escola superior de tecnologia e gestão
instituto politécnico de leiria

Integração e Controlo de Produção num Sistema I4.0

Mestrado em Engenharia Mecânica – Produção Industrial

Dissertação

Kevin Cândido Ferreira

Leiria, *março de 2025*

escola superior de tecnologia e gestão
instituto politécnico de leiria

Integração e Controlo de Produção num Sistema I4.0

Mestrado em Engenharia Mecânica – Produção Industrial

Dissertação

Kevin Cândido Ferreira

Dissertação de Mestrado realizada sob a orientação do Doutor Carlos Neves, Professor Coordenador da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e do Doutor Marcelo Gaspar, Professor Adjunto da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, março de 2025

Originalidade e Direitos de Autor

A presente dissertação é original, elaborada unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para a elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual a mesma foi realizada, a saber, Curso de Mestrado em Engenharia Mecânica – Produção Industrial, no ano letivo 2024/2025, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Agradecimentos

Começo por agradecer ao Instituto Politécnico de Leiria pelas condições disponibilizadas para que concluísse este ciclo da minha vida académica com sucesso.

Um agradecimento para os meus orientadores Carlos Neves, Professor Coordenador da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, e Marcelo Gaspar, Professor Adjunto da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, pela oportunidade e o apoio prestados ao longo de todo este processo.

Um agradecimento especial ao autor da ferramenta, Aljosha Köcher, pelo apoio e esclarecimentos quanto ao funcionamento do *SkillMex*, e ao David Pereira, por todo o apoio disponibilizado no LRAFI.

Agradeço à minha família e amigos por todo o suporte e compreensão ao longo dos períodos difíceis.

Resumo

O presente relatório visa documentar a implementação de um ciclo produtivo experimental definido através de um sistema de gestão da produção MES (M*anufacturing Execution Systems*). Numa primeira etapa do trabalho foram aprofundados conceitos relativos ao sistema de gestão da produção em estudo, focando particularmente no MES Semântico, nomeadamente no conhecimento que envolve semanticamente a descrição e integração de toda a informação do produto, dos seus processos produtivos e de todos os recursos envolventes. Neste contexto foram analisados e discutidos os conceitos de módulo, capacidade e *skill*, assim como o conceito de integração da informação recorrendo a um protocolo agnóstico, no caso o OPC UA, uma norma para a integração e troca de informação com crescente utilização em contexto industrial.

Os estudos dos conceitos anteriores convergiram na adoção de uma metodologia de implementação específica, o que inclui a seleção de um sistema MES, neste caso, um do tipo *open-source*, assim como o respetivo estudo e demais ferramentas para o efeito. Neste estudo, o procedimento produtivo foi organizado num diagrama BPMN (*Business Process Model and Notation*) e foram produzidos módulos referentes a cada equipamento do ciclo e competências (*skills*) referentes a cada funcionalidade. Todo este procedimento foi iterado no Laboratório de Robótica Avançada e Fábricas inteligentes (LRAFI) da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Palavras-chave: *MES*, *skill*, módulo, OPC UA

Abstract

This document reports the implementation of an experimental production cycle defined through a Manufacturing Execution System (MES). In the first stage of this work, concepts related to production management systems will be presented, focusing on the MES that was the target of this project, namely the Semantic MES, which refers to the knowledge that semantically involves the description and integration of all product information, its production processes, and all surrounding resources. Additionaliy, the concepts of module, capacity, and skill were analysed and discussed, l as well as the concept of OPC UA, a standard for the integration and exchange of information with increasing adoption in industrial context.

The study of the previous concepts led to the adoption of a specific implementation methodology, which includes the selection of a MES system, in this case, an open-source software, to study it and other tools for this purpose. In this study, the production process was organized in a BPMN (Business Process Model and Notation) diagram, in which dedicated modules were produced for each equipment in the cycle, including skills for each functionality. This entire procedure was carried out at the Advanced Robotics Laboratory and Smart Factories (LRAFI) of the School of Technology and Management of the Leiria Polytechnic Institute.

Keywords: MES, skill, module, OPC UA

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

- AHP - *Analytic Hierarchy Process*
- BOM – *Bill of Materials*
- BPMN - *Business Process Model and Notation*
- CPPS - *Cyber-Physical Production Systems*
- CPS – *Cyber-Physical Systems*
- ECO - *Engineering Change Orders*
- ERD - *Entity-Relationship Diagrams*
- ERP – Sistemas de Gestão Integrada
- HMI - *Human-Machine Interface*
- IDEFO - *Integration Definition for Process Modelling*
- IIoT – *Industrial Internet of Things*
- IoT – *Internet of Things*
- IT – Tecnologias de Informação
- JIT – *Just-In-Time*
- KPI - *Key Performance Indicator*
- LM – *Lean Manufacturing*
- LTS - *Long-term support*
- MBSE - *Model-Based Systems Engineering*
- MES – *Manufacturing Execution System*
- MOM – *Manufacturing Operations Management*
- npm - *Node Package Manager*
- OEE - *Overall Equipment Effectiveness*
- OPC UA - *Open Platform Communications Unified Architecture*
- OT – Tecnologias Operativas
- PDCA - *Plan-Do-Check-Act*
- PLM – *Product lifecycle Management*
- RFID – *Radio Frequency IDentification*
- RFP – *Request for proposal*
- SCADA - *Supervisory Control and Data Acquisition*
- SQL - *Structured Query Language*

- SMED - *Single Minute Exchange of Die*
- SOW - *Statement of Work*
- SPC - *Statistical Process Control*
- TPM - *Total Productive Maintenance*
- TPS – *Toyota Production Systems*
- TQM - *Total Quality Management*
- UI – Unidade Industrial
- WIP – *Work in Progress*
- WS - *Weihenstephan Standards*

1. Introdução

1.1. Enquadramento

No atual contexto industrial, a globalização tem vindo a provocar uma revolução sem precedentes, permitindo uma maior abertura da economia mundial e incentivando um efetivo incremento da competitividade das empresas e organizações. Como consequência, as leis do mercado conduziram à necessidade de um investimento significativo em inovação, criatividade e aptidão para a implementação de soluções na gestão da produção que permitam às empresas e organizações serem cada vez mais eficientes, sustentáveis e competitivas.

A incremental evolução no âmbito das ferramentas tecnológicas e industriais têm permitido um significativo crescimento da produtividade e eficiência das organizações, o que tem levado a uma cada vez maior capacidade de satisfação das necessidades do cliente sob a forma de produtos e/ou serviços finais de qualidade superior. Neste contexto, pode destacar-se o recurso às tecnologias de informação (*IT*) que têm permitido uma crescente digitalização de equipamentos e processos industriais. Cumulativamente, as tecnologias operativas (*OT*) têm permitido materializar o potencial das ferramentas *IT* no incremento da produtividade e eficiência na indústria. De modo particular, estas tecnologias têm vindo a convergir com o conceito de indústria 4.0, cuja integração visa otimizar todo o processo produtivo das organizações.

No contexto nacional, as consecutivas crises económicas e financeiras têm-se tornado em desafios e oportunidades que levaram as organizações portuguesas a abrir caminhos para uma maior internacionalização. Como consequência, o crescimento das exportações levou a uma maior exposição das empresas nacionais a padrões de competitividade e exigência por parte do cliente final, o que levou a que estas empresas se tivessem de adaptar a esta realidade por forma a sobreviver e a crescer neste contexto. É de destacar que um dos fatores que mais pesa nas empresas e organizações é o incremento dos custos internos, nomeadamente, os custos com energia, matérias-primas e/ou produtos e serviços subcontratados. Deste modo, torna-se indispensável que as organizações tenham um efetivo conhecimento e controlo das operações ao longo de toda a sua cadeia de abastecimento e que permita uma maior eficiência

na gestão dos processos de produção. Com vista a contribuir para essa maior eficiência de processos, encontra-se disponível um cada vez maior leque de sistemas de informação que permitem dar resposta a esta digitalização na gestão, como é o caso dos sistemas de gestão integrada (*ERP*). Assim estes sistemas permitem que as empresas possam controlar e interligar as diferentes áreas ou departamentos do negócio, do fabrico e produção. Por seu turno, os sistemas automáticos disponíveis no âmbito dos processos industriais das empresas, podem igualmente ser controlados com vista a contribuir para um incremento da eficiência e da produtividade destas organizações. Assim, a integração destes dois níveis na maior parte das empresas e organizações permite torná-las cada vez mais competitivas e preparadas para os desafios de produtividade cada vez mais incontornáveis no atual contexto de globalização dos mercados

Na indústria nacional e internacional, a racionalização e controlo de toda a atividade fabril constitui uma vantagem competitiva, na medida em que se uma organização consegue implementar, para um mesmo produto, um preço mais reduzido, comparativamente com a concorrência, consegue obter maiores margens de lucro. O âmbito da presente dissertação, intitulada 'Integração e Controlo de Produção num sistema I4.0', assume uma conexão irreversível num contexto industrial com o conceito do Sistema de Execução da Produção, comumente denominado por *MES,* que visa precisamente dar resposta a esta necessidade de racionalização e controlo de processos.

1.2. Objetivos

A presente dissertação visa à concretização de objetivos específicos relativos à temática da gestão da produção, nomeadamente, o estabelecimento da ponte entre os sistemas de gestão de topo, como os *ERP*, e sistemas de automação de chão de fábrica, de modo particular os sistemas *MES*. A determinação da viabilidade de aplicação deste tipo de sistemas de informação será um tópico forte na presente prática, especialmente por via de metodologias com base em ontologias, módulos e *skills*. Neste sentido, foi selecionada uma ferramenta para o efeito, em que se modelou um ciclo produtivo experimental, ou seja, uma sequência de processos que será aplicada a um determinado produto a implementar no Laboratório de Robótica Avançada e Fábricas Inteligentes (LRAFI) da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria. Assim, no âmbito deste trabalho pretende-se analisar e discutir a sua aplicação no âmbito referido e destacar os resultados

que permitam a sua implementação real na indústria, assim como identificar potenciais limitações que possam surgir desta execução.

1.3. Estrutura

No âmbito da atual dissertação, para além desta secção introdutória, apresenta-se no Capítulo 2 o estado da arte, em que são explorados os temas diretamente ou indiretamente relacionados com os sistemas *MES*, visando fomentar e estabelecer o estado atual do conhecimento sobre estes elementos, ou seja, permite verificar o que os especialistas e investigadores concluíram e visionam acerca destes conceitos. No Capítulo 3, procede-se à caracterização do ambiente produtivo, assim como dos equipamentos a utilizar no LRAFI, procedendo-se posteriormente à definição de um ciclo produtivo a aplicar num determinado produto, em que será ainda apresentada uma proposta de solução baseada no estado da arte realizado, nas ferramentas disponíveis (*open-source*, implementações livres ou disponibilizadas por parceiros) e na fábrica inteligente do LRAFI, em que se procederá à definição e execução de um diagrama *BPMN* (*Business Process Model and Notation*). No Capítulo 4, o foco centra-se nos resultados de aplicação obtidos e na apresentação de soluções para os problemas que poderão decorrer do processo. Por último, no Capítulo 5, será produzida uma análise crítica da ferramenta e que resultará num conjunto de conclusões quanto à sua aptidão e eficiência analítica, em que serão propostas melhorias de resolução e indicações para trabalhos futuros.

2. Estado da Arte

2.1. Sistemas de Informação na Indústria

Na sequência dos objetivos descritos na secção 1.2 torna-se relevante discutir a importância dos sistemas de informação aplicados à gestão da produção. No contexto industrial das empresas, existem diversos elementos a analisar, medir e estudar com vista a potenciar o incremento da produtividade e a minimização de custos, nomeadamente no âmbito da eficiência dos processos nos seus diversos departamentos, como é o caso da receção, gestão e transformação de matérias-primas, a troca de informação e comunicação entre os vários departamentos e a gestão de stocks. Assim, o controlo destes elementos traduz-se em informação, cuja análise permite a resolução de problemas, a melhoria de processos e o incremento da qualidade dos produtos finais. É de referir que não pode existir gestão efetiva sem informação numa empresa [1].

O desafio da informação na indústria não é um problema apenas surgido na atual sociedade do conhecimento. Aliás, o primeiro tratado de gestão remete para o ano de 1494, com o lançamento da obra[1] de Luca Pacioli em Veneza, que contém um capítulo sobre contabilidade. Torna-se assim necessário realizar a distinção entre um sistema de informação e um sistema informático. O último permite a resolução de problemas de memorização, recolha, processamento ou cálculo, além de permitir a consulta da informação, sendo inevitável que um sistema de informação necessite desta integração. A informação é todo o elemento essencial à atividade da empresa, que influencia as tomadas de decisão, ou seja, um sistema de informação integra elementos físicos, humanos, lógicos e financeiros de uma organização, que interagem entre si, promovendo o cumprimento de regras de combinação produtivas que visam concretizar um objetivo produtivo. Em síntese, todos os sistemas de gestão prossupõem a sua associação a um modelo de sistemas de informação.

A teoria da informação pressupõe ainda que o responsável da produção necessita de informação a montante, como os prazos de entrega, os abastecimentos, os stocks e as limitações financeiras, como a jusante, os clientes, as ordens de encomenda e a disponibilidade da procura. Por outro lado, as empresas devem analisar dados relativos a recursos humanos, de direção comercial e financeira. A integração de toda esta informação

[1] A obra lançada por Luca Pacioli, 'Summa de arithmetica, geometria, proportioni et proportionalita'[72]

culmina numa melhor tomada de decisões, devendo ter como referência a cadeia interna de valor acrescentado. Independentemente de uma empresa possuir produções em série ou por encomenda, existem sempre elementos de informação que são relevantes conhecer, quantidades de produção útil, defeituosos, tempos de produção de máquinas e de processos. Importa salientar que cada empresa se insere num mercado único, na medida em que as variáveis a medir na empresa i, podem não ser relevantes para a empresa j [1], [2].

