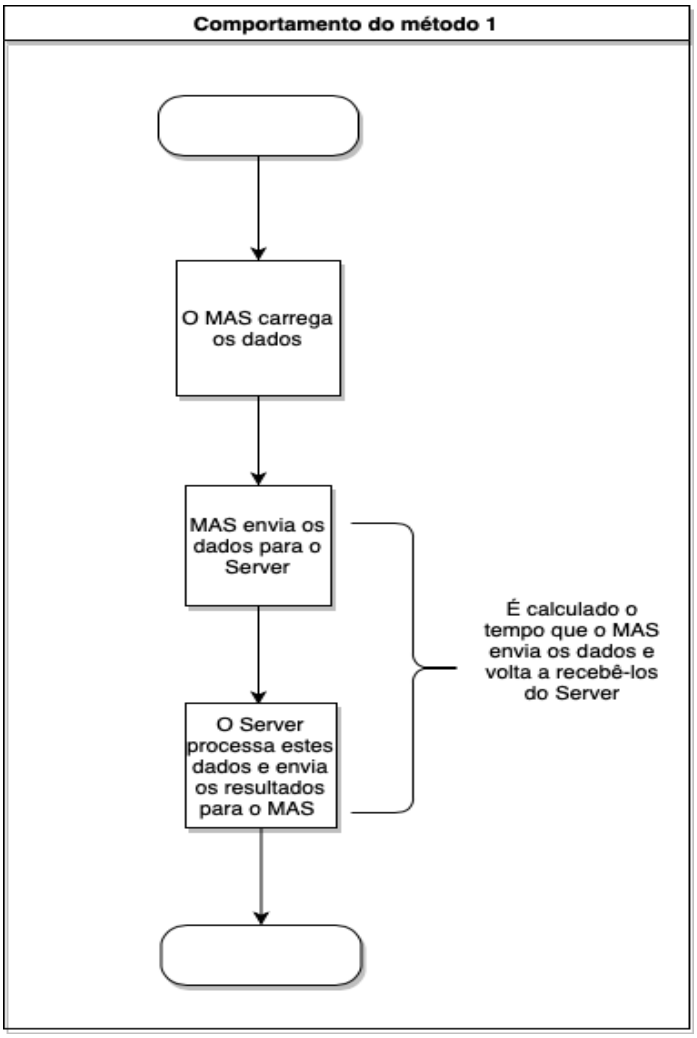

Figura 11 - Fluxograma do método 1

Como se pode observar a partir da figura anterior, numa primeira instância o MAS carrega os dados. De seguida, estes dados são enviados para um *server* na *Cloud*. Este *server* tem algoritmos que permitem processar os dados, que serão explicados no próximo capítulo. Depois de estes dados estarem processados, o *server* envia-os para o *MAS*.

No momento que o *MAS* envia os dados para o server é requisitado o tempo atual. Assim como quando recebe os dados do *server*. Desta maneira, é possível calcular a diferença de tempo, que será alvo de estudo no quinto capítulo.

3.3.4.2 Edge

Ao contrário do método anterior, neste método não é necessário fazer com que os dados viagem até à *Cloud* para serem processados, podendo ser tratados mais perto de onde foram recolhidos. Pode-se observar na figura 11, o funcionamento de uma forma geral deste método.

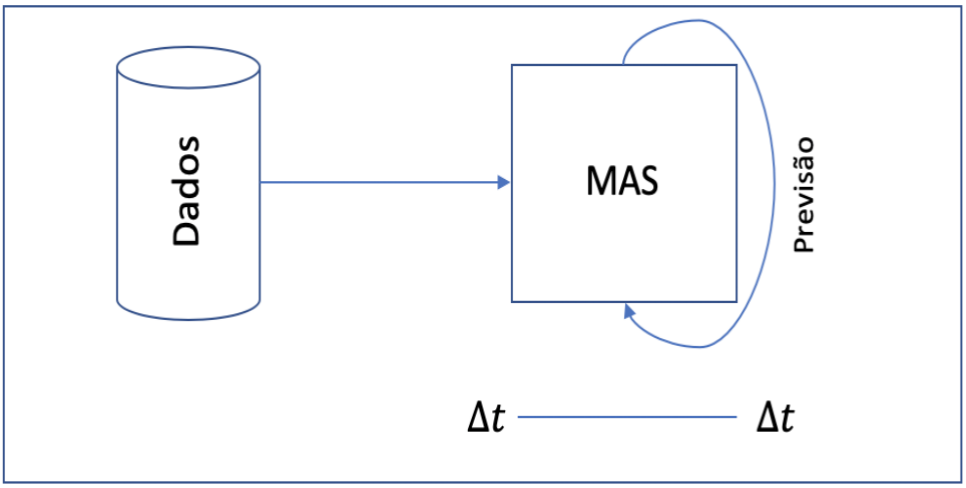

Figura 12 - Funcionamento do método 2

Como se pode verificar na figura acima, é tudo feito no *MAS*. Ao serem processados onde foram recolhidos, ganham-se algumas vantagens. Como por exemplo, permite melhorar a sua eficiência operativa e a experiência do cliente. De seguida, é explicado na figura 12, o comportamento deste método.

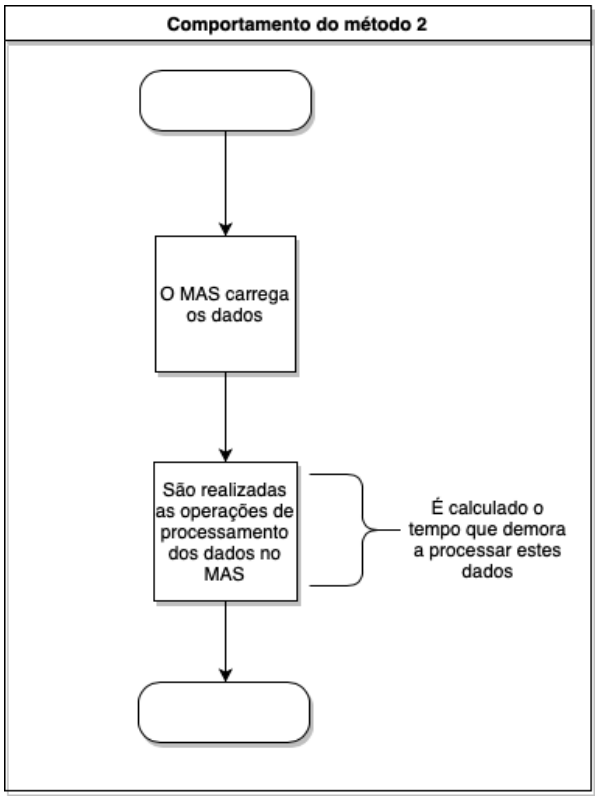

Figura 13 - Fluxograma do método 2

Como se pode observar na figura anterior, primeiramente os dados são carregados pelo *MAS*. Posteriormente, estes dados serão processados no *MAS*, que contém todos os algoritmos

necessários para este processamento. Estes algoritmos serão abordados detalhadamente no próximo capítulo.

Quando começa o processamento dos dados no *MAS* é requisitado o tempo atual. Bem como, quando termina este processamento. Desta forma, consegue-se obter a diferença de tempo, que depois será comparada com o método anterior em várias simulações.

3.4 Comunicação entre Agentes

Como foi analisado anteriormente, para que seja possível a interacção entre os agentes que compõem um Sistema Multi-Agente (*MAS*) é necessário que exista uma plataforma de comunicação, uma linguagem de comunicação e que os agentes possuam um vocabulário comum bem definido. Nesta solução pode-se destacar as comunicações existentes entre os agentes *DA*, *RA* e *SA*, de maneira a que os agentes comuniquem entre si para realizar as operações necessárias. A tabela 3, permite observar os agentes iniciantes e os participantes nas diversas comunicações entre eles realizadas.

Participantes / Iniciantes	DA	RA	SA
DA	-	**X**	-
RA	-	-	**X**
SA	-	**X**	-

Tabela 3 - Comunicação entre Agentes

4

4 Implementação

Neste capítulo será abordada a implementação da arquitetura detalhada no capítulo anterior. Em primeiro lugar, será explicada uma breve descrição das tecnologias utilizadas na implementação, seguindo-se uma descrição pormenorizada do trabalho efetuado e das decisões tomadas ao longo do trabalho.

4.1 Tecnologias de Suporte

Para que seja possível a implementação proposta no capítulo anterior, são necessárias algumas tecnologias. Seguem-se breves descrições relativas às tecnologias utilizadas, de modo a contextualizar a implementação da arquitetura.

4.1.1 Java

Java é uma linguagem de programação orientada a objetos. Esta linguagem foi utlizada como linguagem base da plataforma multiagente. Uma das vantagens de se ter utilizado esta linguagem é o facto de possuir uma grande comunidade que trocam informações e interagem entre si, partilhando os seus problemas e as suas soluções, tornando-se assim fácil resolver adversidades que aconteçam durante o desenvolvimento.

Foi utilizado como ambiente de programação o *Netbeans Integrated Development Environment,* pela sua robustez e versatilidade usando *Java*.

4.1.2 JADE

Para a implementação do *MAS* recorreu-se ao *JADE* (*Java Agent Development Framework*), uma plataforma para desenvolvimento de aplicações de agentes em *Java* (Bellifemine, Caire, & Greenwood, 2007). Existem muitas vantagens da utilização desta plataforma, entre as quais, a sua utilização em várias aplicações devido à sua flexibilidade, boa documentação e à sua capacidade de atender aos padrões de comunicação entre os agentes (Carrasco et al., 2014).

Esta plataforma garante a comunicação entre os vários agentes através das especificações impostas pela *Foundation for Intelligent Physical Agents*, referenciado na literatura como **FIPA**. Existem dois principais protocolos de comunicação entre agentes **FIPA**, o FIPA *Request* e o FIPA *Contract Net*.

Assim, ao usarmos esta plataforma, é possível a execução de múltiplas tarefas, pois esta modela os comportamentos dos agentes.

4.1.2.1 FIPA Request

O FIPA *Request* permite a um agente solicitar a realização de uma certa ação a outro agente (FIPA, 2002).

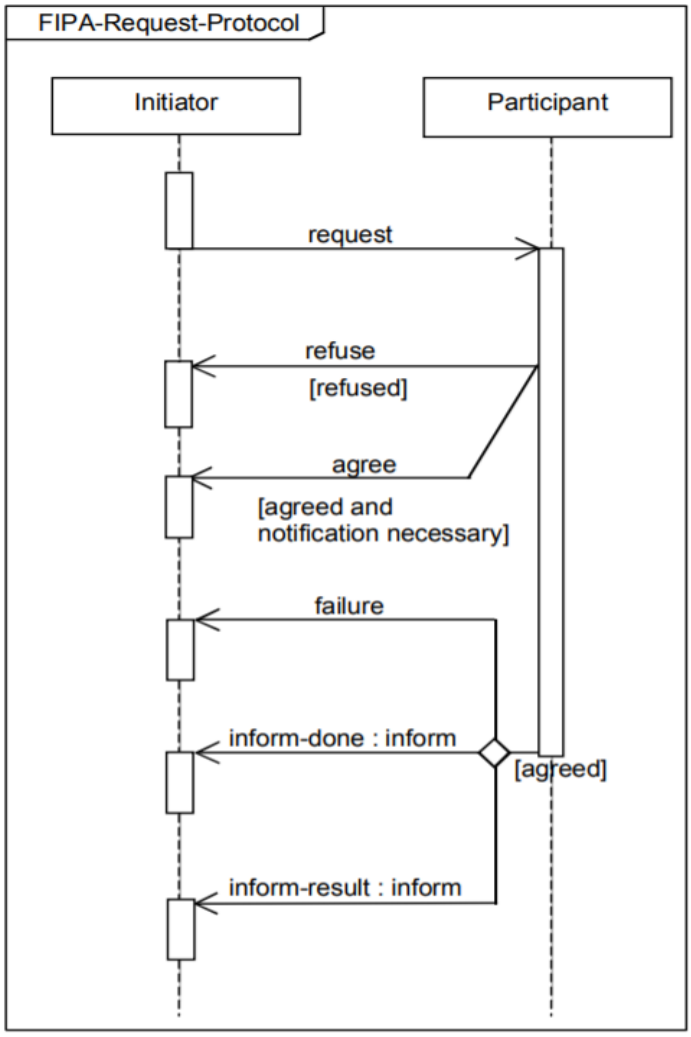

Figura 14 - Protocolo FIPA Request (adaptado de FIPA, 2002)

Como se pode verificar na figura 13, para dar início a este protocolo, o agente iniciante deve enviar um pedido ao participante.

Após este pedido, o participante pode optar por aceitar ou recusar enviando uma mensagem consoante seja a opção. Caso aceite, executa o pedido e envia uma resposta, ou seja, se este foi bem-sucedido ou não, podendo ainda enviar o resultado. Caso contrário envia uma mensagem de falha.

4.1.2.2 FIPA Contract Net

O FIPA *Contract Net* oferece a possibilidade de negociação entre o agente iniciante e outros agentes participantes, permitindo ao iniciante avaliar diferentes propostas e escolher a que melhor se adapta às suas necessidades (FIPA, 2002).

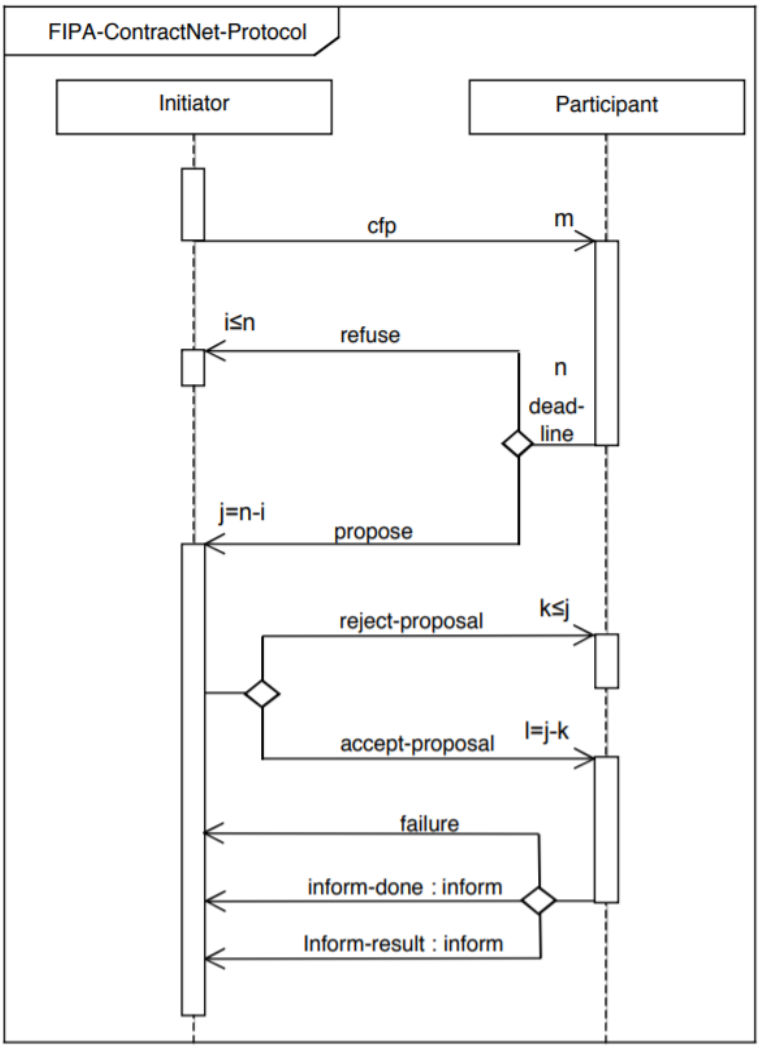

Figura 15 - Protocolo FIPA Contract Net (adaptado de FIPA, 2002)

Este protocolo como se pode verificar na figura 14, tem início com um pedido de proposta aos m agentes participantes. Estes por sua vez, podem recusar ou responder com uma proposta. Após o iniciador receber todas as mensagens, quer sejam rejeições ou propostas, este vai avaliar as mesmas, para saber quais são as mais favoráveis.

Cada proposta vai ser então respondida de acordo com a avaliação feita, ou seja, é enviado uma mensagem para aceitar ou recusar a proposta efetuada. Por sua vez, os participantes depois de realizarem a ação respondem com uma mensagem a informar que a ação foi bem-sucedida, ou caso contrário enviam uma mensagem de falha.

4.1.3 Scikit-learn

Para a realização da análise de dados foi utilizado o *Scikit-learn*, que é um módulo em *Python* para *ML*. As vantagens associadas a este módulo são várias, entre as quais: o fornecimento de ferramentas simples e eficientes para análise de dados e o facto de ser acessível a todos e reutilizável em diversos contextos

4.1.4 Sockets

No método um, depois de já analisados no, *MAS*, foi necessário estabelecer a comunicação entre este e o *server* na *Cloud*. Para tal, foi utilizado *Sockets*. Estes são um mecanismo de comunicação, usado normalmente para implementar um modelo cliente/servidor, que permite a troca de mensagens entre os processos de uma máquina servidor e uma máquina cliente. Apesar de poderem ter sido usadas outros tipos de tecnologia como serviços, optou-se pela utilização de *sockets* devido à sua performance em comparação com outros tipos de serviços.

4.2 Implementação do trabalho proposto

Para o desenvolvimento do trabalho foram necessárias várias etapas. Numa primeira fase, foi necessário entender os dados e como analisar estes. Para tal, recorreu-se a *ML*. Devido a não ter qualquer conhecimento na área antes, foi imprescindível a pesquisa sobre a mesma. Como *ML* tem bastante informação online, esta pesquisa foi mais fácil.

A segunda etapa prende-se com a adaptação à ferramenta que implementa o sistema multiagente. Deste modo, foi necessário estudar a variadíssima documentação que é disponibilizada pela própria plataforma. Como foi dito anteriormente, devido a ter uma vasta documentação online, foi possível uma aprendizagem rápida e fácil ao ambiente multiagente. Para além disso, o facto de já ter realizado trabalhos com recurso a esta plataforma foi importante, pois já tinha presente os conceitos e os comportamentos de agente, bem como o funcionamento desta plataforma.

Depois foi necessário pesquisar como conectar os dados analisados no sistema multiagente para a realização do método 2. Posteriormente, depois de já ter o sistema multiagente construído e a processar os dados, foi possível proceder à criação do server para a realização do método 1. Para esta fase, foi necessário pesquisar em que linguagem de programação seria este e como ligá-lo ao *MAS*.

Por último, depois de ambos os métodos, introduzidos no capítulo anterior estarem a funcionar, procedeu-se à realização de testes e validações no ***Raspberry pi* 3 *Model* B**, de modo a emular a aquisição de dados no recurso físico.

4.2.1 Análise dos Dados

Aqui será explicado como é realizada a análise dos dados, executada no bloco *KM*, introduzido no capítulo anterior. Uma das principais partes deste trabalho, era analisar os dados provenientes de uma frota de motores do mesmo tipo. Cada motor começava com diferentes graus de desgaste inicial e variação de fabricação, desconhecidos pelo usuário.

Na figura 15, observa-se como se procedeu à análise destes dados, sendo explicado cada bloco desta figura detalhadamente a seguir.

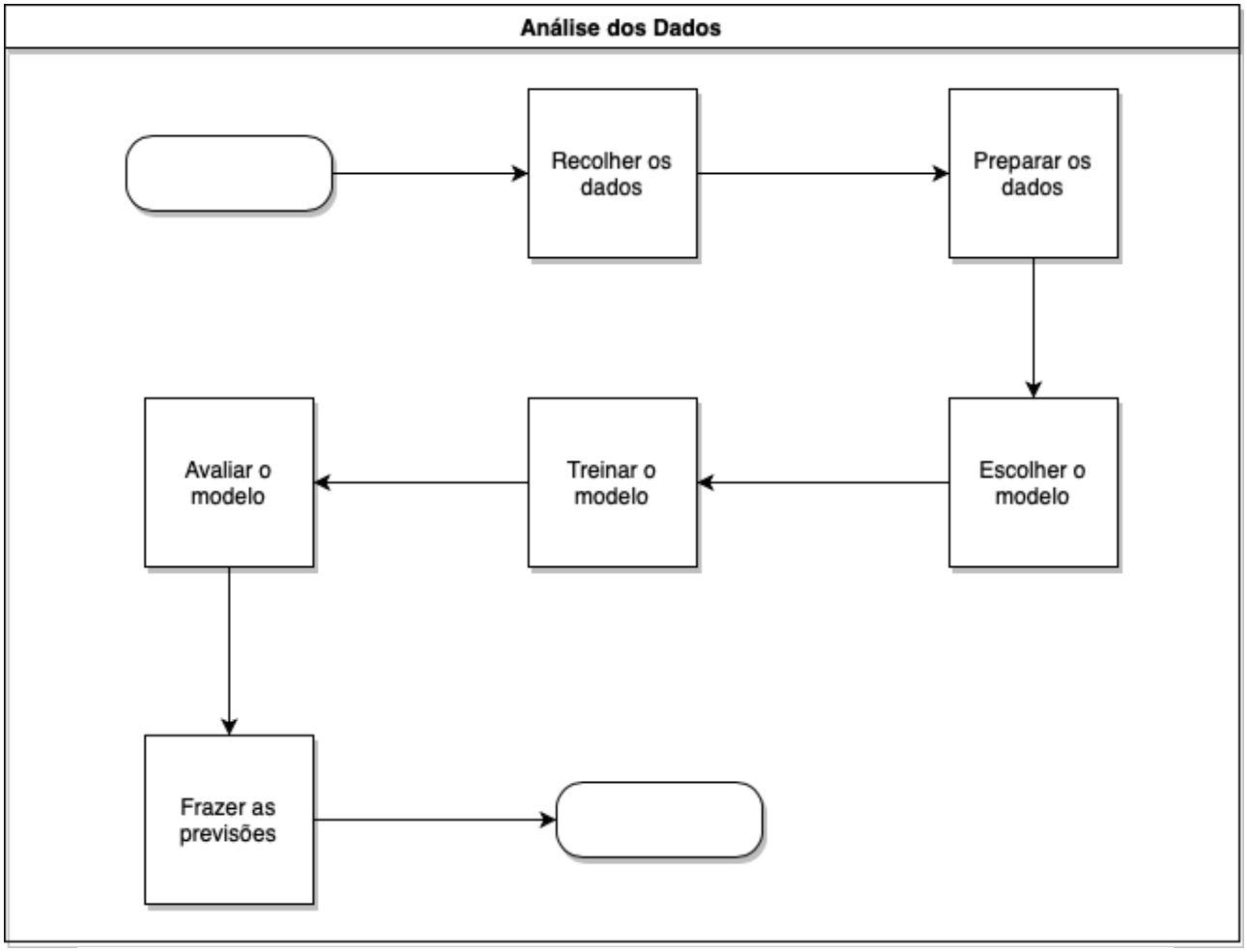

Figura 16 - Passos Fundamentais para a Análise dos Dados

Como se pode observar na figura acima descrita, o primeiro passo a realizar é recolher os dados e prepará-los de seguida. Estes dados foram gerados a partir de medições em motores de turbina. Os dados gerados foram usados como dados de desafio para a competição *PHM* (*Prognostics and Health Management*) 2008. O objetivo desta competição, assim como o deste estudo, é prever o número de ciclos operacionais restantes antes da falha ocorrer.

Antes de se poder proceder à análise destes dados, recorrendo a técnicas de ML, é fundamental conhecer estes antes. De seguida, será explicado na tabela 4, o que cada coluna representa.