2.2. Caracterização Geral do MES

Um sistema *MES*, do inglês *Manufacturing Execution System*, corresponde a um sistema de informação centralizado que visa o incremento da eficiência dos processos produtivos. O conceito de *MES* surgiu originalmente em 1990, sendo classificado inicialmente como sendo a camada intermédia de soluções de *software* entre a automação em chão de fábrica e os sistemas de planeamento e gestão empresarial, os *ERP*. Os *MES* permitem a execução, monitorização, análise e controlo de um conjunto pré-definido de tarefas implementadas em chão de fábrica, além de se integrarem com outros sistemas em toda a organização e com a cadeia de abastecimento [3]. Este sistema controla as metodologias, as variáveis relevantes e os materiais utilizados no processo produtivo. Por outro lado, permite também uma efetiva gestão das ordens de trabalho, indicando e fornecendo as ferramentas necessárias para a resolução de problemas. Estas visam ainda a otimização dos procedimentos através de análises em tempo real de dados referentes ao processo produtivo. O *MES* pode fornecer informação relevante e completa dos processos produtivos em tempo real, garantindo a qualidade dos produtos finais a um menor custo e em menos tempo. São ainda de destacar as vantagens que esta ferramenta fornece às organizações, podendo constituir uma das estratégias a implementar para fazer face à globalização e ao incremento da competitividade nos mercados [4].

De uma forma mais sucinta e dinâmica, os objetivos que podem vir a ser alcançados com a implementação do *MES* são os seguintes:

- Otimização e controlo do fluxo produtivo, por via de documentação detalhada em tempo real;

- Melhoria da acessibilidade e qualidade da informação relativa aos processos, materiais e produtos;
- Visibilidade e transparência durante todo o processo produtivo, cujos desvios ao normal funcionamento do fluxo de produção são devidamente comunicados e analisados;
- Minimização de custos relativos ao armazenamento de materiais associados ao *Work In Progress* (*WIP*), devido a tempos de entrega (*Lead Time*) inferiores;
- Minimização de trabalhos administrativos relativos à gestão e manutenção de documentos;
- Minimização de peças e lotes perdidos e defeituosos;
- Minimização de custos operacionais, devido ao elevado nível de integração;
- Melhoria dos processos de tomada de decisão, através de um acesso facilitado à informação produtiva para todos os casos de negócio mais críticos.

Por outro lado, existem limitações, nomeadamente a complexidade da integração deste com outros tipos de *software*, uma vez que a compatibilidade entre sistemas nem sempre existe. Algumas organizações, como a *Oracle* e a *SAP*, possuem esta perceção e tentam encontrar soluções para esta questão [5]. Os modos de estruturação, armazenamento e recuperação de dados diferem entre sistemas, originando, assim, uma discrepância entre soluções de *software*. Por outro lado, o *MES* é um sistema relativamente rígido, implicando problemas com a flexibilidade de produção. Porém, aquando exigido um elevado nível de flexibilidade numa organização, nomeadamente no chão de fábrica, o *MES* não é capaz de se adaptar facilmente a essa mudança de processo. Basicamente, o *MES* pode ser configurado para qualquer ambiente de fabricação, porém não possui uma alta velocidade de adaptação e mudança, em ambientes altamente flexíveis [6]. As fábricas mais modernas produzem uma grande quantidade de dados produtivos, em que o *MES* enfrenta desafios relativamente à gestão adequada desses dados, devendo possuir soluções analíticas para apoiar a sua recolha, armazenamento e análise. [3].

Comparativamente ao *ERP*, o *MES* assume-se como uma solução mais viável para a resolução de processos produtivos complexos, com múltiplas variações e uma elevada quantidade de transações, uma vez que o *ERP* é idealmente projetado para suportar um processo homogéneo com dados operacionais da organização. No entanto, ambos são críticos para efetuar uma eficaz gestão e planeamento dos processos produtivos, já que quando combinados com os sistemas automáticos de chão de fábrica, tornam expetável um

incremento da competitividade pela própria organização [7]. Por fim, é de destacar que, desde os anos 90 do século passado, os *MES* evoluíram significativamente para *softwares* mais poderosos e integrados com a evolução das *IT*. Mais recentemente, estes podem dar resposta à maioria dos processos de fabricação, desde a ordem de fabrico até à entrega do produto. Porém, não conseguem adicionar recursos de tomada de decisão em organizações mais dinâmicas, de acordo com as estratégias de reabastecimento da cadeia de abastecimento, e simultaneamente responder dinamicamente a mudanças imprevistas [7] [4].

2.3. *MES* Semântico

A incontornável transformação digital corresponde a um dos maiores desafios na produção industrial, nomeadamente na automatização dos processos internos, cujos objetivos incluem o estabelecimento do acesso à informação dos sistemas técnicos, à partilha de informação, no aumento da eficácia e eficiência do processamento da informação. Os sistemas técnicos correspondem a componentes e tecnologias que suportam a execução e controlo dos processos de produção [8]. A maior parte das organizações utilizam ficheiros do tipo *PDF* ou semelhantes, porém, apesar de serem representações de dados digitais, eles não providenciam um significado semântico dos dados ou uma visão integrada em múltiplas fontes de dados. Deste modo, o potencial de uma estratégia de transformação digital adequada não é devidamente concretizado.

No atual contexto de globalização dos mercados, as empresas e organizações encontram desafios particulares relativos aos seus custos de produção, flexibilidade dos processos e uma crescente exigência dos clientes finais. Este fenómeno tem levado a uma crescente procura por produtos cada vez mais customizados e produzidos em menores quantidades, que aquando efetuadas reconfigurações manuais determinam elevados custos relativos à customização. Assim, a tendência deve ser a automatização desses sistemas para incrementar a eficiência e tempos produtivos. Por outro lado, os processos produtivos devem ser acompanhados e controlados para verificar o cumprimento dos requisitos, documentando informações relevantes. Por norma, as organizações possuem informação relativa aos processos de produção, mas não possuem uma representação semântica dos mesmos [9]. Esta representação dos dados na indústria refere-se à forma como os dados são organizados para que possam ser compreendidos e utilizados de uma forma eficiente por sistemas e pessoas [10].

As ferramentas *PLM* (*Product lifecycle Management*) indicam descrições acerca dos diversos dados e ficheiros, mas não possuem uma compreensão semântica das suas conexões e implicações. Porém, existem propostas de conceito de engenharia digital baseadas no conhecimento que envolve semanticamente a descrição e integração de toda a informação do produto, dos seus processos produtivos e recursos [9]. As soluções de automação industrial são caracterizadas por um grande esforço de engenharia e de adaptação a componentes específicos de *hardware* e *software*. Além disso, o conhecimento de domínio e algum senso comum são frequentemente incorporados nas implementações de *software* que operam os sistemas. O conhecimento é implicitamente representado e oculto no código-fonte. Por isso, é recomendável a separação do conhecimento das implementações de software, descrevendo-o em ontologias utilizando uma linguagem formal de representação do conhecimento para torná-lo reutilizável e compartilhável. As ontologias são utilizadas para modelar e combinar o conhecimento de senso comum, domínio e aplicação para fornecer uma compreensão mais profunda da tarefa [11]. Com base na descrição semântica, as atividades de engenharia relacionadas à produção podem ser automatizadas.

Por outro lado, surge o conceito de *skill* ou, em português, competência[2], correspondente a uma abordagem centrada em ferramentas para a modelação e execução de processos que visa a simplificação das funcionalidades fornecidas pelos componentes de *hardware* e *software* [9]. Tendo em conta os princípios da representação explicita do conhecimento utilizando ontologias e a integração dos recursos produtivos, surge o conceito *OPC UA* (*Open Platform Communications Unified Architecture*) que corresponde a uma norma para a integração e troca de informação nos sistemas de produção modular, entre o chão de fábrica e os níveis de decisão estratégica das organizações, que ganhou força desde 2003 por tornar as comunicações *standard [12]*. Este foi desenvolvido pela fundação *OPC* e representa uma evolução do *OPC Classic* para formar um padrão de interoperabilidade independente da plataforma utilizada para permuta de dados [13].

Um modelo pode ser composto por diversas funcionalidades de alto nível que podem ser aplicadas no chão de fábrica. Nos últimos anos, verifica-se uma investigação ativa em relação à representação do conhecimento para a automação industrial e robótica, em que muitas das abordagens seguem o paradigma do produto-processo-recursos (PPR), em que estes devem ser caracterizados e modelados através de um software *CAD*. A combinação de

[2] Daqui em diante, decidi utilizar apenas o termo *skill* em vez de competência, para efeitos de coerência.

skills e a representação explícita de conhecimento é frequentemente utilizada para minimizar a complexidade da programação de sistemas produtivos. Uma abordagem popular referente às tarefas de montagem remete para uma arquitetura de sistema em que os constrangimentos geométricos sejam especificados pelo procedimento de montagem e sejam descritos nas *skills* aplicadas a uma determinada célula de trabalho. Posteriormente, estas *skills* devem ser implementadas e controladas por máquinas de estados finitos.

Por outro lado, existe uma importância crescente em implementar uma produção autónoma que represente uma transformação digital associada aos produtos e recursos produtivos. Para se obter uma produção autónoma e uma autogestão dos sistemas produtivos, é necessário ter acesso a dados e informações pertinentes e a uma representação formal que codifique o seu significado semântico [9]. A extensão da produção autónoma pode ocorrer com as anotações automatizadas de dados dos processos e através de indicadores-chave de desempenho, os *KPIs* (Figura 1). Portanto, é recomendado seguir um modelo de processo centrado no produto, em vez de nas ferramentas, que se baseiam unicamente na sequência de invocações de *skills* específicas, durante todo o seu processo de criação [14]. Os modelos de processos centrados no produto incluem descrições, independentes de *hardware*, das etapas de fabrico necessárias que levam à síntese de um determinado produto, onde são identificados componentes de *hardware* e *software* adequados com base numa descrição formal das suas capacidades.

A partilha de conhecimento é dificultada pelas várias linguagens de programação existentes. Pode ser utilizada uma linguagem para definir e instanciar ontologias na *World Wide Web*, a *OWL* (*Ontology Web Language*), que fornece métodos para descrever formalmente entidades como classes, instâncias dessas classes e propriedades que definem as relações entre elas ou as ligam com números ou *strings* [15]. Devido à sua ampla aplicabilidade e à disponibilidade de muitas ferramentas de *software*, estes conceitos podem ser usados noutras áreas, como os domínios da automação industrial e da robótica. Novas classes e relações complexas podem ser criadas combinando entidades existentes.

Um aspeto fundamental corresponde à utilização de uma representação comum para descrever diversos dados de fontes heterogéneas e diferentes tipos de conhecimento (Figura 1), o que inclui o conhecimento do produto a ser construído, o processo de produção que cria o produto, os recursos de fabricação que executam o processo e o conhecimento geral dos domínios de automação e aplicação. A integração semântica de determinados tipos de

informação pode ser automatizada, enquanto para outros ainda deve ser realizada manualmente. Como o esforço manual só precisa ser gasto uma vez e os modelos resultantes podem ser compartilhados e reutilizados, o esforço inicial pode levar a ganhos de eficiência a longo prazo. Na *OWL*, cada entidade possui um Identificador de Recursos Internacionalizado (*IRI*), que pode ser usado para referenciar a entidade de forma inequívoca e vinculá-la a outras entidades.

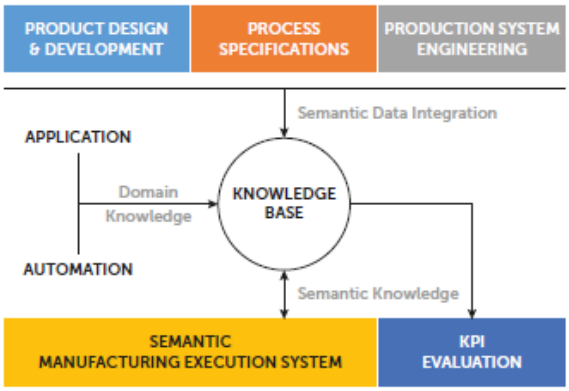

Figura 1.: Representações semânticas de vários tipos de conhecimentos integrados no *MES* semântico [9]

O componente *Knowledge Base* (*KB*) é responsável pelo armazenamento central, interpretação e acesso ao conhecimento semântico no sistema (Figura 1). Para isso, ele utiliza os recursos de armazenamento triplo, porque utiliza o formato sujeito, predicado e objeto [16], e de inferência *OWL 2 RL* da base de dados *Ontotext GraphDB*. O *KB* utiliza o servidor *OPC UA* de código aberto *Eclipse Milo*, para que outros componentes de *hardware* e *software* numa célula de trabalho possam comunicar diretamente com esta via *OPC UA*. Para cada caso de utilização, como por exemplo, para uma célula de trabalho de um robô para a montagem de fusíveis, tanto as ontologias *OWL* gerais quanto as especializadas emitem o domínio de aplicação, processo de produção e célula de trabalho, sendo carregadas no *KB*. Uma classe *OWL* pode ser vinculada a várias solicitações através da mesma propriedade. O sistema *MES* semântico (*sMES*) controla os componentes de *hardware* e *software* de uma determinada célula de trabalho, sendo responsável pela realização de um processo de trabalho. Este possui as funções de controlar e executar os processos produtivos. O *sMES* é gerido por um *MES* de chão de fábrica, que recolhe informações e aciona o *sMES* com o *ID* do produto correspondente.

Todos os componentes do dispositivo no sistema implementam um modelo genérico da *skill*, com estados e transições predefinidos. Assim que um dispositivo recém-conetado é detetado, o *sMES* aciona o seu detetor de *skills* interno para procurar no servidor remoto os

tipos de *skills* fornecidos. Assim, é constantemente mantida uma lista atualizada de *skills* disponíveis, que podem ser utilizadas no procedimento de representação de dados e processos. As *skills* mais simples podem ser agrupadas hierarquicamente para compor tipos de *skills* novas e mais complexas. Por exemplo, um tipo de *skill Pick-and-Place* é composto por *skills* de movimento do robô e da manipulação de garras. Esta composição hierárquica leva a uma maior abstração de funcionalidades de *hardware* e permite a modelação intuitiva de *skills* complexas [9].