Colunas	Variáveis
1	Número de unidade
2	Tempo, em ciclos
3	Configuração operacional 1
4	Configuração operacional 2
5	Configuração operacional 3
6	Medição do sensor 1
7	Medição do sensor 2
...	
26	Medição do sensor 26

Tabela 4 - Constituição das colunas

Como se pode observar a partir da tabela acima, estes dados têm 26 colunas, onde cada coluna é uma variável diferente. Cada linha é uma captura instantânea de dados obtidos durante um ciclo operacional.

Estes dados já vinham divididos em dados de treino e de teste, mas só para a variável de entrada, necessitando de se fazer os mesmos para a variável *target*. No conjunto de dados fornecido vinha informação para a realização do passo anterior. Depois de já ter estes dados corretamente divididos, procedeu-se então à escolha do algoritmo de *ML* para prever os resultados.

De modo a selecionar o modelo final para prever os resultados, foram realizadas algumas comparações que serão descritas de seguida. Inicialmente foram selecionados três modelos, *Random Forest, Gradient Boosted* e *K-Nearest Neighbors*.

De seguida, será explicado como foram avaliados os modelos anteriormente enunciados. Existem vários tipos de métrica para avaliar modelos de *ML*. Neste trabalho foi usado:

- Erro médio absoluto (*MAE*)
- Precisão

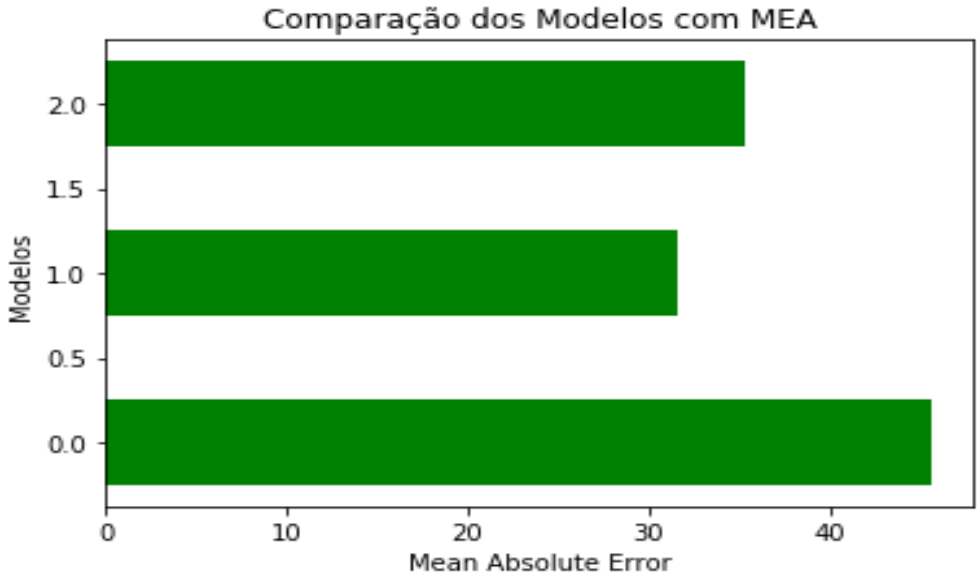

Figura 17 - Erro Médio Absoluto (MAE)

	MAE (ciclos)	Precisão (%)
Gradient Boosted	35.25	72.44
Random Forest	31.6	75.36
K-Nearest Neighbors	45.6	66.42

Tabela 5 - Comparação dos modelos

Como se pode observar a partir da figura 16 e da tabela 5 o modelo de *ML* escolhido foi o *Random Forest* (*RF*), pois reúne a melhor precisão e o menor *MAE*. Para além disso, este é dos algoritmos mais simples e mais usado. O *RF* é um conjunto de árvores de decisão aleatórias, logo enquanto cada árvore de decisão dá um voto para a previsão da variável de destino, o *RF* escolhe a previsão que obtém mais votos, aumentando assim a precisão. Uma das vantagens associadas a este modelo é a eficiência em grandes conjuntos de dados e a capacidade de lidar com vários recursos de entrada sem a necessidade de exclusão destes. Pelo que foi enunciado anteriormente, considerou-se o *RF* o modelo adequado.

Depois da escolha do modelo, é tempo de treinar este modelo. Isto significa que é fornecido todos os dados de treino para o algoritmo treinado e vamos deixar este algoritmo descobrir o mapeamento entre as entradas e a saída que minimiza o erro de previsão.

Por último, com o modelo já completamente treinado foi possível obter as previsões do mesmo. Durante este processo todo, houve a necessidade de gravar o modelo de *ML*, de modo, a que possa ser usado a qualquer momento no futuro, evitando assim o desperdício de tempo de treinamento. Para este efeito, usou-se a biblioteca do *scikit-learn* ***joblib.***

Pode-se observar na figura 17, como foi realizado cada um dos passos anteriormente descritos.

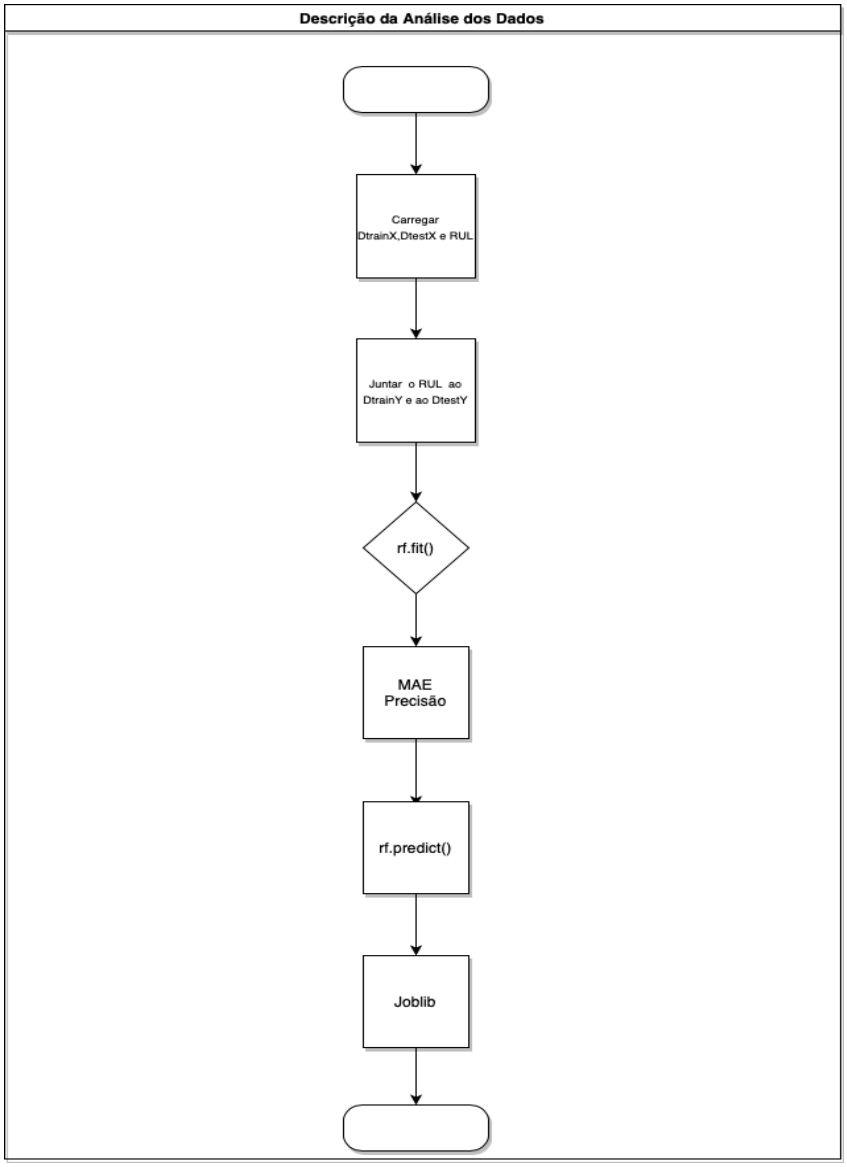

Figura 18 - Descrição da Análise dos Dados

Como se pode observar a partir figura 16 e da tabela 5, os resultados do modelo escolhido não foram os melhores, portanto houve a necessidade de tentar resolver este problema. Um dos primeiros passos que se deve fazer é reunir mais dados, mas neste caso já não era possível pois foi usado um conjunto de dados que já tinha os dados reunidos, usou-se então o modelo *hyperparameter tuning*. De seguida será explicado na figura 18 como foi realizado.

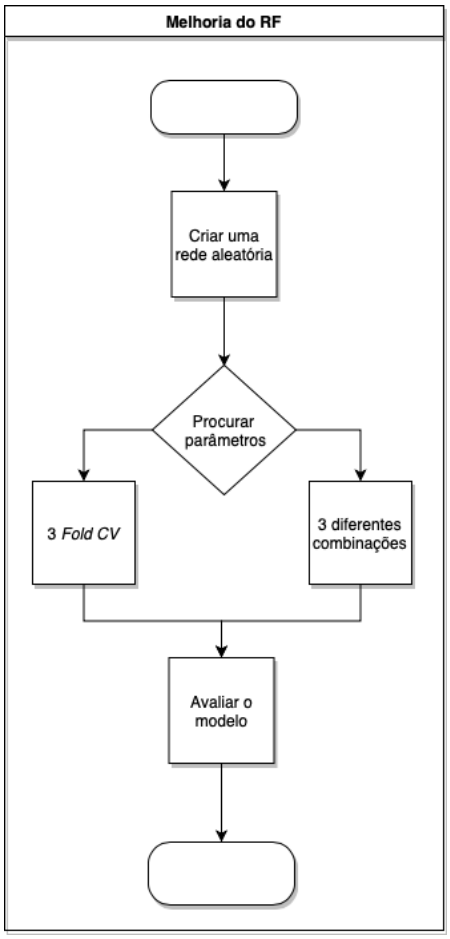

Figura 19 - Melhoria do RF

Como se usou o *RandomizedSearchCV* o primeiro passo foi criar uma rede aleatória de parâmetros. De seguida usou-se esta rede aleatória para procurar os melhores parâmetros. Para esta procura usou-se 3 vezes validação cruzada ao longo de 3 combinações diferentes. Por último, para determinar se esta pesquisa aleatória produziu um modelo melhor comparamos o modelo base com este modelo de pesquisa aleatória.

4.2.2 Deployment Agent

Como referido no capítulo 3, antes de lançar os agentes, o *DA* adquire informações a partir de uma fonte externa. Este ficheiro contém o número de *RAs* que devem ser lançados no sistema. Este método é ilustrado na figura 19.

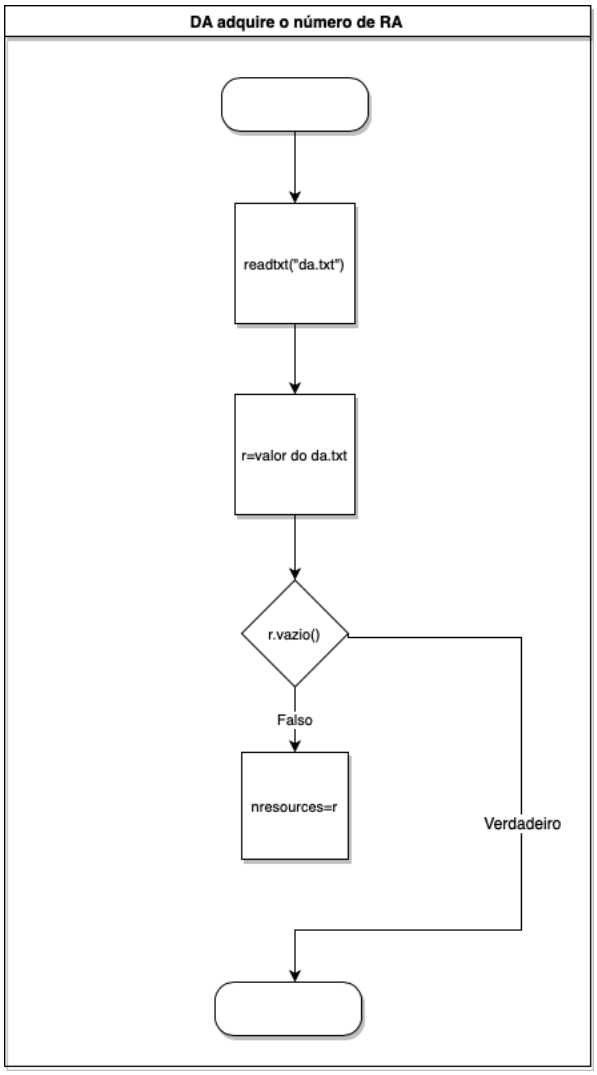

Figura 20 - Implementação do DA

O agente começa por analisar o ficheiro, obtendo o número que estiver nesse ficheiro. Caso esse número seja positivo, este número será o número de *RA* que irá ser lançado no início de execução do sistema.

Por outro lado, se este ficheiro estiver vazio, se tiver um número negativo ou um valor nulo, não serão lançados *RAs* no momento de execução do sistema, pois, como dito anteriormente o *DA* só lida com valores positivos no ficheiro.

4.2.3 Resource Agent

Como explicado anteriormente, este agente é o mais importante do sistema, pois é neste que estão os mecanismos indispensáveis para a realização deste estudo. O número deste tipo de agentes que irá ser lançado no sistema já foi obtido pelo *DA*, caso o fabricante opte por usar este agente ou então serão introduzidos manualmente pelo fabricante. Todo o comportamento deste agente, desde o momento a que é lançado, até ao momento em que a execução do sistema é terminada é ilustrado na figura 20.

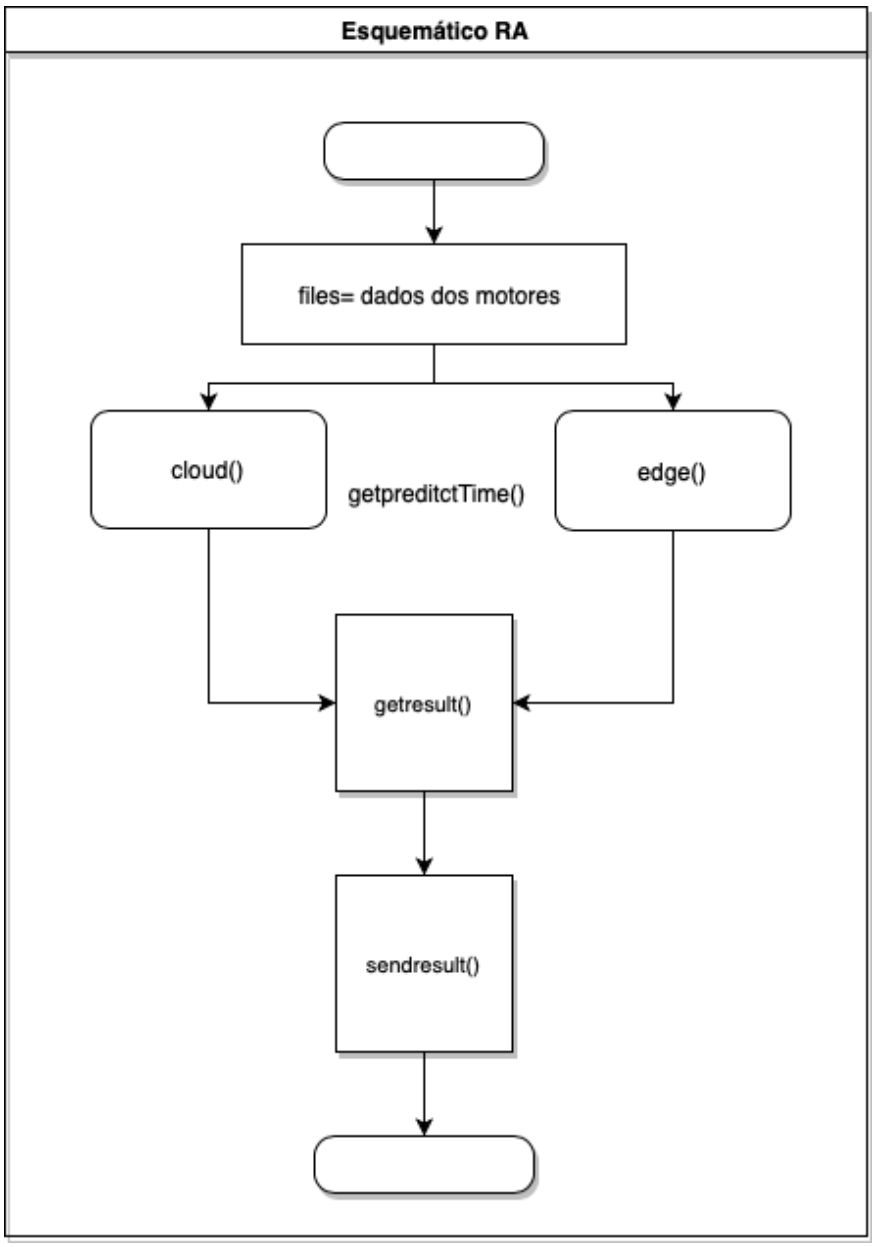

Figura 21 - Implementação do RA

Numa primeira parte, todos os dados vão ser postos num *ArrayList* de objetos. Consoante o número de *RAs* que será lançado, cada um destes vai automaticamente carregar um número de dados, dados estes que serão distintos uns dos outros, de modo a que cada *RA* não preveja o mesmo que os outros. Depois de já ter estes organizados, vai ser possível selecionar qual o método que deve ser escolhido para o processamento destes dados. Como referido no capítulo anterior, a escolha pode recair em dois métodos.

Por um lado, temos o método *Cloud*. Este já foi introduzido no capítulo anterior e será explicado detalhadamente neste capítulo. Por outro lado, existe o método *Edge*, que tal como o anterior será alvo de estudo neste capítulo. Estes dois métodos distintos, serão alvo de comparação no próximo capítulo.

Quando começa cada um destes métodos é requisitado o tempo atual e o mesmo será requisitado quando acabar o processamento de uma linha destes dados, calculando assim o ***getpredictTime()***. Cada resultado é obtido a partir do método ***getresult()*** e de seguida é enviado para o *SA*, usando o método ***sendresult()***.

De modo a correr todas as linhas dos dados periodicamente, recorreu-se a um *TickerBehaviour*. O motivo desta escolha prende-se pelo facto, deste analisar todo o sistema num intervalo escolhido. Estes tempos serão referidos no capítulo seguinte. Este *TickerBehaviour* está representado na figura 21.

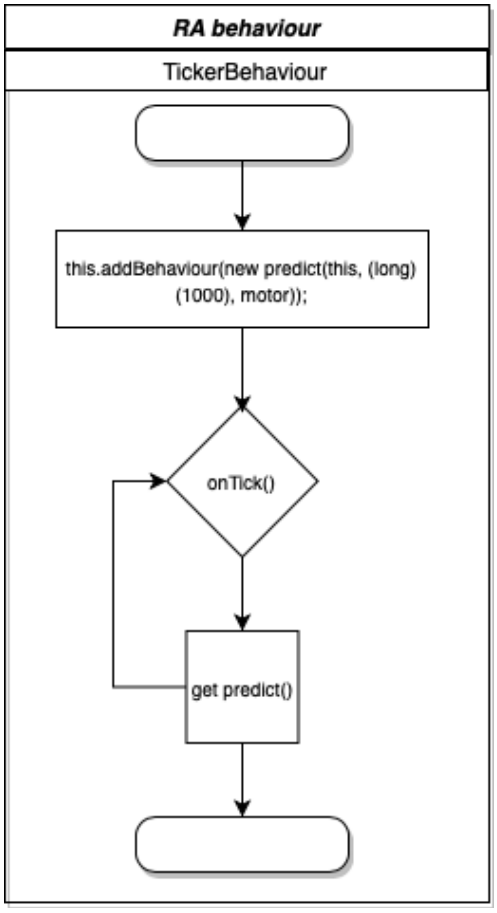

Figura 22 - Behaviour responsável por correr tarefas periodicamente

Este *behaviour* é acionado assim que o agente inicia a sua execução, com um intervalo de execuções que pode ter 3 valores distintos. Estes valores foram alterados durante as simulações e serão divulgados no próximo capítulo.

4.2.3.1 Seleção do método

Como explicado anteriormente, o *RA* vai obter conhecimento de algumas características do sistema. De seguida será explicado quais são estas características e como estas influenciam a escolha do método. Para a escolha do método foram utilizadas três características:

- Tamanho do conjunto de dados
- Número de *RAs* em execução
- Tempo (ms) do *TickerBehaviour*

Pode-se observar na figura 22 como esta escolha é realizada, tendo em conta as características anteriormente apresentadas.

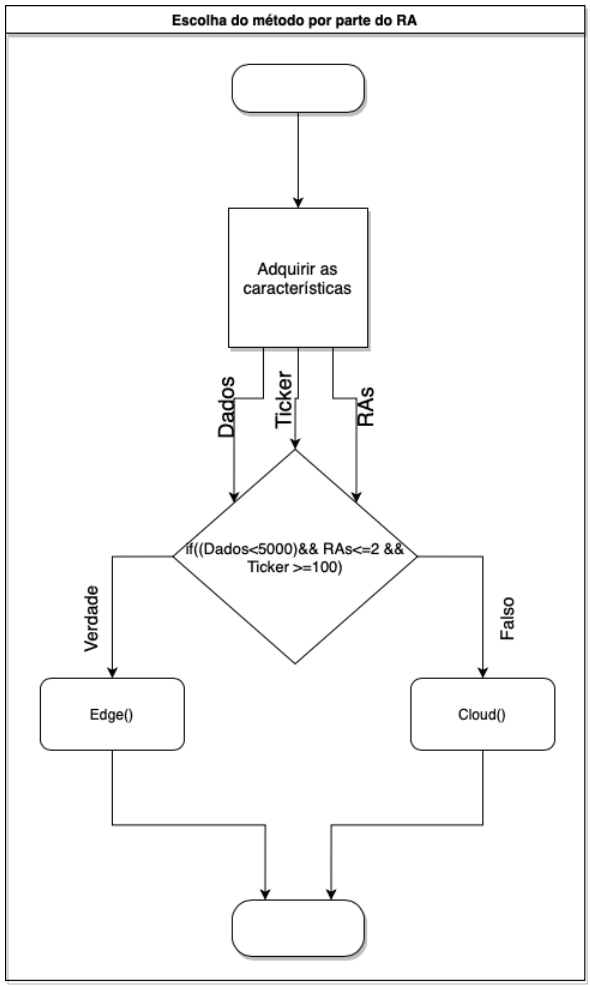

Figura 23 - Escolha do método por parte do RA

Como se pode observar a partir da figura anterior, numa primeira instância o *RA* vai adquirir as características enunciadas anteriormente. De seguida, vai realizar uma condição, comparando as três características. Se essa condição for verdadeira o método selecionado é o *edge()*, caso contrário o método *cloud()* vai ser selecionado. Esta escolha consoante a comparação das características só foi possível após uma primeira simulação dos resultados, na qual foi possível perceber de que maneira estas características influenciavam o comportamento do sistema, de modo a criar a condição representada na figura 22.

4.2.4 Subsystem Agent

Este agente vai recolher os resultados enviados pelos *RAs*, apresentado os mais importantes ao utilizador. Todo o comportamento deste agente é ilustrado na figura 23.