2.4. Guias e Standards de implementação de um *MES*

Os *MES* seguem determinadas diretrizes *standard*, para o modelo e para as suas respetivas relações, definidas pela organização sem fins lucrativos *MESA (Manufacturing Execution System Association) International* [17], denominando-se a norma por *ANSI/ISA 95*. Estas normas determinam metodologias, formas de trabalhar, de pensar e de comunicar eficientemente, contendo modelos e terminologias bem definidos que podem ser utilizadas para analisar qualquer empresa no âmbito da produção industrial. A posição do *MES* na hierarquia da fábrica, bem como as suas funções e interfaces estão definidos e normalizados na *ISA 95* (Figura 2). Na cadeia de abastecimento, os *MES* contribuem para uma integração vertical, que agrega todos os sistemas disponíveis numa organização, incluindo o sistema *ERP* e os sistemas automáticos de chão de fábrica, que podem ser integrados num único sistema. Com a utilização da norma *ISA 95*, os esforços desenvolvidos pelas organizações serão minimizados. A norma fornece uma base para a distribuição de interfaces e de tarefas perante a implementação de um *MES*. Na norma, encontra-se definido um sistema hierárquico de cinco níveis dos sistemas de informação presentes numa organização, remetendo para conceitos que constam na Figura 2 [18], representando a logística empresarial, o *MOM* (*Manufacturing Operations Management*), o controlo da produção e as funções do processo produtivo.

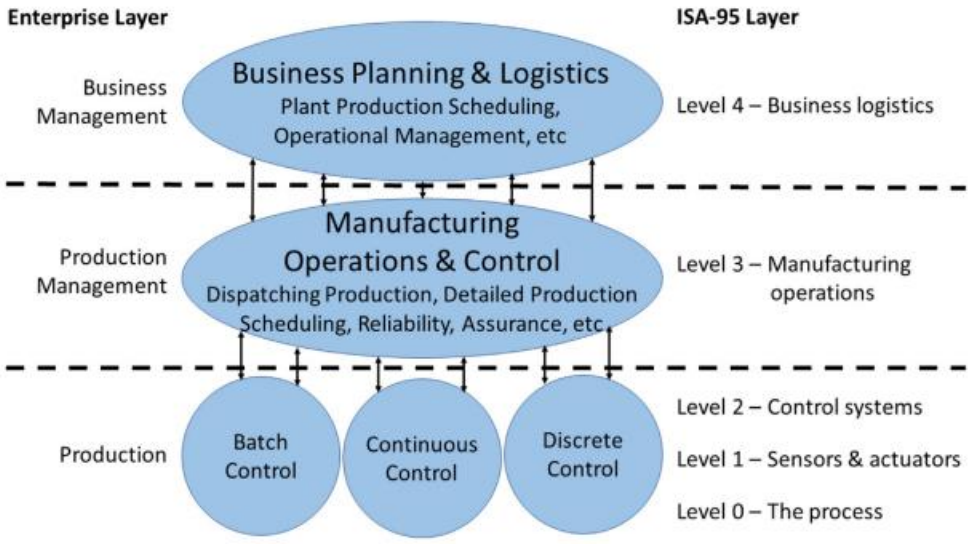

Figura 2.: Sistema hierárquico dos sistemas de informação proposto na norma *ISA* 95 [18]**.**

A norma considera quatro categorias de recursos (pessoal, equipamentos, materiais e energia e segmentos de processos) além de quatro modelos de processos, produtos e produção, que incluem a capacidade e respetivas definições, o produto, prazos produtivos e de entrega e desempenhos da produção. A nível empresarial, o *MOM* compreende a supervisão de uma vasta gama de elementos, desde as atividades de chão de fábrica às atividades de gestão e envolve várias operações de fabrico, matérias-primas, receitas, manutenção, testes de qualidade e expedição. A distribuição de funções pelos diversos sistemas que suportam diversas tarefas individuais da melhor forma possível, pode traduzir-se numa tarefa de fácil implementação. Porém, é necessário definir qual o sistema que vai gerir a respetiva informação. Por exemplo, o material de um determinado produto deve ser definido no *ERP*, porém, posteriormente, os atributos e dados podem ser adicionados pelo *MES*, para garantir que os diferentes aspetos produtivos são eficiente e viavelmente geridos. Embora o sistema *ERP* deva simplesmente fornecer planos de produção, o *MES* é responsável pelas operações no chão de fábrica. Os modelos formais de dados podem ser definidos com a utilização da linguagem *standard* para arquitetura de *softwares UML* (*Unified Modeling Language*), que representam a informação trocada entre o sistema *ERP* e o *MES* [18].

2.5. Planeamento e Implementação de um sistema MES

Os *MES* podem ser implementados com modos e características distintas, uma vez que não existem dois *MES* iguais. Os únicos conceitos recorrentes numa determinada instalação correspondem à execução e integração. Para se poder proceder à implementação de um sistema desta natureza em âmbito industrial, é necessário seguir um conjunto de ações e requisitos. É necessário estar ciente de toda a situação decorrente no *software ERP*. Para se proceder à implementação de um *MES*, são necessárias dez entradas ou *inputs* obrigatórios, como se verifica na Figura 3. A primeira entrada que deve existir corresponde à procura, que define os requisitos funcionais do *MES*, impulsionando o agendamento e a gestão de stocks. A segunda entrada corresponde ao agendamento (planeamento). A programação de alto nível e prazos de entrega são entradas diretas do *MES*, na camada do planeamento. Um cronograma básico fornecerá as diretrizes essenciais ao *MES* para conduzir a produção, que inclui os objetivos iniciais e finais. Por outro lado, temos o inventário, que é um dos aspetos mais relevantes para o controlo de bens de uma organização, ao nível do planeamento, permitindo acionar ordens de compra de acordo com os *stocks* controlados. Este processo decorre automaticamente através do acesso a uma lista de materiais (*BOM - Bill of Materials*) e de cálculos do inventário correspondente. A *BOM* assume informações como nome, quantidade, *Lead Time* [3] e outros parâmetros provenientes do *ERP*.

O *MES* efetua análises ao calendário, nomeadamente de acordo com os prazos de entrega previstos, custos, trabalho necessário, fomentando o cumprimento desses prazos. Se a empresa possuir um sistema de gestão de materiais, o *MES* consegue seguir o seu rasto, transferindo dados relevantes aos operadores sobre a localização e o estado dos materiais. Os dados produtivos correspondem ao sexto requisito do *MES*. O sistema sintetiza ou calcula qualquer informação, aliás funciona como um intermediário com a nuvem do *ERP*. O planeamento da capacidade total de produção também corresponde a um tópico relevante, nomeadamente o controlo da capacidade produtiva em cada etapa. Todos os *inputs* e *outputs* são devidamente tidos em conta, o que inclui a receção de bens dos fornecedores, a localização de inventários e o controlo da qualidade, sendo posteriormente devidamente documentado pelo *MES*. O nono *input* do *MES* corresponde aos dados operacionais reais, que compreendem grande parte dos *inputs* utilizados nos relatórios do sistema *MES*.

[3] *Lead Time* corresponde ao tempo total necessário para a conclusão de um processo, desde a solicitação do cliente até à entrega do produto [73].

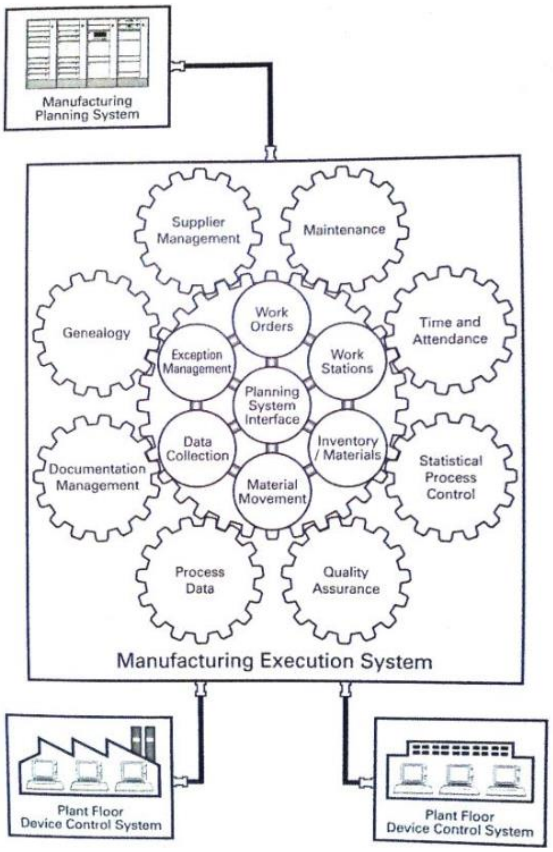

Figura 3.: Inputs do planeamento e controlo do *MES* [6]**.**

E, por último, temos a execução de ordens de alteração da engenharia *ECO* (*Engineering Change Orders*), que é mantido no *ERP* e este, posteriormente passa para o *MES* [6]. O *ECO* corresponde a uma ferramenta que descreve adições ou alterações de itens e das listas de montagens e materiais *BOM* propostas pelos desenhadores dos produtos correspondentes, geralmente utilizada durante a fase do desenvolvimento, visando a comunicação, a solicitação de aprovação e a implementação. A informação que consta no *ECO* inclui nomeadamente o motivo das alterações ou adições, os riscos associados, a documentação referente, detalhes da revisão, custos e desenhos [19].

A fase mais difícil da implementação do *MES* remete para a definição da especificação das funcionalidades desejadas do *software* e dos processos reais produtivos, desafiando pequenas e médias empresas a reverem e padronizarem criteriosamente os seus processos internos. O âmbito produtivo de uma organização pequena/média por vezes pode ser mais complexo do que aplicado a uma grande organização. Numa empresa mais pequena, uma mesma tarefa é desempenhada por apenas uma ou duas pessoas, que assumem uma metodologia própria para a implementação dessa mesma tarefa. Como cada operador tem uma perspetiva diferente para cada tarefa, as grandes organizações possuem um maior

espaço para abranger essas diferentes perspetivas e estabelecer uma normalização de determinadas atividades. Por outro lado, para a construção digital do chão de fábrica no *MES*, é necessária a definição de rotas para cada produto, processos e recursos, o que permite determinar o *layout* ideal a implementar no chão de fábrica. Um engenheiro de projetos de instalação deve analisar os movimentos físicos de produtos em âmbito produtivo. Após todas as especificações anteriores, pode-se seguir uma abordagem de planeamento sistemática para uma viável implementação do *MES* [6].

2.5.1. Metodologia V-Model

Da ideia inicial ao estado final, a monitorização das etapas de implementação do *MES* permite estabelecer um procedimento apto de descrição do melhor caminho para que a implantação do sistema e sua receção na organização sejam bem-sucedidas. Deste modo, devem-se considerar cuidadosamente todas as especificações, focando num pensamento de engenharia de sistemas. Na engenharia de sistemas, existe um modelo que permite efetuar um retrato dos processos de alto nível, fomentando uma categorização dos estágios de implementação. O modelo denomina-se de *V-Model* e pode ser verificado pela Figura 4.

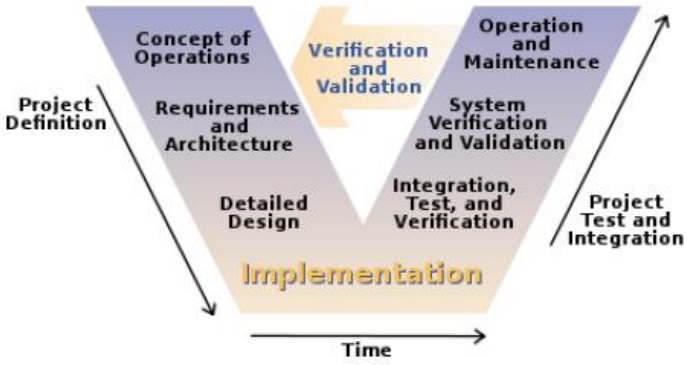

Figura 4.: *V-Model* dos sistemas de engenharia [6].

A primeira etapa do *V-Model* consiste no conceito das operações, coincidente com a conceção inicial do *MES*. A presente metodologia é constituída pelas catorze etapas seguintes para a implementação do *MES*, com base no artigo [6]:

- Etapa 1 – Exploração do *MES* – As organizações pesquisam sobre o tema, por meios pessoais ou por via de um consultor, visando solucionar os problemas de gestão da produção nas próprias organizações. Os pontos fortes do *MES* atraem os tomadores de decisão com promessas de melhorias. À medida que as camadas de

MES são avaliadas, a complexidade aumenta e os custos são um fator chave. Como saídas desta etapa, deve surgir um desejo de continuação do processo.

- Etapa 2 – Verificação da viabilidade do projeto *MES* – Como as soluções fornecidas pelo *MES* podem ser diversas e nem todas são necessárias, são definidos os requisitos iniciais e os envolvimentos das partes interessadas. Por outro lado, os meios reais de execução do projeto são utilizados pela primeira vez. A definição de recursos marca o início da segunda fase do *V-Model*. A definição de requisitos para aplicações específicas alterará as etapas de execução realmente utilizadas neste trabalho, mas, de um modo geral, ir-se-á considerar um *MES* completo. Como resultado desta etapa, deve ocorrer o envolvimento das partes interessadas, deve surgir uma discussão inicial sobre os recursos, identificando os recursos primários. Por fim, deve ser verificada a viabilidade do projeto. Estes fenómenos remetem para uma gestão de expetativas, equilibradas pelas restrições temporais e financeiras.
- Etapa 3 – Análise de projeto – São efetuados esforços de verificação dos estados atuais e futuros do âmbito produtivo, fomentando o caminho a seguir e quais os resultados esperados com tal abordagem. Nesta etapa, ocorre a procura de consultores por parte dos tomadores de decisões. Como *output*, deve surgir a assimilação dos requisitos do *MES,* pelos tomadores de decisões, e a finalização da documentação referente ao projeto de alto nível e à entrega do mesmo.
- Etapa 4 – Decisões estratégicas – As empresas deparam-se com o dilema de comprar ou fazer. Cada opção, possui vantagens e desvantagens, devendo ser comparadas lado a lado. A opinião proveniente da consultadoria poderá contribuir para a tomada de decisão. Acima de tudo, a decisão deverá ser devidamente informada e fundamentada. Com as linhas gerais elucidadas, as solicitações de propostas (*RFP*) são produzidas e enviadas por via de um consultor profissional, perito nas complexidades do *MES*. Por outro lado, outro fator significativo consiste no *ERP* existente e os seus módulos que interagem estreitamente com o sistema *MES*. Devido a esse relacionamento intensivo de dados, o *MES* selecionado deve interagir com o mínimo de problemas com os sistemas existentes. Tendo em atenção estas especificidades, pode ser selecionada a melhor abordagem.