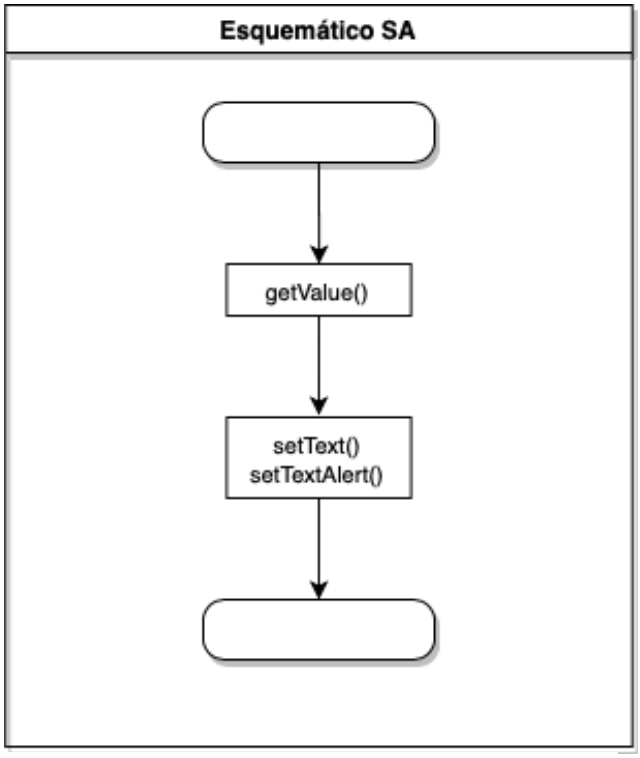

Figura 24 - Implementação do SA

Como se pode observar na figura anterior, o SA vai obter os valores do *RA* a partir do método ***getValue()***. De seguida, vai apresentar estes valores ao utilizador a partir de dois métodos. Um deles é o método ***setText()***, que vai apresentar o menor valor dos resultados enviados pelos *RAs*. O segundo método é o ***setTextAlert()***, que vai lançar um alerta quando o resultado for menor que o valor escolhido por o fabricante para avisar uma possível falha.

4.2.5 Cloud

Depois de já ter os dados analisados recorrendo a técnicas de *ML* e o sistema multiagente construído, procedeu-se à criação do *server*. Como foi exposto no capítulo anterior, o objetivo deste é processar os dados que estão no *MAS* num *server* na *Cloud*. De modo a conectar o *MAS* e este *server*, foi usado *sockets*. Será explicado de seguida, na figura 24 como é realizado esta comunicação.

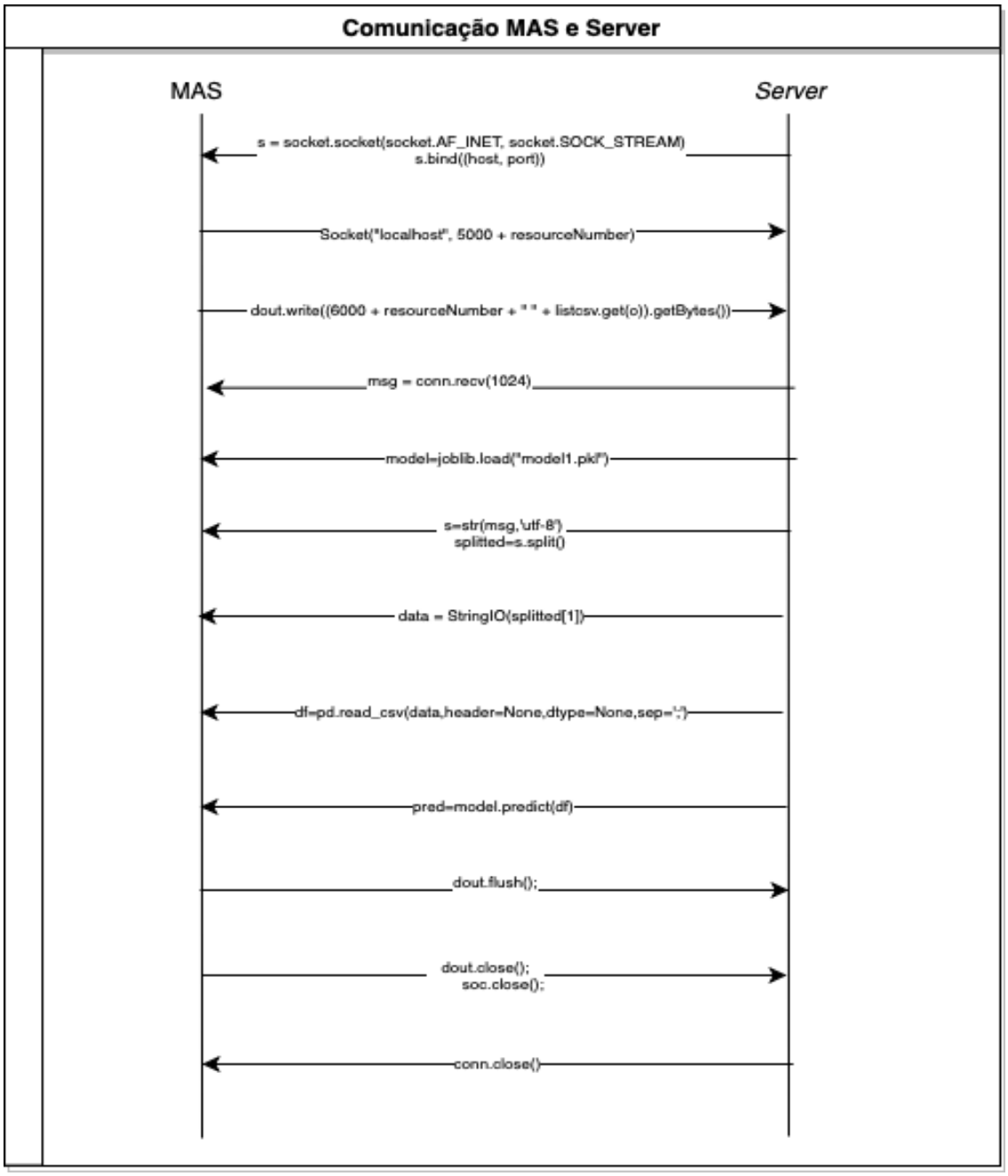

Figura 25 - Comunicação entre o MAS e o Server

Na figura 24, é possível visualizar como é realizada a comunicação entre o *MAS* e o *Server* usando sockets. Esta figura demonstra a comunicação entre um Java cliente, que é o *MAS* e um *Python server,* que é um *server* na *Cloud.*

Este *server* tem os algoritmos necessários para processar os dados, pois, como se pode verificar na figura acima, depois de este receber os dados do *MAS*, este carrega o modelo previamente guardado usando a biblioteca do *scikit-learn* ***joblib***, necessitando depois de fazer o ***model.predict()*** para obter o *Remaining Useful Life* (*RUL*), que é o objetivo final.

Também foi necessário realizar a comunicação contrária, pois o objetivo deste método é enviar os dados do *MAS* para um *server*, onde este processa os dados e depois este enviar de volta os dados processados, ou seja, o *RUL* para o *MAS*. A figura 25 explica essa comunicação.

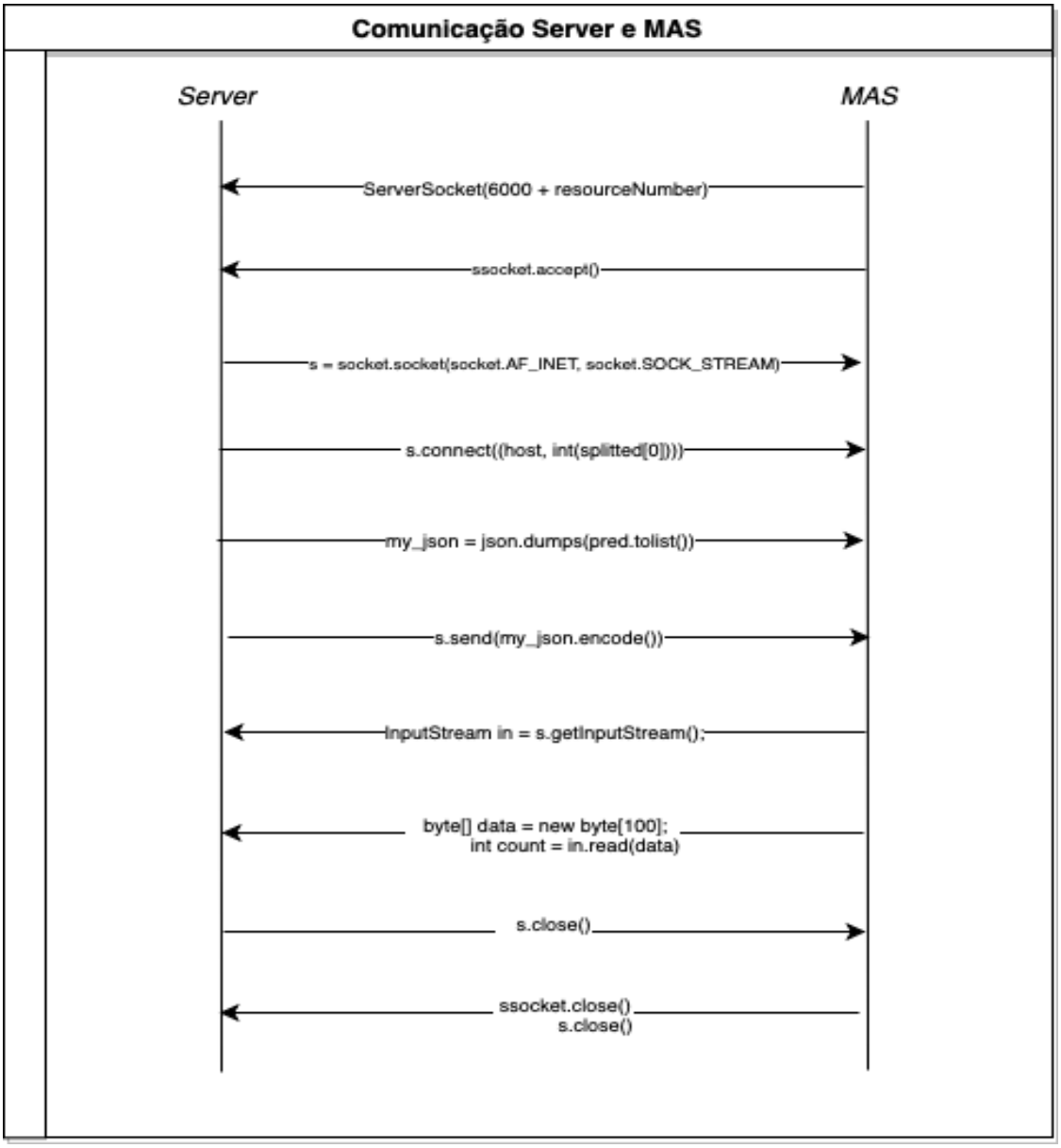

Figura 26 - Comunicação entre o Server e o MAS

Desta vez, a figura anterior demonstra a comunicação entre um *Python* cliente, que é um *server* na *Cloud* e um *Java server*, que é o *MAS*. Depois de o *server* ter processado os dados, e ter obtido os *RULs*, é necessário passar estes para o *MAS*. Como referido anteriormente, é calculado o tempo que demora a fazer estas duas comunicações.

4.2.6 Edge

Ao contrário do método anterior, neste método todo o processamento dos dados é realizado localmente. Numa primeira fase, depois de ter o modelo *ML* realizado, foi necessário introduzir este modelo no *MAS*. Como a linguagem de programação do *MAS*, é *Java*, houve a necessidade de saber como conectar o modelo de *ML* à linguagem *Java*. Na figura 26, pode-se observar os passos necessários para esta ligação.

Figura 27 - Sklearn2pmml (adaptado de Bensrhir, 2017)

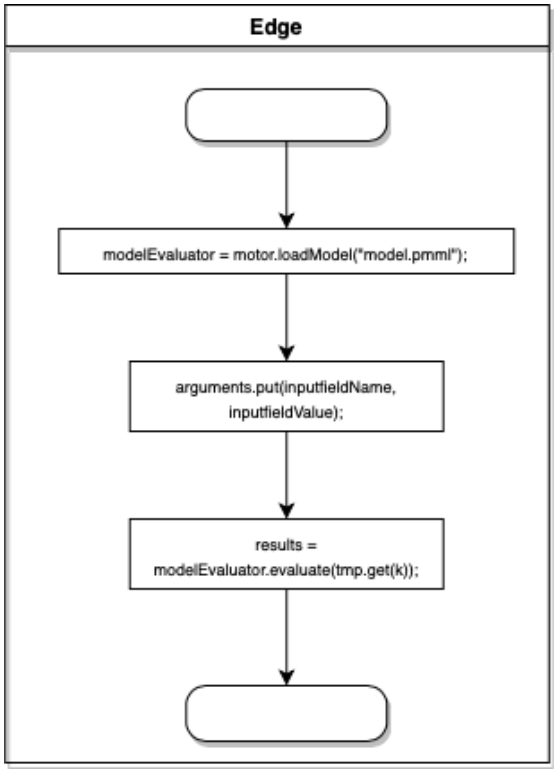

Figura 28 - Comportamento do método *Edge*

Como se pode observar na figura 27, o *MAS* importa o ficheiro **pmml** numa primeira instância. De seguida, começa-se a construir um avaliador de previsão para o modelo carregado. Por último, realiza-se a previsão, obtendo os *RULs*. Tal como no método anterior, aqui também é calculado o tempo que demora a acontecer o que foi descrito antes.

5 Validação e Testes

Este capítulo descreve os testes executados para validar a implementação da arquitetura proposta. Na secção 5.1 é apresentado o comportamento do sistema nos dois métodos, sob diferentes *workloads*. Por fim, na secção 5.2 é realizada uma comparação entre estes 2 métodos, bem como a discussão destes resultados.

Estes testes, como foi referido anteriormente, foram realizados no ***Raspberry pi* 3 *Model* B**, de modo a emular a aquisição de dados no recurso físico.

O ***Raspberry pi*** é um computador de baixo custo, de dimensões reduzidas, que permite realizar diversas coisas que um computador normal possibilita. O ***Raspberry pi* 3 *Model* B** tem os seguintes aspetos técnicos:

- Processador de 4 núcleos ARMv8 1.2 GHz
- 1 GB RAM

Utilizou-se assim este ***Raspberry*** para alocar tanto o *MAS*, como o *server* usado no método um. De realçar também, que apesar deste *server* na *Cloud* estar na mesma máquina que o *MAS*, este podia ser um *Server Cluster* local ou remoto.

5.1 Testes

O propósito destes testes é comparar os dois métodos anunciados nos capítulos anteriores. De modo a executar esta tarefa, foi necessário realizar algumas simulações que serão explicadas neste capítulo. Para os testes foram feitas quatro simulações, uma com um *RA*, outra com dois *RAs*, outra com três *RAs* e por último com quatro *RAs*. O objetivo da realização de quatro simulações é verificar o comportamento do sistema com diferentes requisitos.

Todas estas simulações foram realizadas sob as mesmas condições, que serão enunciadas de seguida:

- Calcular a médio do tempo que demora a prever cada linha após cem previsões;
- Os tempos usados pelo *TickerBehaviour* são 100 ms, 500 ms e 1000 ms;
- De modo a comprovar a veracidade dos resultados, cada uma destas quatro simulações foi realizada três vezes.

Este número de cem previsões foi escolhido, pois após este número já é possível obter uma boa média para servir de comparação.

Os tempos de cada *TickerBehaviour* são 1000 ms, 500 ms e 100 ms. Estes tempos vão ser a janela de tempo entre execuções. Isto significa que as tarefas, ou seja, o método um e o método dois, vão ser executadas periodicamente. Numa primeira instância, as tarefas vão ser executadas de 1 em 1 s, depois serão executadas de 500 ms em 500 ms e por último de 100 ms em 100 ms.

Para além destas duas condições acima descritas, cada uma destas simulações foram realizadas três vezes. Deste modo, foi possível comprovar se estes resultados estavam a seguir algum padrão ou não.

Como foi explicado anteriormente, esta arquitetura conta com três tipos de agentes, o *RA*, o *DA* e o *SA*. De seguida, será apresentada a interface do SA. Não serão apresentadas a interface do *DA*, pois este agente é responsável por lançar um determinado número de agentes no sistema e não tem interface e o *RA* será explicado no próximo subcapítulo.

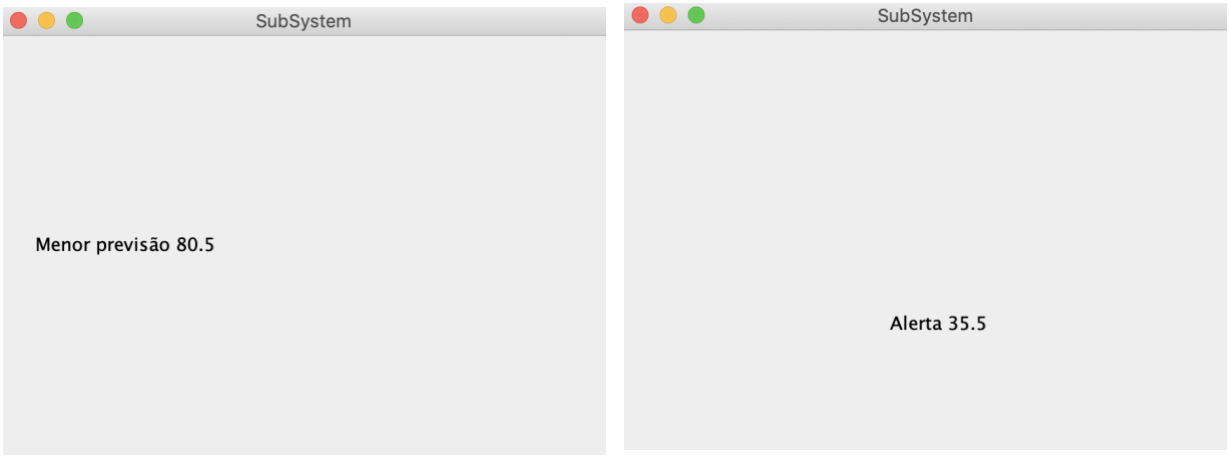

Figura 29 - Interface SA

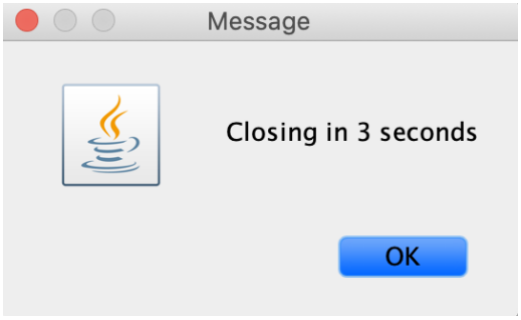

Figura 30 - Alerta

O *SA* como foi explicado anteriormente vai obter os valores dos *RAs* e apresentar ao utilizador o menor valor e um alerta quando um valor é inferior a um determinado valor, como se pode observar a partir das figuras 28 e 29.

5.1.1 Primeira simulação

Nesta primeira simulação, será comparado o comportamento do sistema usando um *RA*, tanto para o método *Edge*, como para o método *Server*. Pode-se observar na tabela 6, o comportamento desta primeira simulação.

1 Resource Agent		
	***Edge* (ms)**	***Cloud* (ms)**
100 ms	5589	139
500 ms	5632	158
1000 ms	5720	181

Tabela 6 - Comportamento do Sistema com 1 RA

Na tabela anterior é possível verificar os tempos que demora a prever cada linha dos dados em ambos os métodos. Estes dois métodos serão agora alvo de comparação.

Tanto no método *Edge*, como no método *Cloud* existe uma diminuição deste tempo à medida que o valor do *TickerBehaviour* diminui. Este resultado já se estava à espera pois como as tarefas vão sendo executadas mais rápidas, menor é o tempo que demora a prever cada linha.

Para além do tempo que demora a prever cada linha dos dados, também foi calculado o tempo de execução, ou seja, após atingir as cem previsões para os dois métodos. Na figura 30 é possível verificar este tempo.

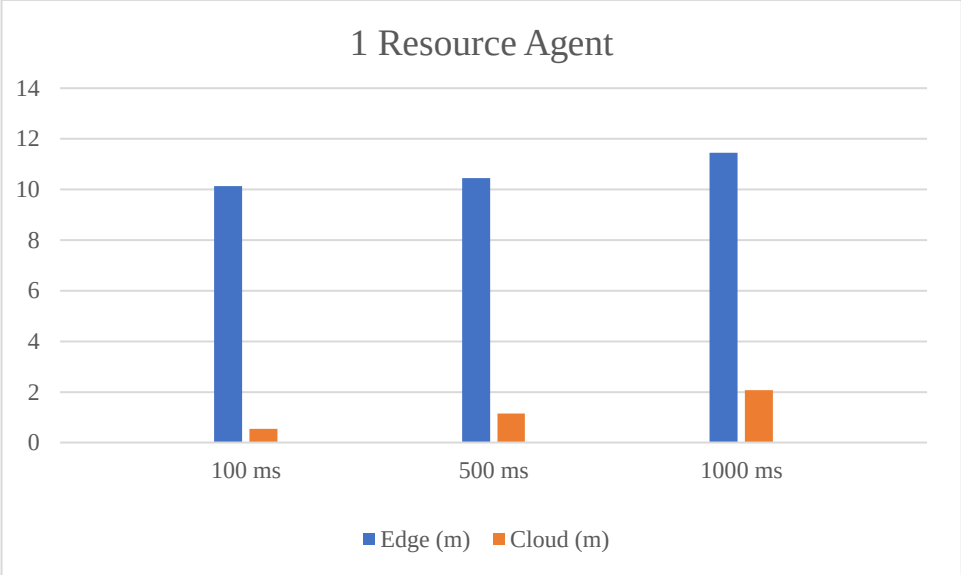

Figura 31 - Tempo de execução do sistema com 1 RA

Como seria de esperar, o tempo de execução é maior no método *Edge* do que no método *Cloud*. Este tempo, como se pode observar na figura 30 vai aumentando em ambos os métodos à medida que o valor do *TickerBehaviour* aumenta.

5.1.2 Segunda simulação

Na segunda simulação será comparado o comportamento do sistema usando dois *RAs*. É possível verificar este comportamento na tabela 7.

2 Resources (R1 e R2)		
	***Edge* (ms)**	***Cloud* (ms)**
100 ms	R1=6859 e R2=6925	R1=180 e R2=177
500 ms	R1=7278 e R2=7243	R1=192 e R2=193
1000 ms	R1=7421 e R2=7406	R1=209 e R2=213

Tabela 7 - Comportamento do Sistema com 2 RA

Tal como na primeira simulação, tanto no método *Edge* como no método *Cloud* existe uma diminuição deste tempo à medida que o valor do *TickerBehaviour* diminui. As mesmas explicações que foram usadas para a primeira simulação em relação a estes valores também são validas para esta. Na figura 31 é possível verificar o tempo de execução do sistema com dois *RAs*.