A melhor solução varia. Por exemplo, numa empresa com uma variabilidade e volume de produtos relativamente altos, a melhor solução é comprar, uma vez que já existem sistemas no mercado que se encontram bastante desenvolvidos. Este tipo de empresa corresponde à melhor das aplicações do *MES*, visto que aproveita o máximo

de funcionalidades do sistema. Quanto à opção de construir um sistema, é menos comum de ocorrer, mas utiliza-se quando o produto ou ambiente de fabricação é mais obscuro e individualizado. Esta opção requer mais disponibilidade temporal e financeira. Uma outra consideração, um *MES* fornecido por pacotes *ERP* abrangentes, como *Oracle* ou *SAP*, deve ser selecionado, desde que o sistema *ERP* já esteja instalado. Nesta etapa, a segunda fase do *V-Model* termina, pois, os requisitos de sistema de alto nível são atendidos e o projeto básico foi estabelecido.

- Etapa 5 – Seleção do fornecedor – Deve ser efetuada uma escolha em relação a qual o fornecedor que vai implementar o sistema. Em média, uma organização solicita 2 a 5 orçamentos para o efeito. Nesta etapa, podem ser implementadas ferramentas como o processo de hierarquia analítica (*AHP*) ou por comparação por pares que podem ser utilizadas em análises qualitativa e quantitativas das soluções propostas. O *AHP* constitui uma ótima ferramenta para a determinação dos pontos fortes dos modelos, a níveis quantitativos. Esta ferramenta compara questões importantes a uma maior profundidade, incluindo a interface do utilizador, a funcionalidade, o tempo de execução, a gestão de mudanças ou o risco. Esses fatores são identificados e medidos qualitativamente para cada proposta. Por fim, por meio de cálculos matriciais, é identificada a proposta com maior pontuação.

- Etapa 6 – Declaração do Trabalho – Nesta etapa, ocorre a colaboração entre a organização e o fornecedor da ferramenta na definição dos aspetos do projeto, incluindo o produto piloto. Desta etapa, deve resultar uma declaração de trabalho (*SOW*) completa, devidamente assinada pelo fornecedor e pela organização. Nesta constam as responsabilidades que ambas as partes têm de seguir. Geralmente, inclui um plano de solução, um cronograma do projeto, um plano de recursos, um orçamento do projeto, uma avaliação do risco e um plano de formações. Também são definidos os protocolos de comunicação, as expetativas de recursos de ambas as partes e um plano de implementação piloto, onde surge a identificação do(s) produto(s) que trabalharão com o *MES*. Esta identificação deve ser implementada segundo três características principais, deve(m) ser (um) produto(s) que incorpore muitos dos processos de fabricação da empresa, o respetivo diagrama de fluxo deve ser analisado para identificar os modos de recolha de dados e deve(m) ter um grande impacto financeiro dentro da empresa para que as análises financeiras apresentadas junto com o programa também sejam fortes. Após a identificação do(s) produto(s), um programa piloto bem construído pode ser realizado. Uma *SOW* eficiente

determina uma solução que descreva a configuração da aplicação, bem como o *software* e *hardware* necessários para implementar o programa piloto, que inclui um cronograma que determina as entregas, recursos e durações do projeto. Por último, é efetuado um plano de treino/formação, pois o grupo de consultores deverá transferir os seus conhecimentos das operações para a organização em causa.

- Etapa 7 – Análise de lacunas – São analisadas as discrepâncias ou lacunas reais existentes que possam impedir a implementação do *MES*, o que inclui a estruturação dos dados, a interface com os sistemas circundantes ou os requisitos dos utilizadores. As informações disponíveis são variadas e devem ser representadas em bases de dados no *ERP* e devidamente assimiladas, para auxiliarem na identificação de problemas que possam surgir no programa piloto. A interface do sistema deverá possuir os dados como base. O rigor dos dados, tanto no conteúdo quanto no armazenamento, determina o sucesso do programa como um todo. Deve-se proceder à modelação real do chão de fábrica com todos os seus elementos. Por último, a interface real utilizada pelo utilizador é desenvolvida. Uma análise completa de lacunas também evidencia às empresas de adoção e consultoria uma compreensão clara do trabalho necessário para implementar o *MES*. Como resultado desta etapa, deve ser concluída a análise de lacunas e dos requisitos para a integração, para além de uma assimilação das etapas necessárias para a integração.
- Etapa 8 – Arquitetura do Sistema – A partir das especificações, desenvolve-se o *software* e o *hardware*. Estas especificações baseiam-se nos requisitos do sistema previamente definidos. A dimensão do sistema define a complexidade e limita os requisitos do *hardware*. Os níveis mais altos de complexidade exigem um melhor *hardware* em termos de capacidade de armazenamento de dados, conexões periféricas (estações de trabalho, máquinas e impressoras), redes locais e requisitos computacionais, o que representa um maior custo. O objetivo principal de qualquer *software* adicional seria processar dados para o seu uso no *MES*. A complexidade do sistema, os requisitos de outros *softwares* com os quais o *MES* comunica e a variedade dos dispositivos de recolha de dados definem a extensão da arquitetura de programação adicional necessária. No fim desta etapa, a documentação relativa aos sistemas de comunicação, *layout* e arquitetura do sistema devem estar bem definidos.
- Etapa 9 – Programa Piloto – Devem ser assimiladas as funcionalidades pré-definidas para o programa piloto e os modos de determinação dos dados. A determinação do fluxo de processos permite a definição das estações de trabalho, das

operações, das máquinas, das medidas de qualidade, do transporte e dos locais de armazenamento com os quais o produto interage. Estes são os pontos essenciais de recolha de dados necessários para implementar as funcionalidades básicas do *MES*.

Como resultado, deve surgir documentação que detalha os elementos produtivos e os pontos de recolha de informação, de modo a auxiliar a fase de execução.

- Etapa 10 – Implementação – Após a definição e detalhe de todo o projeto, procede-se à sua execução em termos práticos. A definição das operações necessárias para fabricar o produto piloto domina o início da fase de implementação. Cada operação deve ser definida e armazenada para referência no *MES*. Devem ser adicionados os recursos e credenciais de mão de obra e das máquinas, assim como o os requisitos de tempo e material para cada etapa produtiva. Deste modo, estabelece-se o fluxo e as ordens de trabalho, e as operações são devidamente rotuladas, ou seja, atribui-se uma etiqueta identificativa, para ocorrer em série ou em paralelo. O *MES* utiliza o detalhe das operações específicas para construir cronogramas ideais para execução rápida. O armazenamento e o inventário também devem ser definidos para que o *MES* construa totalmente o ambiente de fabricação. A capacidade e níveis de *stocks* devem ser referidos para cada local de armazenamento.

O sistema de planeamento é um dos elementos complexos do *MES*. Geralmente, o *MES* inicia-se com o agendamento mais simples, com base na capacidade da área de produção e nos prazos de entrega. Porém, é possível iniciar com planeamentos mais complexos. As funcionalidades básicas do *MES* geralmente identificadas no projeto piloto, incluem as seguintes: execução orientada por despacho; visualização das ordens de trabalho e suas instruções; funcionalidade de entrada/saída dos funcionários; relatórios e controlo de transações; gestão de exceções; implementação em série; dispositivo de chão de fábrica e integração de equipamentos de teste; impressão automatizada de rótulos e etiquetas; validação de *skills* laborais; painel de supervisão. Com a implementação bem-sucedida das funcionalidades, um *MES* funcional do programa piloto pode ser implementado. Todos os elementos requerem que ocorra a recolha de dados e os pontos de entrada (máquinas e equipamentos, operadores, sistemas de controlo de qualidade, sistemas *ERP* e sensores) são definidos, para cada operação. Assim, o *MES* controla a execução do plano estabelecido e reagenda caso desponte algum erro ou alteração.

A execução orientada por despacho serve ao operador como o ponto de partida para todas as ações. Além de apresentar as ordens de serviço por prioridade, oferece a capacidade de registar operações relacionadas com o trabalho. Estes relatórios de operações incluem horários de início e fim, ordens de trabalho, refugo ou desperdícios e exceções. Estes dados, gerados automaticamente ou pela intervenção do operador, são enviados para o *MES*, podendo, se necessário, dar origem a alterações ao cronograma. A apresentação da ordem de trabalho é um dos impulsionadores na eliminação do desperdício e na busca por um ambiente de fabricação sem papel, mostrando informações relevantes acerca do fluxo produtivo.

O *MES* dá suporte à entrada/saída de horários, com foco no sistema *ERP*, em que o operador registaria estas informações manualmente. Como este sistema estimula o controlo de horários e de ordens de trabalho, os cálculos para o tempo total gasto para um determinado projeto são simplificados. O controlo do tempo é um ótimo exemplo de comunicação da interface *MES* com o *ERP*. Os dados reais são comunicados ao *ERP* e são aludidos na fabricação e na documentação financeira. Uma interface de utilizador bem concebida permite ao operador relatar as operações, duração de tempo despendido e resultados de qualidade. Com a gestão de exceções, os operadores podem relatar eventos como componentes ausentes/inadequados, máquinas inoperantes, problemas fatais de qualidade ou operadores ausentes. Esta informação é utilizada para alertar os supervisores e para evitar quotas de produção perdidas. Aquando da informação de um problema, o *MES* procede a uma reavaliação de prioridades e ao reagendamento de operações.

A serialização ou sequenciação de operações existe como uma tarefa acessível no *MES* quando comparada com a respetiva complexidade nos sistemas controlados em papel. Muitos dos dados de serialização são inseridos gradualmente à medida que os subconjuntos são construídos ou que as ordens de encomenda são recebidas e são predeterminados por algoritmos associados a regras e padrões utilizados para nomear e categorizar as ordens de compra de forma sistemática e consistente que existem noutros módulos através do *ERP*. O *MES* utiliza os números de série predeterminados para compilar, armazenar e rever históricos. Os *MES* bem integrados ocasionalmente interagem com as informações do fornecedor, o que possibilita a capacidade de rastrear a produção até à sua origem. A serialização também incrementa a capacidade

de atribuir determinada produção a ordens de serviço concretas e manter o controlo dos produtos específicos durante toda a produção. A impressão de rótulos e etiquetas pode ser automatizada com o *MES*, de acordo com a necessidade existente. Com o *MES*, podem ser armazenados registos que avaliam a capacidade do operador de realizar uma tarefa, permitindo reconhecer a adequação da tarefa, a eventual necessidade de formação ou até a sua salubridade e segurança.

Por fim, o painel do supervisor será implementado. A interface de utilizador oferece uma interação mais complexa com o *MES*. Nesta aplicação, os trabalhos podem ser acelerados, as prioridades reatribuídas e as exceções resolvidas. Este nível de controlo permite melhorar a tomada de decisões, em que a produtividade é maximizada e as operações contínuas são asseguradas. O painel do supervisor extrai os dados armazenados para fornecer aos gestores informação em tempo real sobre as ações do chão de fábrica. As principais métricas incluem a produção de acordo com o plano (determina a progressão conforme os prazos estabelecidos), a escassez de ordens de trabalho (identifica ausência de recursos ou capacidade), as métricas de desempenho laboral e o denominado '*First Pass Yield*' (que emite alertas relativas à qualidade do produto). Em suma, como saída desta fase, deve surgir um protótipo funcional do sistema *MES* específico para o produto piloto.

- Etapa 11 – Teste do protótipo – O teste inicia-se com uma reunião entre os membros de alto nível dentro da organização, atribuindo papéis teóricos, em que o fornecedor exerce formações aos utilizadores do sistema, acerca do seu funcionamento e controlo. Além disso, o *MES* piloto será testado em ambiente de fabricação real, aquando devidamente integrado no chão de fábrica e nas atividades diárias. Em suma, para que o critério de saída seja atendido, o sistema *MES* deve estar instalado e utilizado com sucesso pelas funções de operador e supervisor.
- Etapa 12 – Verificação e validação – Surge uma nova fase do *V-Model* e uma validação e qualificação das vantagens diretas e indiretas do sistema, determinando a adequação da implementação aos níveis de funcionalidade desejados. Este estágio inicia-se após a recolha de dados suficientes para efetuar cálculos de comparação entre os estados antes e depois, atribuindo custos e vantagens diretas e indiretas. A validação do sistema procura garantir que o *software* implementado efetivamente atendeu aos níveis desejados de funcionalidades, sendo realizado por meio de testes operacionais e por meio de comparação com características predeterminadas na

SOW. Além disso, é efetuada através do cálculo dos custos produtivos, que são certamente inferiores aos obtidos anteriormente à aplicação do *MES*. Outras vantagens correspondem à minimização do tempo de receção, introdução e tratamento dos dados obtidos. As vantagens diretas obtidas são mais fáceis de quantificar, porém as vantagens indiretas (como a melhoria na qualidade do produto, redução de desperdícios e melhoria na tomada de decisões) são muito superiores. As vantagens indiretas incluem uma maior satisfação do cliente, uma maior responsabilidade e comunicação dos funcionários, relatórios mais precisos, uma manutenção de registos históricos, maior visibilidade das operações e qualidade. As vantagens diretas remetem para um aumento da eficiência operacional, melhoria na rastreabilidade e da gestão de recursos e na redução de erros. Deste processo, deve resultar um sistema validado e verificado quanto aos objetivos estabelecidos.

- Etapa 13 – Implementação em grande escala – As ações utilizadas na implementação piloto são aplicadas a outras linhas de produtos e funcionalidades. A operação em escala real mostra-se mais complexa, podendo ocorrer problemas quando as linhas de produtos partilham mão de obra, máquinas e matéria-prima. A operação *MES* em grande escala é muito mais fácil quando uma sólida prova do conceito foi implementada porque a maioria das estruturas de dados, linhas de comunicação, operações, e interfaces de utilizador já existem. As métricas devem ser novamente impostas ao novo sistema num esforço de verificar e validar o sistema. A implantação do *MES* no chão de fábrica ganha força quanto mais linhas de produtos estiverem envolvidas e o nível de sucesso aumentar. Como *output*, surge a verificação e validação do *MES* aplicado a todo o sistema produtivo.
- Etapa 14 – Monitorização e Manutenção – Deve ser mantida a operação diária e devem ocorrer adições e/ou subtrações de funcionalidades para melhor auxiliar o sistema produtivo. Desta fase, entra-se oficialmente no ciclo de vida *MES*.

A proposta dos catorze passos anteriores visa a implementação viável de um sistema MES e uma potencial solução para gestão da produção nas empresas atuais [6].

2.5.2. Metodologia fundamentada na ISA 95

Por outro lado, existe outra metodologia proposta no artigo [4], cuja implementação de um *MES* numa determinada organização remete para 5 etapas distintas: 1) Avaliação inicial; 2) *Design*; 3) Configuração, construção e Teste; 4) Implantação e 5) Operação. Esta

metodologia encontra-se sucintamente apresentada na Figura 5. A metodologia de execução de um *MES* foca-se nos seguintes estágios:

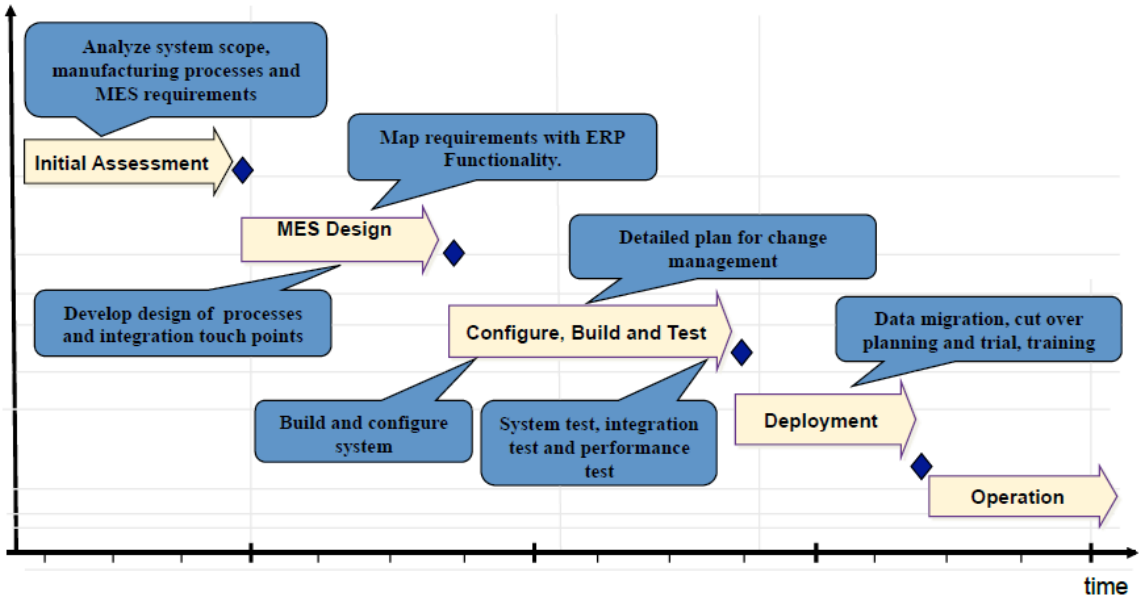

Figura 5.: Proposta da metodologia de implementação de um *MES* [4].

1) Avaliação Inicial – Existem duas atividades-chave seguintes implementadas neste passo.

- Determinar o alcance da implementação – De acordo com o modelo hierárquico de sistema do *ISA 95*, existem cinco níveis de sistema quanto aos processos de fabrico, do qual o *MES* situa-se no nível 3 (pode ser visualizado no Tópico 2.4, na Figura 2). Uma das principais preocupações corresponde à distribuição das funções pelos diversos sistemas, que devem suportar as tarefas individuais da melhor forma possível. Além disso, os modelos hierárquicos dos equipamentos da *ISA-95* que mostram a hierarquia dos ativos físicos das empresas envolvidas em atividades de fabricação, podem ser utilizados para determinar o limite físico do sistema *MES*.
- Analisar os requisitos funcionais do *MES* – A informação disponível sobre o *MOM* (*Manufacturing Operations Management*), contida na parte três da norma *ISA-95*, pode servir de guia para analisar os requisitos do sistema. O modelo *MOM* contém a descrição dos aspetos funcionais do *MES*. Podem ser utilizados os diagramas disponibilizados na norma para guiar o desenvolvimento.

2) Design do *MES* – Nesta fase, ocorrem as duas etapas seguintes.

- Design genérico – Este divide-se em duas partes distintas, o modelo de função genérica e o diagrama de sequência genérica. O modelo de função genérica encontra-se nas partes um (Modelos e Terminologia) e três da norma que ajudam a identificar as principais atividades relativas à gestão das operações produtivas. Por outro lado, auxilia na identificação da informação que circula entre as diferentes atividades da organização. Uma fronteira pode ser delineada entre os níveis 3 e 4, porém existem algumas etapas que atravessam ambos os níveis, como a gestão da manutenção preventiva e corretiva, de inventários e da qualidade [20]. Pode ser utilizada uma metodologia de modelação para descrever funções de fabricação, assim como, ações e atividades numa organização, denominada de *IDEF0* (*Integration Definition for Process Modelling*), cujo nível inerente pode ser determinado pela equipa responsável pela implementação do *MES*. O *IDEF0* corresponde a um dos métodos *IDEF*. Este corresponde a um conjunto de métodos *standard* e a uma família de linguagens gráficas utilizadas na modelação da informação na área da engenharia de software e nos processos negociais [21]. Deve ser definido um *IDEF0* genérico que inclua todas as atividades do nível 3 e as respetivas comunicações com algumas atividades do nível 4. Com o *ISA-95*, o modelo funcional é desenvolvido de forma a separar os processos de negócio dos processos produtivos. Desta forma, permite que os processos de nível 3 (produção) ocorram sem nenhuma exigência dos processos de nível 4 (*ERP*).

  O diagrama de sequência genérica representa a informação relativa à ordem em que as diferentes atividades ocorrem no fluxo de produção, fornecendo uma perspetiva sobre a implementação das atividades. Nesta fase, recorre-se a diagramas *UML* (Linguagem de modelação unificada) que mostram a troca de informação e o modo como a comunicação evolui entre os diferentes elementos. Este diagrama deve descrever toda a troca de informação decorrente entre os níveis 3 e 4, tendo em conta todas as atividades e objetos identificados no diagrama.

- Design Específico – Esta etapa divide-se em duas partes, modelo de função específico e diagrama de sequência especifica. O modelo de função específico é uma adaptação do modelo genérico a cada caso de aplicação. O primeiro passo, é definir o *IDEF0* específico da organização em causa,

devendo formar-se uma equipa multidisciplinar dentro da organização. Deve-se construir o modelo anterior tendo em conta o estado final espectável, que se possa concretizar. Por outro lado, desenvolve-se o *UML* específico aplicado à organização em causa, através do *UML* genérico, tendo em consideração o modelo *IDEF0* criado no primeiro passo.

3) Configuração, construção e teste – Nesta fase, o objetivo corresponde à configuração, construção e teste dos componentes do módulo aprovado no *design* específico, de acordo com as respetivas especificações que foram desenvolvidas com base nos requisitos do *MES*. De um modo geral, a aplicação do *MES* é desenvolvida, a migração de informação é realizada e são implementados testes ao sistema. Estes testes incluem um teste de unidade (verifica-se o funcionamento de componentes individuais do *software*, como funções ou métodos específicos), de integração (verifica-se a interação entre diferentes módulos ou componentes do sistema) e de desempenho (avalia-se como o sistema se comporta sob diferentes condições de carga e uso).

4) Implantação – Nesta fase, a preparação final do sistema de transição é implementada. Ocorrem formações e atividades de correções de erros, antes que o sistema novo e final seja colocado em operação.

5) Operação – O novo sistema é colocado em operação e a equipa de apoio pós-projeto é usada para auxiliar os novos utilizadores a trabalhar neste novo sistema. É efetuada uma auditoria final à qualidade do sistema.

2.5.3. Metodologia sustentada no WS

Esta abordagem ao *MES* remete para uma metodologia adotada por uma organização que opera no âmbito das embalagens alimentares [8], constituída por cinco etapas representadas pela Figura 6. A abordagem encontra-se integrada com o *WS* (*Weihenstephan Standards*) que é um modelo de informação padrão que define uma interface de comunicação universal entre vários sistemas de controlo de processos e o *MES [22]*. Com o *WS*, o processamento dos dados da máquina e do processo podem ser definidos de uma forma sólida para as funções do *MES [23]*.

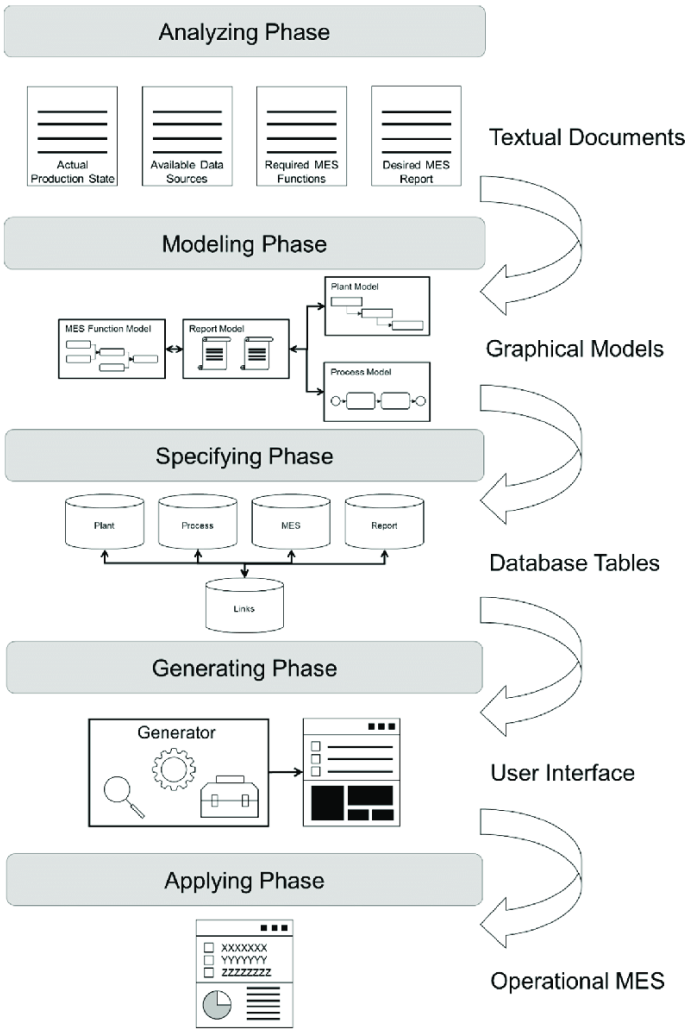

Figura 6.: Metodologia de implementação adotada [8]**.**

- Etapa 1 – Estudo preliminar – Neste estágio, efetua-se uma análise quanto ao estado real da instalação produtiva, os seus processos produtivos, as fontes de dados adquiríveis, os objetivos e os relatórios a obter com o *MES*. Esta fase destina-se aos utilizadores finais. Além disso, deve reunir os membros da organização com diferentes focos e experiências de trabalho, de modo que, juntos, consigam convergir numa solução *MES*, com as funções adequadas e um fluxo de informações completo entre departamentos. Como resultado desta etapa, deve surgir a documentação relativa à situação real do estado da produção, incluindo a descrição dos sistemas técnicos e processos produtivos, as fontes de dados disponíveis, as funções requeridas e o relatório com o resultado pretendido da implementação do *MES.*

- Etapa 2 – Modelação do *MES* – Os objetivos definidos na etapa anterior devem ser modelados. Para isso, é necessária a aplicação de uma sintaxe específica para obter modelos comparáveis e implementações de ferramentas compatíveis.

Deste modo, surge a linguagem de modelação *MES* (*MES-ML*) [24] formal que provou ser adequada para modelar o *MES* com um conceito orientado aos modelos. Esta pode ser estendida para atender a requisitos específicos de determinados processos produtivos ou ferramentas. Na *MES-ML* estendida, o modelo da instalação é estruturado de acordo com um diagrama de árvore com seis níveis hierárquicos para apresentar os sistemas técnicos inspirados na norma *ISA-95*, ou seja, a fábrica, a área, a secção, a linha, o equipamento e associados. Sob cada elemento técnico no nível de hierarquia, podem ser atribuídos pontos de dados para indicar a informação que pode ser fornecida. Para modelar a função *MES*, os dados dos sistemas técnicos e dos processos são as entradas e os resultados das funções *MES* são as saídas dos relatórios. Cada elemento do relatório pode apresentar as informações de um determinado modo, através de texto, tabelas ou diagramas, e está conectada a uma função *MES* que fornece as saídas. Embora os dados necessários de elementos técnicos e/ou processos a serem tratados pelas funções *MES* sejam modelados, as fontes de dados não são fixas na fase de modelação. Ao definir as entradas e saídas dos relatórios, primeiramente são estabelecidas as ligações entre o modelo da instalação, o modelo de processo e o modelo de função *MES.*

- Etapa 3 – Especificação – A principal tarefa desta fase é transformar as informações dos modelos gráficos num formato que o *software* possa utilizar. As bases de dados são a plataforma para a especificação do *MES*. As diferentes tabelas relacionadas com os modelos na fase de modelação são definidas para representar as informações no modelo gráfico.