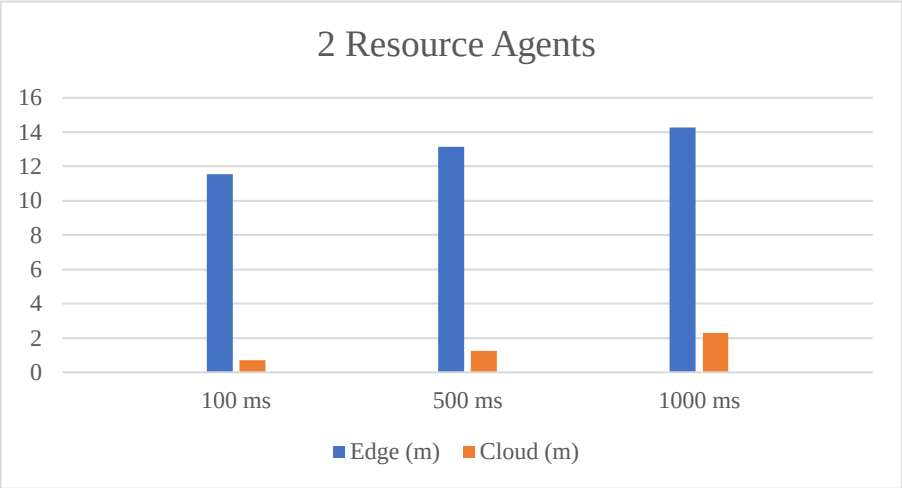

Figura 32 - Tempo de execução do sistema com 2 RA

Tal como na primeira simulação o tempo de execução é muito menor no método *Cloud*, do que no método *Edge*. Como se pode observar na figura anterior o tempo de execução vai aumentando significativamente com o aumento dos valores do *TickerBehaviour*.

5.1.3 Terceira Simulação

Na terceira simulação será comparado o comportamento do sistema usando três *RAs*. É possível verificar este comportamento na tabela 8.

3 Resources (R1, R2 e R3)		
	***Edge* (ms)**	***Cloud* (ms)**
100 ms	_	R1=209 e R2=208 e R3=206
500 ms	R1=8912 e R2=8823 e R3=8763	R1=173 e R2=175 e R3=169
1000 ms	R1=9422 e R2=9474 e R3=9351	R1=216 e R2=223 e R3=220

Tabela 8 - Comportamento do sistema com 3 RA

É possível observar na tabela 8, que tal como nas duas simulações anteriores existe uma diminuição do tempo que demora a prever cada linha à medida que o valor do *Ticker* diminui no método *Edge*. Não existe tempo quando o *Ticker* tem o valor de 100 ms, devido a limitações do ***Raspberry pi***.

Esta diminuição não acontece no método *Cloud* em comparação com as duas simulações anteriores. Neste método consegue-se verificar a partir da tabela anterior, que o menor tempo é quando o *Ticker* é de 500 ms. Por outro lado, o maior tempo é quando o *Ticker* toma o valor de 1000 ms, como seria de esperar. Estas diferenças serão explicadas na próxima secção, assim como as restrições encontradas ao longo deste estudo.

De seguida, na figura 32 consegue-se obter conhecimento sobre os tempos de execução desta simulação.

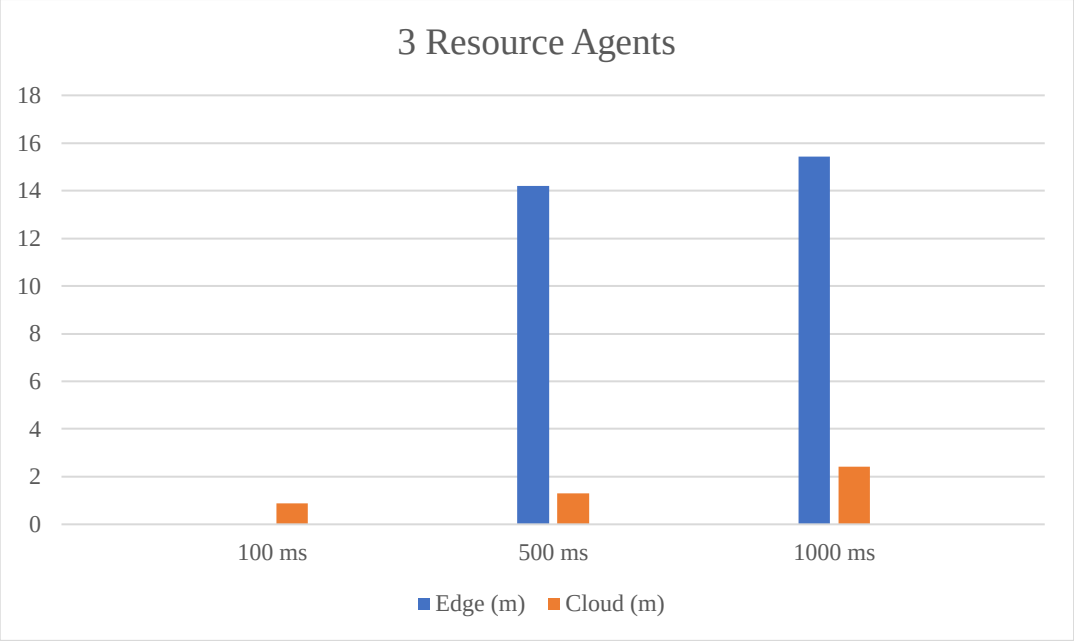

Figura 33 - Tempo de execução do sistema com 3 RA

Do mesmo modo que nas simulações anteriores o tempo de execução é bastante menor no método *Cloud* do que no método *Edge* e tal será assim na próxima simulação. Como referido anteriormente e como se pode observar na figura acima, a execução falhou quando o *Ticker* é 100 ms, portanto não foi calculado o tempo de execução para este valor.

5.1.4 Quarta simulação

Na quarta simulação será comparado o comportamento do sistema usando quatro *RAs.* É possível verificar este comportamento na tabela 9.

4 Resources (R1, R2, R3 e R4)		
	***Edge* (ms)**	***Cloud* (ms)**
100 ms	_	R1=240 e R2=243 e R3= 241 e R4=239
500 ms	_	R1=188 e R2=185 e R3= 183 e R4=183
1000 ms	_	R1=256 e R2=248 e R3=253 e R4=257

Tabela 9 - Comportamento do sistema com 4 RA

Devido às mesmas limitações que aconteceram na simulação anterior, não existe valores quando usado o método *Edge*, como se verifica a partir da tabela 9. Tal como na terceira simulação, no método *Cloud*, com 1000 ms de *TickerBehaviour* é quando se apresenta o maior tempo para prever uma linha, assim como o menor tempo é com 500 ms de *TickerBehaviour*.

Na figura 33, é possível observar os tempos de execução desta simulação.

Figura 34 - Tempo de execução do sistema com 4 RA

Como se verifica na figura anterior os únicos tempos de execução ocorrem no método *Cloud*. Como acontece nas simulações anteriores, este tempo vai aumentando com o aumento do valor do *TickerBehaviour*.

5.2 Discussão de Resultados

Nesta subsecção serão discutidos os resultados anteriormente enunciados mais detalhadamente, assim como as limitações encontradas ao longo deste estudo. De uma forma geral, a partir desta subsecção vai ser possível perceber qual o melhor método e porquê.

Na primeira simulação foi possível perceber que o as limitações do dispositivo usado já se começaram a sentir, mas ainda de uma forma ligeira comparada com as últimas duas simulações. Nesta simulação, onde foi testado unicamente um *RA* em execução, é a que se encontra mais idêntica com as utilizadas nos dias de hoje, onde cada dispositivo corre apenas um agente. Sabendo disto, com a realização destas simulações, quis-se verificar até que certo ponto era possível usar vários agentes a correr em simultâneo num dispositivo.

Convém referir que os resultados anteriormente descritos na secção anterior já eram esperados. Em todas as simulações o método *Edge* foi o método que demorou mais tempo a prever devido às limitações do *Raspberry pi*, que serão descritas na próxima subsecção. Neste método, os tempos que demoram a prever cada linha dos dados ficam à volta dos 5600 ms para um *RA*, à volta de 7000 ms para dois *RAs* e vai deixando de ser possível calcular este tempo à medida que vamos adicionando mais *RAs* em execução, como se pode verificar na terceira e quarta simulação.

O método *Cloud*, pelo contrário foi o que melhor rendimento obteve. Em todas as simulações foi possível verificar que obteve valores baixos. Com um *RA*, o tempo que demora a prever cada linha fica à volta dos 160 ms, passando para o valor de 180 ms com 2 *RAs*, para 190 ms com 3 *RAs* e para 205 ms com 4 *RAs*. Como se pode verificar, ainda era possível juntar mais agentes, de modo a continuar a avaliar este método, mas tal não era importante, pois o objetivo era comparar os dois métodos e com quatro simulações já é possível ter uma boa base de comparação.

A partir da terceira simulação, verificou-se que no método *Cloud*, este não seguia o mesmo padrão que nas simulações anteriores. Nas últimas, o tempo diminui à medida que o valor do *TickerBehaviour* diminui, mas neste caso tal não acontecia. Na terceira e quarta simulação o menor tempo é quando o *Ticker* toma o valor de 500 ms e não de 100 ms como acontecia na primeira e segunda simulação. O maior tempo continua a ser quando é 1000 ms, pois as tarefas são executadas de 1 em 1s, demorando mais tempo a prever cada linha. Como se pode verificar, quando o *Ticker* é de 1000 ms, como de 100 ms os tempos são próximos. Isto acontece, porque no caso de 100 ms, as tarefas são executadas tão rápido, num intervalo de 100 em 100 ms, que não dá tempo para prever em cada tempo de execução, demorando mais tempo a prever cada linha, devido ao número de *RAs*, como às limitações do dispositivo. No caso de ser 1000 ms, tal é devido à janela de tempo entre execuções ser de 1 em 1s, o que é um valor alto para prever este tipo de dados. Quando toma o valor de 500 ms, a periodicidade em que estas tarefas são executadas permite obter o menor tempo. Tal tempo, por ser um tempo intermédio entre os outros dois tempos possibilitou alcançar este tempo.

5.2.1 Limitações

Neste estudo existiram algumas limitações, estas devido ao uso do *Raspberry pi*. Numa primeira instância quando se começou a fazer as simulações no Raspberry pi para o método *Cloud*, no momento em que se iniciava o *server* este dava uma exceção "***ValueError: Buffer dtype mismatch, expected 'SIZE_t' but got 'long long***". Tal exceção era devido ao facto de ter treinado o meu modelo numa máquina com 64 bit *Python* e estar a correr numa máquina 32 bit *Python*. De modo a resolver este problema, o modelo foi treinado na mesma máquina onde irá correr, ou seja, no *Raspberry pi*.

Para além do método *Cloud*, também foram encontradas algumas limitações no método *Edge*. Como se verificou nas tabelas e figuras apresentadas na seção anterior, quando se aumentou o número de agentes a correr simultaneamente não foram apresentados alguns resultados. Isto deve-se ao facto de aparecer a exceção "***java.lang.OutOfMemoryError: Java heap space***" que termina a execução destes agentes. De modo a tentar corrigir este erro, foi aumentado o tamanho do *heap*, mas voltou-se a suceder o mesmo erro. Como este é um erro que indica que o tamanho do *heap* é insuficiente para a aplicação, pode-se concluir que é devido ao uso do Raspberry pi por ser um dispositivo com limitações devido aos seus aspetos técnicos.

5.2.2 Método de Eleição

De seguida será explicado, tendo em conta os resultados previamente descritos qual é o melhor método.

Por um lado, temos, o método *Edge* que apesar de apresentar algumas limitações devido ao uso do dispositivo utilizado, pode ser um método bastante interessante de utilizar. Este método como processa os dados perto de onde foram recolhidos pode ter algumas vantagens associadas, como a segurança destes dados e um processamento mais eficiente. Como se verificou a partir dos resultados que se obteve este método esteve muito longe de ter o rendimento que se verificou no outro método. Acredita-se que usando outro tipo de dispositivo, este seja capaz de diminuir a diferença do tempo entre os dois métodos.

Por outro lado, temos o método *Cloud*. Este método devido às restrições identificadas, obteve de longe o melhor rendimento. À medida que se ia aumentando o número de agentes foram obtidos também bons tempos. O máximo de agentes a correr em simultâneo neste método foi de 4 agentes, mas como foi possível perceber a partir dos resultados apresentados neste capítulo podiam-se adicionar mais.