Nesta fase, o processo de transformação é um processo de planeamento, pois a estrutura, o conteúdo e a relação das tabelas foram definidos como uma representação um-para-um do metamodelo *MES-ML* estendido. Esta fase pode ser automática através de uma ferramenta de planeamento que conhece a relação entre o *MES-ML* estendido e as tabelas das bases de dados. Por outro lado, pode-se proceder à implementação de diagramas lógicos de entidade-relacionamento (*ERD*). O *ERD* fornece aos programadores uma compreensão geral dos requisitos de dados, modelação, interações e estruturas de base de dados do sistema de informação antes da fase de implementação [25] de um *MES*. Os sistemas técnicos são

modelados através de diferentes hierarquias e pontos de recolha de dados. Para descrever o modelo de instalação do *MES*, é necessário associar cada elemento do sistema às tabelas *'Location'* e *'LocationDataPoint'*. (Figura 7, à esquerda). A tabela "*Location*" contém atributos como a fábrica, a área, secção, linha, máquina e associados sendo utilizada para apresentar os elementos de modelação em cada hierarquia. A "*LocationDataPoint*" apresenta os pontos de dados em cada elemento. Os tipos e descrições dos pontos de dados também podem ser encontrados com os atributos "*DataPointType*" e "*DataPointDescription*". Além disso, o atributo "*LocationLink*" indica a relação entre um determinado ponto dos dados e o seu elemento hospedeiro. Semelhante ao modelo de instalação, as informações no modelo de processo podem ser representadas com duas tabelas denominadas "*Process*" e "*ProcessDataPoint*". O nome dos elementos de modelação em cada nível de hierarquia no modelo de processo pode ser encontrado na tabela "*Process*". O tipo dos pontos de dados do processo utilizados e ligações podem ser descritos com os atributos "*DataPointType*" e "*ProcessLink*". Utilizando os atributos "*ProcessLink*" e "*LocationLink*" na tabela "*ProcessLocation*", pode-se encontrar a ligação entre os processos e os sistemas técnicos.

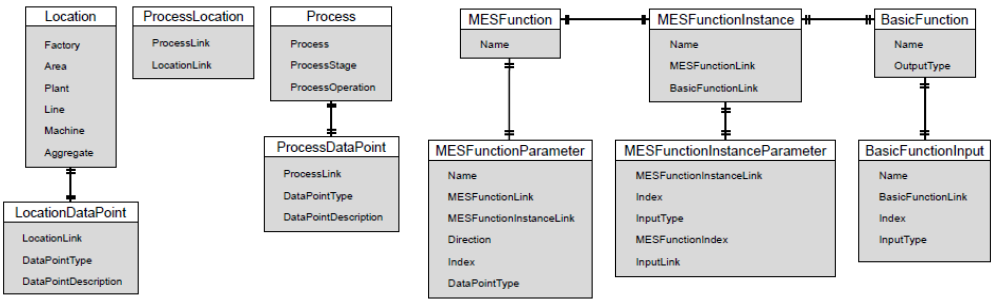

Figura 7.: Relações entre as entidades através de um *ERD* (à esquerda) e descrição das funções básicas das funções *MES* (à direita) [8]**.**

Como as funções *MES* são compostas por funções básicas, existem seis tabelas para representar as informações dessas funções básicas, funções *MES* e a relação lógica entre elas (Figura 7 à direita). As funções básicas utilizadas para compor as funções *MES* e os seus parâmetros de entrada estão descritas nas tabelas "*BasicFunction*" e "*BasicFunctionInput*". As tabelas "*MESFunction*" e "*MESFunctionParameter*" permitem representar o nome da função *MES* e os atributos necessários do total de parâmetros de entrada e saída para realizar esta função *MES*, como a "*Direction*" para diferenciar entre uma entrada (Parâmetro "Direction" = 1) e uma saída ("Direction" = 0). As tabelas "*MESFunctionInstance*" e "*MESFunctionInstanceParameter*", servem como pontes na especificação do modelo para

indicar, de um lado, a ligação entre as funções básicas e as funções do *MES* e, do outro, a atribuição dos parâmetros de entrada e saída das funções *MES* aos parâmetros de entrada das funções básicas. Os dados do modelo do relatório podem ser transformados através das quatro tabelas da Figura 8. A tabela "*Report*" contém o nome e o tipo de relatório. Como um relatório pode conter vários elementos na apresentação dos resultados de diferentes funções *MES*, e cada uma destas necessita de diferentes fontes de dados do modelo de instalação e/ou modelo de processo, definem-se as tabelas "*ReportElement*" e "*ReportInput*". A "*ReportElementOutput*" é responsável por estabelecer a ligação entre os elementos do relatório e as funções do *MES*.

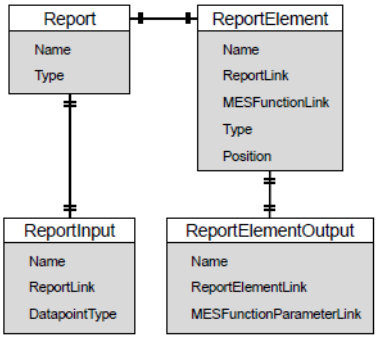

Figura 8.: Tabelas para transformação do modelo [8].

- Etapa 4 – Geração (Parametrização e Implementação das funcionalidades) – Com base na especificação, com as funções exigidas, sem valores para parâmetros de entrada, o *MES* é desenvolvido automaticamente através de um gerador.

Na criação da aplicação, o gerador é constituído por dois componentes, o *frontend* e o *backend*. O *frontend* é uma interface gráfica que o utilizador consegue visualizar e em que pode parametrizar as entradas do *MES*. Os seus programadores são responsáveis pelo desenvolvimento visual da aplicação, visando ser o mais intuitiva possível [26]. O *design* da interface gráfica do utilizador depende da especificação do *MES*. Este contém duas áreas, de entrada e de saída. Na área de entrada, os parâmetros são listados e os valores estão prontos para serem selecionados e/ou modificados pelo utilizador final. O *backend* é o responsável pelo processamento de dados do *MES* e consiste num servidor, numa aplicação e numa base de dados, geralmente em linguagens como *PHP*, *Ruby*, *Python* [26]. Após as funções *MES* terem processado os dados de acordo com os valores dos parâmetros de entrada, os resultados são devolvidos como os valores dos parâmetros de saída, mostrados na área de saída (Figura 9).

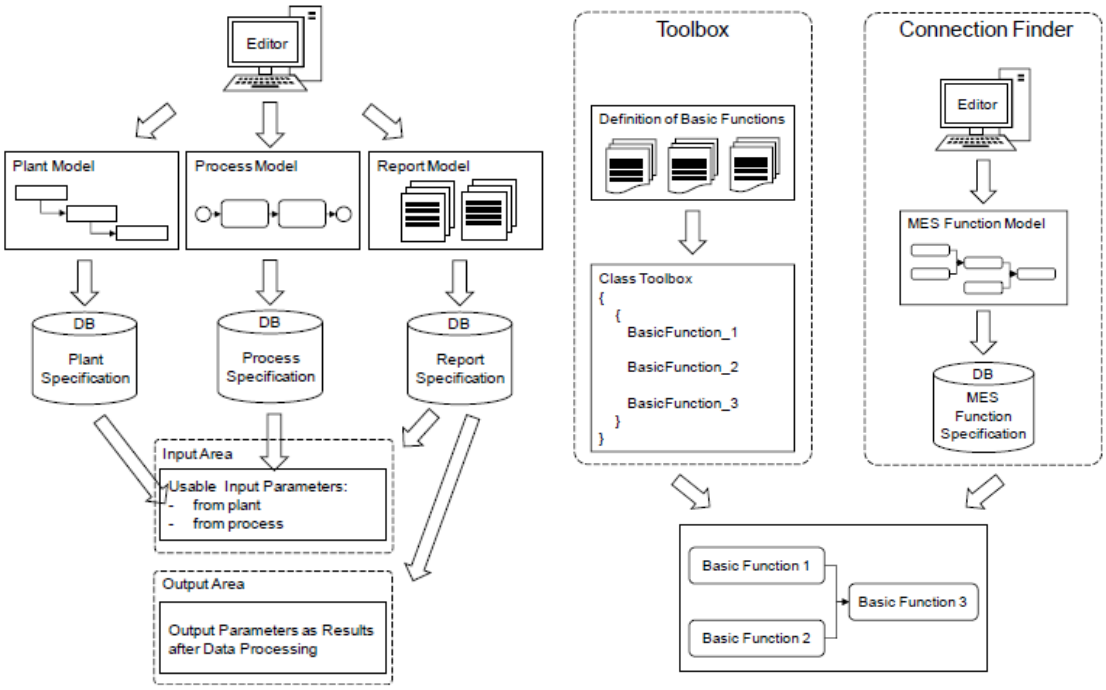

Figura 9.: Parametrização do *Backend* (à esquerda) e local de teste de funções *toolbox* (à direita) [8].

As funções *MES* necessárias já estão executadas nesta fase. Nas fases de modelação e especificação, as funções são apenas definidas ou declaradas, com uma descrição para as respetivas entradas e saídas, bem como com o nome das funções básicas que utiliza e a sua interligação. As funções básicas para o processamento de dados estão definidas e são executadas primeiro como procedimentos executáveis na caixa de ferramentas (*toolbox*) disponível, para desempenhar tarefas. O localizador de ligações que pode invocar as funções básicas da *toolbox* é responsável por reproduzir o fluxo de sequência das funções básicas para garantir a ordem correta de passagem de valor entre as chamadas.

- Etapa 5 – Aplicação / Implementação – Nesta fase, a procura sobre o *MES* e as suas entradas são parametrizadas pelo utilizador final, para que o *MES* específico desejado possa ser desenvolvido. Geralmente, o processamento de dados pode ser dividido em quatro etapas. Primeiro, ocorre a recuperação das funções básicas necessárias da *toolbox*. Segundo, atribuem-se os parâmetros de entrada da área de entrada às funções básicas. Em terceiro lugar, processam-se os parâmetros de entrada através da sequência modelada de funções básicas e em quarto lugar, mostram-se os resultados como parâmetros na área de saída da interface gráfica do utilizador.

- Etapa 6 – Avaliação – Neste estágio, ocorre a verificação do estado da eficiência do sistema global do *MES*, determinando se este se encontra de acordo com o pré-definido ou se necessita de correções [8].

2.6. Integração do MES com o *Lean*

As ferramentas do *Lean Manufacturing* (*LM*) sempre foram reconhecidas pelo facto de auxiliarem as organizações a incrementar a produtividade e flexibilidade, simultaneamente com a minimização de custos. Porém, os reais benefícios desta prática atingem apenas uma pequena percentagem de empresas que adotam esta prática [27]. O *LM* corresponde a uma filosofia de gestão, desenvolvida pela *Toyota Production Systems* (*TPS*) e foca-se nas atividades de valor acrescentado e na remoção de desperdícios. As técnicas *LM* são bastante reconhecidas por permitirem atingir uma maior estabilidade, flexibilidade e dinamismo, permitindo dar uma melhor resposta às necessidades do cliente. O *LM* baseia-se nas seguintes melhores práticas.

- Melhoria Contínua (*Kaizen*) – O *Kaizen* corresponde a uma filosofia japonesa que promove a melhoria dos resultados produtivos, com o incremento do envolvimento humano. Com esta filosofia, todas as partes envolvidas crescem simultaneamente. Apesar de todos conseguirem compreender relativamente bem a sua relevância, nem todos conseguem eficazmente implementar estas práticas, geralmente devido a problemas de comunicação ineficiente ou até na inexistente ou ineficaz partilha de conhecimento. O *MES* fornece dados em tempo real relevantes acerca do processo produtivo, criando informação de base que pode ser usada para alimentar o processo de melhoria contínua [28]. A integração do MES com o *LM*, incita ao desenvolvimento de *dashboards* inteligentes que elaboram *KPIs* (*Key Performance Indicators*) e gráficos em tempo real, além de análises *PDCA* (*Plan-Do-Check-Act*) [29] que, integradas com o *MES*, possibilitam a minimização do tempo de ciclo e melhoram o desempenho do processo.
- Eficiência do Fluxo – Os princípios de fluxo contínuo e da produção que é iniciada com base na procura real dos clientes, em vez de previsões de procura [30], servem de base ao conceito *Just-In-Time* (*JIT*)[4]. O *JIT* constitui uma estratégia de gestão da produção e de inventário que visa a identificação, minimização e

[4] Também referida na literatura como "produção puxada".

eliminação dos desperdícios, ou seja, atividades que não acrescentam valor ao processo produtivo e que constituem o coração do *LM*. Podem ocorrer falhas decorrentes de problemas de contagens relativos ao inventário e à capacidade de stocks, da existência de sistemas de controlo centralizados e da comunicação ineficiente. A flexibilidade do sistema pode ser insuficiente para lidar com ambientes produtivos altamente complexos. Com o controlo dos processos em chão de fábrica, o *MES* pode ser utilizado para identificar desperdícios e identificar cadeias de valor. Os dados obtidos podem ser úteis para reorganizar o layout da unidade industrial (UI) e do armazém, assim como a posição das máquinas. Por outro lado, podem ser efetuadas previsões relativas ao sistema produtivo. Além disso, para facilitar o controlo da localização de itens na organização, surge a tecnologia *RFID* (*Radio Frequency IDentification*), que utiliza ondas de rádio para identificar e rastrear itens automaticamente [31], o que permite que um *MES* possa apoiar o planeamento avançado de produção, combinando uma melhor gestão e conetividade no chão de fábrica.