Para concluir, apesar de com o método *Cloud* surgir algumas desvantagens como, limitações da largura de banda e segurança, este apresenta melhores resultados para realizar processamento de dados, se for usado um dispositivo como o *Raspberry pi* para correr os agentes. Fica então à escolha do fabricante escolher qual método deseja.

6 Conclusão e Trabalho Futuro

6.1 Conclusão

Com o trabalho desenvolvido ficou assim comprovado que a arquitetura multiagente apresentada neste documento provou ser uma solução eficaz para recolher e processar dados gerados a partir de medições de motores/equipamentos, criando novas maneiras de otimizar estes equipamentos/máquinas e melhorar o uso do usuário num sistema de produção, respondendo assim à questão colocada no subcapítulo 1.2.

Os resultados obtidos neste trabalho foram expectáveis, mostrando ser possível uma colaboração multiagente, de modo a melhorar o desempenho num sistema de produção. Esta solução aborda um contexto cada vez mais presente nos sistemas de produção.

Apesar de todos os testes terem sido realizados em ambiente simulado, foram realizados em número suficiente, de modo a ver o sistema reagir às alterações. Os testes realizados comprovam que o sistema tem resultados satisfatórios no método *Edge* com um *RA* e vai perdendo a sua eficiência à medida que são postos mais RAs a correr em simultâneo. No método *Cloud*, os resultados demonstram que o sistema tem melhor desempenho com um e vários *RAs* a correr em simultâneo em comparação com o método anterior.

Para concluir, apesar de serem expostas duas abordagens que estão presentes hoje em dia nos sistemas de produção industrial e seus comportamentos em contextos iguais, como dito anteriormente, esta escolha recai no fabricante. Num sistema de produção, onde o tempo e custos são extremamente importantes, a diferença entre analisar dados em tempo real mais rápidos e logo de seguida atuar sobre estes pode ser um fator chave.

6.2 Trabalho Futuro

Como trabalhos futuros, estes devem refletir-se principalmente no melhoramento das limitações apresentadas neste estudo, de maneira a tornar esta arquitetura mais eficiente e capaz de analisar dados mais complexos mais rapidamente.

Para saber onde têm melhor rendimento, devem ser escolhidos intervalos mais curtos de *TickerBehaviour*. Pode-se ter em conta o uso de um dispositivo com mais capacidades de processamento do que o *Raspberry pi*, testando assim o comportamento do sistema com vários agentes em execução em simultâneo.

Num mundo em constante evolução, a combinação destas duas abordagens vai ter um papel fundamental. Usando *Hybrid Cloud*, é possível obter benefícios que ao se usar uma só não se vai obter. Desta maneira, era possível ter duas arquiteturas que se complementam para garantir os melhores resultados para as empresas e seus clientes.

7 Referências

Antzoulatos, N., Castro, E., de Silva, L., Rocha, A. D., Ratchev, S., & Barata, J. (2016). A multi-agent framework for capability-based reconfiguration of industrial assembly systems. *International Journal of Production Research*, *55*(10), 2950–2960. https://doi.org/10.1080/00207543.2016.1243268

ATS. (2006). Downtime costs auto industry $22k/minute. Retrieved from https://news.thomasnet.com/companystory/downtime-costs-auto-industry22k-minute-survey-481017

Barata, J., Camarinha-Matos, L., & Cândido, G. (2008). A multiagent-based control system applied to an educational shop floor, *24*, 597–605. https://doi.org/10.1016/j.rcim.2007.09.008

Bellifemine, F., Caire, G., & Greenwood, D. (2007). Developing Multi-agent Systems with JADE. *Developing Multi-Agent Systems with JADE*, 1–286. https://doi.org/10.1002/9780470058411

Bensrhir, A. (2017). Deploying Machine Learning Models to Production. Retrieved from https://www.slideshare.net/AnassBensrhirDatasci/deploying-machine-learning-models-to-production

Bousdekis, A., Magoutas, B., Apostolou, D., & Mentzas, G. (2015). A proactive decision making framework for condition-based maintenance. *Industrial Management & Data Systems, 115(7),*1225-1250.

Bukkapatnam, S. T. S., Afrin, K., Dave, D., & Kumara, S. R. T. (2019). CIRP Annals - Manufacturing Technology Machine learning and AI for long-term fault prognosis in complex manufacturing systems. *CIRP Annals - Manufacturing Technology*, *68*(1), 459–462. https://doi.org/10.1016/j.cirp.2019.04.104

Carrasco, A., Hern, M. D., Romero-ternero, C., Sivianes, F., Oviedo, D., & Escudero, I. (2014). PeMMAS: A Tool for Studying the Performance of Multiagent Systems Developed in JADE, *44*(2), 180–189.

Cavalcante, I. M., Frazzon, E. M., Forcellini, F. A., & Ivanov, D. (2019). International Journal of Information Management A supervised machine learning approach to data-driven simulation

of resilient supplier selection in digital manufacturing. *International Journal of Information Management*, *49*(February), 86–97. https://doi.org/10.1016/j.ijinfomgt.2019.03.004

Cotteleer, M. M., & Sniderman, B. (2017). Forces of change: Industry 4.0. *Deloitte Insights*. https://doi.org/10.1007/s11947-009-0181-3

Drath, R., & Horch, A. (2017). Industrie 4.0: Hit or Hype?, *8*(June 2014), 0–6. https://doi.org/10.1109/MIE.2014.2312079

FIPA (2002). FIPA Contract Net Interaction Protocol Specification. Retrieved from: http://www.fipa.org/specs/fipa00029/SC00029H.html

FIPA. (2002). FIPA Request Interaction Protocol Specification. Retrieved from http://www.fipa.org/specs/fipa00026/SC00026H.html

Guo, Q. lin, & Zhang, M. (2010). An agent-oriented approach to resolve scheduling optimization in intelligent manufacturing. *Robotics and Computer-Integrated Manufacturing*, *26*(1), 39–45. https://doi.org/10.1016/j.rcim.2009.02.003

Harmouch, F. Z., Krami, N., & Hmina, N. (2018). A multiagent based decentralized energy management system for power exchange minimization in microgrid cluster. *Sustainable Cities and Society*, *40*(April), 416–427. https://doi.org/10.1016/j.scs.2018.04.001

Hermann, M., Pentek, T., & Otto, B. (2015). Design Principles for Industrie 4.0 Scenarios: A Literature Review. *Journal of Colloid And Interface Science*, *514*(01), 172–181. https://doi.org/10.1016/j.jcis.2017.12.027

Hu, L., Miao, Y., Wu, G., Mehedi, M., & Humar, I. (2019). iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing. *Future Generation Computer Systems*, *90*, 569–577. https://doi.org/10.1016/j.future.2018.08.006

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008) "Random Survival Forests. Annals of Applied Statistics", 2(3), 841–860.

Kagermann, H., Wahlster, W., & Helbig, J. (2013). Recommendations for implementing the strategic initiative INDUSTRIE 4.0, (April).

Pouspourika, K. (2019). The 4 Industrial Revolutions. Retrieved from https://ied.eu/project-updates/the-4-industrial-revolutions/

Khan, M. W., Wang, J., Ma, M., Xiong, L., Li, P., & Wu, F. (2019). Optimal energy management and control aspects of distributed microgrid using multi-agent systems. *Sustainable Cities and Society*, *44*(November), 855–870. https://doi.org/10.1016/j.scs.2018.11.009

Lechevalier, D., Narayanan, A., & Rachuri, S. (2014). Towards a Domain-Specific Framework for Predictive Analytics in Manufacturing Towards a Domain-Specific Framework for

Predictive Analytics in Manufacturing, (October). https://doi.org/10.1109/BigData.2014.7004332

Lee, J., Bagheri, B., & Kao, H. A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

Macgregor, J., & Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. *Computers and Chemical Engineering*, *47*, 111–120. https://doi.org/10.1016/j.compchemeng.2012.06.017

Manyika, James, et al. (2011). "Big data: The next frontier for innovation, competition and productivity." Technical report, McKinsey Global Institute

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.

Ng, K. K. H., Cao, Y., & Lee, C. K. M. (2017). Big Data Analytics for Predictive Maintenance Strategies. *Supply Chain Management in the Big Data Era*, *15*(2), 161-164. https://doi.org/10.4018/978-1-5225-0956-1.ch004

Oliveira, E., Fischer, K., & Stepankova, O. (1999). Autonomous Systems Multi-agent systems: which research for which applications, *27*, 91–106.

Oliveira, J. A. B. de. (2003). Coalition based approach for shop floor agility – a multiagent approach. Retrieved from http://hdl.handle.net/10362/2483

Peres, R. S., Rocha, A. D., Leitão, P., & Barata, J. (2018). IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0. *Computers in Industry*, *101*(June), 138–146. https://doi.org/10.1016/j.compind.2018.07.004

Radetzky, M., Rosebrock, C., & Bracke, S. (2019). to process Approach to process Approach to on Approach to adapt. *IFAC PapersOnLine*, *52*(13), 1773–1778. https://doi.org/10.1016/j.ifacol.2019.11.458

Red Hat. (2020). Edge Computing. Retrieved from https://www.redhat.com/pt-br/topics/edge-computing/what-is-edge-computing

Ribeiro, L. (2017). Cyber-physical production systems' design challenges. *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 1189-1194.

Ribeiro, L., & Barata, J. (2011). Computers in Industry Re-thinking diagnosis for future automation systems: An analysis of current diagnostic practices and their applicability in emerging IT based production paradigms. *Computers in Industry*, *62*(7), 639–659. https://doi.org/10.1016/j.compind.2011.03.001

Ribeiro, M., Grolinger, K., & Capretz, M. A. M. (2015). MLaaS: Machine Learning as a Service. *IEEE International 71 Conference on Machine Learning and Applications*, (c).

Sabar, M., Montreuil, B., & Frayret, J. M. (2009). A multi-agent-based approach for personnel scheduling in assembly centers. *Engineering Applications of Artificial Intelligence*, *22*(7), 1080–1088. https://doi.org/10.1016/j.engappai.2009.02.009

Schäfer, C. (2007) 'On the Modularity of Manufacturing systems', in IEEE Industrial Electronics Magazine, 1(3), 20–27.

Shang, C., & You, F. (2019). Data Analytics and Machine Learning for Smart Process Manufacturing: Recent Advances and Perspectives in the Big Data Era. *Engineering*, *5*(6), 1010–1016. https://doi.org/10.1016/j.eng.2019.01.019

Shin, S., Woo, J., & Rachuri, S. (2014). Predictive analytics model for power consumption in manufacturing. *Procedia CIRP*, *15*, 153–158. https://doi.org/10.1016/j.procir.2014.06.036

Wang, K. (2016). Intelligent Predictive Maintenance (IPdM) system – Industry 4.0 scenario. *WIT Transactions on Engineering Sciences*, *113*, 259–268. https://doi.org/10.2495/IWAMA150301

Wooldridge, M. (2002). An Introduction to Multi-Agent Systems. Wiley, Chichester, England.

Yan, Z., & Wang, J. (2014). "Robust Model Predictive Control of Nonlinear Systems With Unmodeled Dynamics and Bounded Uncertainties Based on Neural Networks," in *IEEE Transactions on Neural Networks and Learning Systems*, 25(3), 457-469.

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud Computing: State-of-the-art and Research Challenges Cloud computing: state-of-the-art and research challenges, (May 2014), 6–18. https://doi.org/10.1007/s13174-010-0007-6

Zhao, Z., Lin, P., Shen, L., Zhang, M., & Huang, G. Q. (2020). Advanced Engineering Informatics IoT edge computing-enabled collaborative tracking system for manufacturing resources in industrial park. *Advanced Engineering Informatics*, *43*(January), 101044. https://doi.org/10.1016/j.aei.2020.101044

Zhou, X., Xi, L., & Lee, J. (2007). Reliability-centered predictive maintenance scheduing for a continuously monitored system subject to degradation. *Reliability Engineering and System Safety*, *92*(4), 530–534. https://doi.org/10.1016/j.ress.2006.01.006