- Mudança rápida de ferramenta – O conceito de mudança de ferramenta em um minuto (*SMED)* corresponde a uma metodologia para melhorar a produtividade, através da minimização de tempos de *setup*. Através de reduzidos tempos de *setup* é possível adaptar melhor a produção à necessidade do cliente e da indústria, o que se traduz progressivamente no conceito de customização em massa. Assim, a maximização da implementação de atividades *standard* minimiza os tempos de *setup*, o que determina uma maior disponibilidade da máquina para a atividade produtiva geradora de valor. Uma vez que o *MES* efetua registos automáticos de dados referentes ao processo produtivo, a conexão das máquinas do chão de fábrica com os sistemas *MES* permite que a informação obtida possa ser descarregada e carregada automaticamente nos locais apropriados, levando a uma monitorização, controlo e planeamento de *setups* mais eficientes.
- Manutenção Produtiva Total – O âmbito da manutenção dos equipamentos no chão de fábrica sempre foi uma questão fundamental para as organizações. Existem sempre pausas planeadas e principalmente não planeadas originadas, por exemplo, pela produção de peças defeituosas ou por erros nos equipamentos, o que interfere com a produtividade. O *Total Productive Maintenance* (*TPM*) corresponde às atividades que visam a maximização da eficiência dos equipamentos, que devem ser suportadas por um grupo de pessoas devidamente treinadas/formadas, que incluem

elementos de gestão de topo e trabalhadores de chão de fábrica [32]. O *MES* pode melhorar as práticas do *TPM*, permitindo alcançar uma maior qualidade [33]. Adicionalmente, o *MES* pode distribuir documentos e informação relevantes aos operadores das máquinas, através de um terminal pré-definido [34]. Por outro lado, as tarefas de manutenção podem ser rapidamente configuradas, planeadas e documentadas, auxiliando no cálculo do indicador *OEE* (*Overall Equipment Effectiveness*), que corresponde a uma métrica utilizada para avaliar a eficiência de um equipamento de produção, tendo em conta os fatores de disponibilidade, desempenho e qualidade [35], o que permite tomadas de decisão mais rápidas.

- Gestão da Qualidade Total – O *Total Quality Management* (*TQM*) corresponde a um dos conceitos *Lean* decorrentes em torno da qualidade, que se foca na minimização de custos, no incremento da eficiência e na entrega de produtos finais que correspondam exatamente à qualidade dos produtos solicitados pelo cliente. Por outro lado, deve visar o alerta e resolução de problemas decorrentes do processo produtivo, devendo estes serem assimilados num sistema de monitorização e controlo, para se poder responder eficazmente na próxima vez que ocorrer um problema semelhante [36]. A integração do *TQM* com o *MES* permite um melhor suporte à melhoria contínua dos *standards* de qualidade e providencia análises do controlo estatístico do processo (*Statistical Process Control*), em tempo real, reportando os acontecimentos relevantes em termos dos parâmetros de qualidade.
- Foco no cliente – A base do *Lean* sustenta-se nas necessidades do cliente, o que comprovadamente se traduz no crescimento sustentável e num incremento do lucro obtido pelas organizações. As empresas adotam uma normalização dos seus processos produtivos de modo a maximizar a eficiência produtiva e melhor servir o cliente. Diferentes parâmetros podem ser controlados em qualquer fase do processo produtivo. Relacionando a ideia de foco no cliente com o MES, ainda não se consegue afirmar que este possa contribuir para este tópico, mas encontram-se em desenvolvimento estudos acerca disto [37].

De facto, o *MES*, dentro de todos os sistemas de informação existentes, é o que converge mais com o *LM*, na sua diversidade de funcionalidades, o que inclui a emissão de análises em tempo real. Muitas das soluções *MES* já possuem incorporadas algumas funcionalidades de *Lean*, embora sendo ainda poucas na maior parte dos casos [28].

2.7. Indústria e Tecnologias 4.0

No âmbito da quarta revolução industrial, a publicação de um artigo [38] preocupou as equipas de investigação mundiais devido às previsões sobre os sistemas de gestão da produção e integração dos *IT*. O governo alemão definiu os investimentos na investigação e desenvolvimento em torno do tema Indústria 4.0 (I4.0) para os anos seguintes, com o objetivo principal de manter a liderança em termos de maquinaria e indústria automóvel, de modo a ser o país a liderar a quarta revolução industrial.

A filosofia em torno da I4.0 é ainda uma noção relativa que engloba os sistemas Ciber-Físicos (*CPS),* promovendo a fusão dos mundos físico e virtual, a internet das coisas e dos serviços. Certamente, terá impacto em cada aspeto das organizações ligadas à indústria. Este conceito ainda parece remeter para um futuro longínquo, porém, as organizações já podem traçar o seu plano de transformação, que surge como um plano inevitável para garantir a sua permanência no mercado. Os sistemas *CPS* correspondem a objetos físicos associados a um determinado *software*, com um determinado poder computacional [39]. Na I4.0, mais produtos serão mais inteligentes e possuirão capacidades de autogestão. Por outro lado, os equipamentos industriais se converterão em *CPPS* (*Cyber-Physical Production Systems*), uma subcategoria dos *CPS*, onde as máquinas são geridas com recurso a um *software* autónomo e dinâmico com um elevado poder computacional, autogerindo uma ampla gama de sensores e atuadores. Estes sistemas conhecem o estado, capacidade e os modos de configuração disponíveis, decidindo autonomamente de acordo com os eventos decorrentes da sua operação. Na indústria, a produção em massa tem vindo a dar lugar à customização em massa, onde cada produto possui características únicas definidas pelo cliente final. Neste paradigma, as cadeias de abastecimento possuirão elevada transparência e integração e os fluxos físicos dos produtos serão controlados por plataformas digitais.

Se os desafios das cadeias de abastecimento são altos, os desafios no chão de fábrica não são menores. A combinação dos *CPS* e *CPPS* desencadeará mudanças na produção e controlo industrial. No paradigma I4.0, os *CPPS* são representativos da capacidade produtiva (procura) no chão de fábrica, em que as máquinas e os sistemas interligados podem ajustar as suas operações para maximizar a eficiência e responder à procura ou às necessidades de produção, representados pelos *CPS*, que utilizam dados de mercado, previsões da procura e outras informações para orientar a produção. Os *CPS* procuram garantir que a produção permaneça alinhada com as necessidades reais dos clientes [38] [40]. Assim, o ambiente

produtivo se baseará num sistema multiagente descentralizado com metas estabelecidas, cujas restrições contraditórias conduzirão apenas à execução de operações eficientes. [40]

2.8. Perspetivas futuras do *MES*

A visão da I4.0 orienta as empresas de fabrico a adquirirem elevados níveis de capacidades digitais, com a ambição de seguir o modelo de 'fábrica inteligente'. Devido às capacidades digitais melhoradas das empresas de produção, as operações produtivas podem mais facilmente ser planeadas, implementadas e controladas do que antes através da rastreabilidade, ou seja, a capacidade de traçar o histórico de todos os recursos na produção [41].

2.8.1. Descentralização

A consequência direta da I4.0 para os sistemas de informação centralizados corresponde à extinção dos mesmos [40] no seu estado atual. Em termos práticos, o *MES*, apesar de corresponder a um sistema centralizado, tem dado uma boa resposta aos problemas decorrentes da evolução da produção industrial em termos de desempenho, qualidade e flexibilidade. Porém, as organizações devem-se preparar para lidar adequadamente com os novos desafios desenvolvidos pela I4.0 [40].

A noção de descentralização necessita de ser física, embora lógica, o que implica que um produto inteligente, encarnando um *CPS*, possua a capacidade de se identificar e de se conectar a um sistema fisicamente centralizado, informando devidamente a sua posição e estado. Deste modo, o seu poder de computação pode estar num outro lugar, certamente numa nuvem, o que coloca em questão se o *MES* se trata de um sistema fisicamente centralizado. Portanto, o *MES* ainda é considerado um único sistema, porém, descentralizado com agentes ou objetos inteligentes a representar o chão de fábrica. Um produto inteligente conhece o seu estado, posição, histórico, o seu produto alvo e as alternativas de fluxo existentes. Por outro lado, um equipamento inteligente ou *CPPS* tem conhecimento do plano de manutenção e da sua gama de configurações exequíveis, entre outras propriedades [40].

2.8.2. Integração Vertical

A conformidade, o controlo ou o cumprimento de qualquer outro processo de negócio corporativo é garantido através da integração vertical. Este conceito, quando aplicada ao

MES, refere-se à capacidade de conectar e sincronizar todos os níveis hierárquicos dentro de uma fábrica, desde o chão de fábrica até à gestão de topo. Deste modo, é possível efetuar uma comunicação fluida e eficiente entre diferentes departamentos e sistemas, garantindo que todos estejam alinhados com os objetivos de produção e gestão. A integração vertical permite que o *MES* automatize processos e fluxos de trabalho, o que incrementa a eficiência e minimiza a necessidade de intervenção humana [41]. Com a integração vertical, os dados são analisados em tempo real, com o acesso aos elementos inteligentes *CPS* e *CPPS*, o que facilita e melhora a tomada de decisões, entrando nesta equação o *MES*, que deve ser um sistema modular e logicamente descentralizado para que todas as funcionalidades possam ser consumidas por materiais e equipamentos inteligentes [40].

2.8.3. Conetividade e redes 5G

A conetividade no chão de fábrica pode ser considerada recente. Hoje, as organizações mais facilmente alcançam uma conetividade mais eficiente. Algumas possuem equipamentos com comunicação bidirecional por meio de interfaces, inferindo leituras de sensores, alarmes, relatórios e até selecionar e efetuar um *download* dos procedimentos produtivos. A I4.0 vem democratizar esta conetividade, permitindo que organizações com diferentes níveis de sofisticação consigam ter acesso a esta. Por um lado, as entidades logicamente autónomas do *MES* podem armazenar as coordenadas relativas ao posicionamento dos elementos e mostrá-las em tempo real num mapa interativo. Por outro, no mundo do *IIoT* (*Industrial Internet of Things*), isso traduz-se em *hardware* de baixo custo e um *Lean OS*, como por exemplo, o *Linux Embedded* a correr sobre um *Raspberry Pi*, que se trata de um pequeno computador [42], permitindo uma verdadeira conetividade com equipamentos que não requerem sistemas e interfaces pesados [40]. A velocidade do fluxo de informação também assume relevância no processo, uma vez que as redes *5G* e, no futuro *6G*, proporcionarão velocidades de dados extremamente rápidas e atrasos ou falhas ultrabaixas, permitindo a comunicação em tempo real entre dispositivos e sistemas *MES*. O *5G* corresponde à quinta geração de redes móveis, sendo a mais recente no mercado, e pode permitir atingir velocidades de *download* de até 50 *Gbps* e *upload* de até 10 *Gbps* [43].No futuro, as aplicações móveis serão usadas para controlar todo o processo produtivo. Por outro lado, o acesso a mapas 3D interativos, orientados por sistemas de posicionamento confiáveis, com potenciais cenários de realidade aumentada podem se tornar-se realidade. Este conceito poderá servir eficazmente as atividades de manutenção e de localização de objetos [40].

2.8.4. Computação em nuvem e *Big Data Analytics*

A computação em nuvem e a análise de Megadados (*Big Data Analytics*) constituem um pilar relevante do *MES* do futuro. O *CPS* e o *CPPS* irão conceber colossais quantidades de dados que necessitam de ser armazenados e processados. Isto resulta na visão da fábrica inteligente 4.0 que permitirá a integração de dados de diversas fontes. Deste modo, são necessárias análises avançadas para compreender completamente o desempenho dos processos de fabrico, a qualidade dos produtos e o estado das cadeias de abastecimento.

A análise também permitirá identificar ineficiências com base em dados históricos e posteriormente permitir que ações corretivas ou preventivas sejam executadas. Podem ser efetuadas análises *offline*, utilizando sofisticados processos estatísticos, associados a bases de dados existentes, ou análises em tempo real, para que determinadas ações sejam executadas antes dos dados serem armazenados. Isto requer a utilização de técnicas como o *In-Memory* para o processamento de eventos complexos [40], referente ao armazenamento e processamento de dados diretamente na memória principal (*RAM*) de um computador, em vez de utilizar dispositivos de armazenamento tradicionais, como os discos rígidos [44].

2.8.5. Inteligência Artificial e Previsões de mercado

A inteligência artificial (IA), que consiste num conjunto de tecnologias que permitem aos computadores realizar tarefas que antes exigiam inteligência humana, desempenhará um papel relevante no *MES*, através do aumento da eficiência (com a minimização de desperdícios e otimização de recursos), do aumento da transparência (em que é possível ter uma visibilidade completa de operações e processos), da melhoria na qualidade (com o rastreamento contínuo e controlado), e, por último, da sustentabilidade (com uma melhor gestão de recursos e práticas mais ecológicas). Estes aspetos tornam a integração da IA uma parte essencial da evolução dos *MES*, contribuindo para uma produção mais eficiente e alinhada com os objetivos estratégicos da empresa [45]. O mercado global dos *MES* permanece em ascensão desde a sua criação, com destaque para a América do Norte e para a Ásia-Pacífico. Projeções apontam para um crescimento significativo, com uma taxa de crescimento anual composta (*CAGR*) de 9.2%, atingindo um valor de 23 biliões de dólares até 2029 [46].

3. Metodologias

Com base na revisão da literatura anterior, é possível a perceção dos pontos fundamentais do estudo da implementação de uma metodologia *MES* e a sua aplicação a um caso de estudo exemplificativo de um sistema de gestão de produção *MES*, a ser selecionado no presente capítulo. Para isso, a metodologia presente neste capítulo expressa-se nas seguintes etapas.

- Contextualização do ambiente produtivo que constitui o caso de estudo no presente projeto.
- Definição de um ciclo produtivo de exemplo, expresso por via de um(uns) diagrama(s) do tipo *BPMN*;
- Seleção de uma ferramenta *MES*, dentro das disponíveis do tipo *open-source*;
- Caracterização do sistema escolhido, assim como de todas as ferramentas associadas;
- Desenvolvimento aprofundado do sistema produtivo demonstrativo.

3.1. Contextualização do ambiente produtivo

O caso de estudo usado neste projeto integra um conjunto de equipamentos que formam uma instalação didática de produção que faz parte do LRAFI. Dentro destes equipamentos, a metodologia adotada inclui uma cadeia de transporte circular, que se encontra em destaque na Figura 10, uma máquina *CNC*, alguns robots, dispensadores de paletes e de produto e um armazém automático. No percurso da cadeia de transporte circular, existem três pontos de paragem possível, nomeadamente as estações identificadas com a numeração de 1 a 3, de acordo com o demonstrado na Figura 10. Entrando em maior detalhe e seguindo uma ordem numérica decrescente, no ponto 3, temos a estação da máquina *CNC*, no ponto 2, temos a estação dos dispensadores de palete e produto, assim como o armazém automático. E, por último, no ponto 1, temos o local de entrega do produto final genérico. O circuito é fechado e a circulação da palete com o produto é efetuada através de navetes que as suportam e que permitem o transporte de peças entre estações, através da circulação do tapete. É possível a circulação de diversas navetes em simultâneo no sistema.

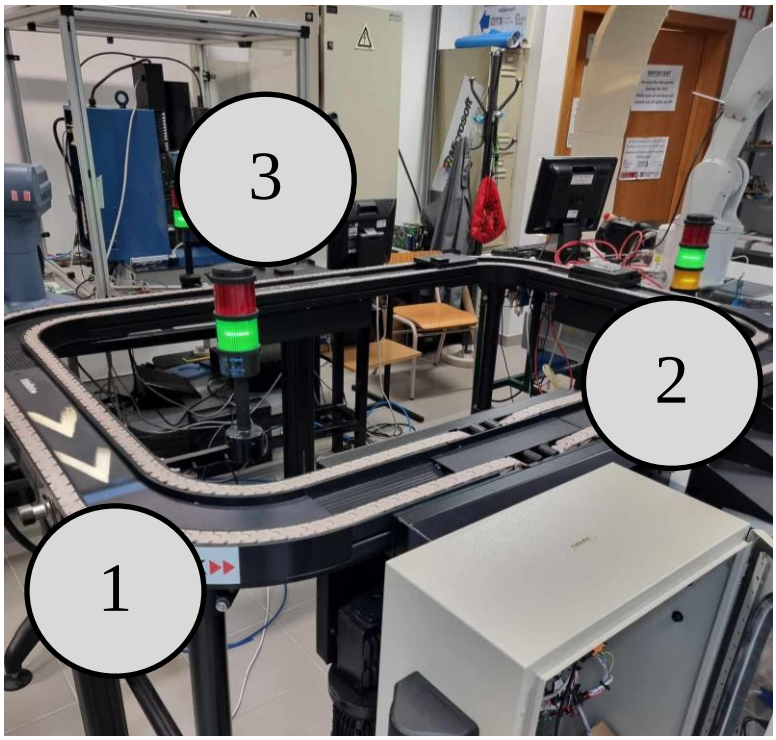

Figura 10.: Layout das máquinas presentes no laboratório e identificação das estações.

As navetes podem efetuar paragens nas estações definidas, em que é possível a colocação das peças nas estações das máquinas. Aos elementos anteriores serão associadas funcionalidades e *skills* que serão exploradas mais intensamente num momento posterior.

3.2. Definição do caso de estudo

Tendo em mente as ferramentas e as limitações anteriores, é possível proceder à implementação de um percurso ou ciclo funcional que visa a simulação de uma linha produtiva. De um modo sucinto, esta fase corresponde a um momento de planeamento do ciclo produtivo, cuja simulação ocorrerá com recurso a uma ferramenta do tipo *MES*. Nesta fase, é crucial a definição de um produto genérico que vai ser submetido a alguns processos que visam a sua transformação, transporte e/ou armazenamento, ou seja, a evolução de uma peça desde o seu estado bruto até ao seu estado final. Portanto, é estabelecido um objetivo final resultante de determinadas etapas produtivas.

Primeiramente, é necessária a definição de módulos funcionais que abranjam as diversas etapas. Neste caso, considera-se que a cada máquina corresponde um módulo. Este ciclo produtivo vai ser associado a um diagrama *BPMN*. Este corresponde a uma linguagem de modelação oriunda da área de processos administrativos, sendo definido como um padrão abrangente para a modelação de processos que deve ser entendido por todos e que também

define a semântica de execução formal que permite a implementação automatizada de processos através dos denominados mecanismos *BPMN*, que além de simplificar a nossa visão sobre o processo, permitem a reunião de todos os elementos necessários num único espaço, o que assegura uma melhor organização dos processos de fabrico [13].

No ciclo produtivo definido, serão utilizados os recursos mencionados na secção 3.1. A cada recurso corresponde um módulo e estão associadas potenciais *skills* que permitem a implementação ontológica do ciclo produtivo em estudo, a seguir ilustrado. No presente projeto, o foco estará no desempenho das funcionalidades correspondentes à cadeia de transporte circular. Para cada equipamento, encontram-se devidamente identificadas as respetivas *skills*, assim como o próprio módulo associado a cada máquina.

- Equipamento 1 - Dispensador de paletes (módulo DP)
    a) Fornece uma palete vazia (‘skill-Palete’).
- Equipamento 2 - Dispensador de produto bruto (módulo PB)
    b) Fornece um produto bruto (‘skill-PB’).
- Equipamento 3 - Robot (módulo R)
    c) Agarra peça (Grip/ Pick) (‘skill-Agarra-Peca’);
    d) Transporta peça (Move) (‘skill-Transporta-Peca’);
    e) Liberta peça (Place) (‘skill-Liberta-Peca’).
- Equipamento 4 - Tapete (módulo T)
    f) Liberta navete – fornece naveta vazia (‘skill-Libertar-Navete’);
    g) Move peça da estação n para a estação j (‘skill-Ir_Para’);
    h) Adia navete (‘skill-Adia-Navete’) – permite à navete prosseguir o circuito, dando uma volta completa e regressa ao mesmo ponto do qual foi atribuída a ordem.
- Equipamento 5 - Máquina *CNC* (módulo *CNC*)
    i) Recebe peça (‘skill-Recebe-Peca’);
    j) Maquina peça (‘skill-Maquina-Peca’);
    k) Devolve peça (‘skill-Devolve-CNC-Peca’).

- Equipamento 6 - Armazém Automático (módulo AA)
    - l) Arruma peça (‘skill-Arruma-Peca’);
    - m) Devolve peça (‘skill-Devolve-AA-Peca’).

A disposição de equipamentos a utilizar no procedimento encontra-se esquematizada na Figura 11, assim como os pontos de paragem nas estações um a três.

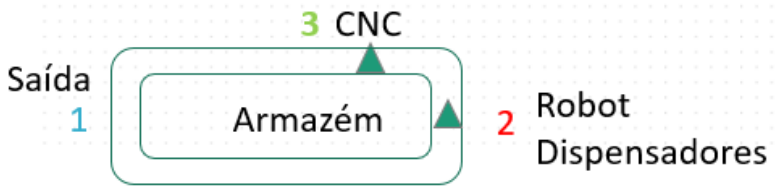

Figura 11.: Layout ou Disposição de equipamentos no laboratório de automação.

O ciclo produtivo definido remete as seguintes etapas produtivas.

1º) O tapete fornece uma navete vazia (*Skill* f - ‘skill-Libertar-Navete’);

2º) Os dispensadores de paletes e produto bruto fornecem novas palete e produto bruto, respetivamente (*skills* a - ‘skill-Palete’ e b - ‘skill-PB’, respetivamente).

3º) Ocorre a colocação da palete com produto bruto no tapete (ponto 2)

- O robô agarra a palete (*Skill* c - ‘skill-Agarra-Peca’);
- O robô move a palete (*Skill* d – ‘skill-Transporta-Peca’);
- O robô larga palete no ponto 2 do ciclo (*Skill* e - ‘skill-Liberta-Peca’).

4º) O tapete desloca a peça do ponto 2 para o ponto 3 (*Skill* g - ‘skill-Ir_Para’);

5º) O robô coloca a palete com a peça do ponto 3 do tapete para a estação da CNC.

- O robô agarra palete do tapete (*Skill* c - ‘skill-Agarra-Peca’);
- O robô move a palete (*Skill* d - skill-Transporta-Peca’);
- O robot larga a palete na estação da CNC (*Skill* e - ‘skill-Liberta-Peca’).

6º) Ocorre a receção, maquinação e devolução da peça.

- A máquina CNC recebe a peça (*Skill* i - ‘skill-Recebe-Peca’);
- A máquina CNC maquina a peça (*Skill* j - ‘skill-Maquina-Peca’);
- A máquina CNC devolve a peça (*Skill* k - ‘skill-Devolve-Peca’).

7º) A peça é colocada da estação do ponto 3, de volta para o tapete.

- O robô agarra a palete que se encontra na estação (*Skill* c - ‘skill-Agarra-Peca’);
- O robô move a palete da estação para o tapete (*Skill* d - ‘skill-Transporta-Peca’);
- O robô larga a palete, no tapete (*Skill* e - ‘skill-Liberta-Peca’).

8º) O tapete desloca-se do ponto 3 para o ponto 2 (*Skill* g - 'skill-Ir_Para');

9º) O robô coloca a palete na estação do ponto 2

- O robô agarra a palete do tapete (*Skill* c - 'skill-Agarra-Peca');
- O robô move a palete (*Skill* d – 'skill-Transporta-Peca'));
- O robot larga a palete na estação do ponto 2 (*Skill* e - 'skill-Liberta-Peca').

10º) A peça é arrumada no armazém automático (*Skill* l - 'skill-Arruma-Peca').

11º) A peça é devolvida à estação 2 (*Skill* m - 'skill-Devolve-Peca').

12º) O robô coloca a palete da estação do ponto 2 para o tapete

- O robô agarra a palete da estação do ponto 2 (*Skill* c - 'skill-Agarra-Peca');
- O robô move a palete (*Skill* d – 'skill-Transporta-Peca');
- O robô larga palete no tapete (*Skill* e - 'skill-Liberta-Peca').

13º) O tapete desloca-se do ponto 2 para o ponto 1 (*Skill* g - 'skill-Ir_Para');

14º) O robô coloca a palete do tapete para a estação do ponto 1

- O robô agarra a palete do tapete (*Skill* c - 'skill-Agarra-Peca');
- O robô move a palete (*Skill* d – 'skill-Transporta-Peca');
- O robô larga palete na estação do ponto 1 (*Skill* e - 'skill-Liberta-Peca').

15º) A peça é submetida para entrega.

O *BPMN* bruto referente ao processo anterior encontra-se demonstrado na Figura 12, em que cada *skill* ou conjunto de *skills* se encontram associados a um bloco no diagrama. Adicionalmente, em cada conjunto de blocos está identificado o ponto de possível paragem do ciclo produtivo em que se situa, de um a três. O ciclo foi desenvolvido no *Camunda Modeler*, que se trata de uma ferramenta gratuita e *open-source* de construção e simulação de diagrama de blocos, correspondendo a uma ferramenta de *design* visual que permite, com o mínimo de código possível, modelar, conectar e preparar abertamente processos de negócios do tipo *BPMN* e, posteriormente, simulá-los.

O *Camunda Modeler* faz parte da plataforma *Camunda* que permite a modelação de processos e a automatização de processos e decisões [47].

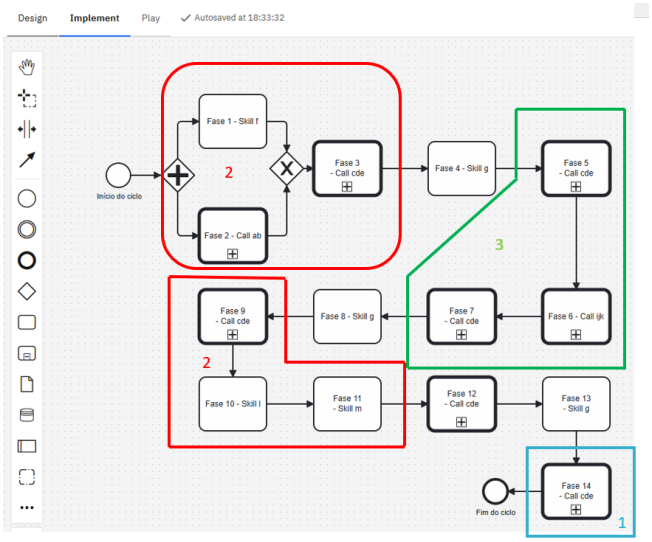

Figura 12.: Diagrama *BPMN* com a demonstração do ciclo produtivo definido.

3.3. Seleção de um software *MES*

Numa primeira etapa, vamos definir uma ferramenta *MES* a utilizar. Para a seleção, foi proposto o requisito fundamental de este corresponder a um sistema de código aberto (*open-source)*, para que fosse possível a manipulação e alteração do código, se necessário. Um *software* de código aberto possui as vantagens de estar disponível para *download*, instalação, edição e partilha, juntamente com seu código-fonte, por qualquer pessoa ou organização, o que significa que está em constantes atualizações. Assim, é possível explorar o código para entender o produto e até melhorá-lo, se necessário, de modo a atender às necessidades operacionais em constante mudança de um modo melhor. Este foi à partida revisto, melhorado e testado por desenvolvedores, codificadores, utilizadores e testadores. Este fenómeno resulta possivelmente numa rápida correção de falhas, uma vez serem mais propensas a serem encontradas. Num contexto de fabricação, dar a um desenvolvedor interno acesso ao código-fonte significa que ele pode manipular o código e fazer alterações conforme seja necessário, em vez de esperar que o fornecedor disponibilize atualizações de *software* que podem ou não ser relevantes ao nosso tipo de atualização. Portanto, este tipo de *software* pode permitir a personalização, o que implica a necessidade de uma formação mais especializada e, por outro lado, a segurança é uma preocupação preponderante [48].

À medida que a concorrência no setor se intensifica, o sucesso empresarial dependerá, mais do que nunca, da velocidade e confiabilidade dos dados, análises e outras tecnologias de *IoT*. O *software* de código aberto já desempenha um papel de destaque neste âmbito.

Encontrar a melhor combinação de *software* e de código aberto será uma parte fundamental para garantir a competitividade dos negócios para a próxima década e além deste patamar [49]. Um outro requisito remete para a linguagem de programação, propondo-se dar prioridade à linguagem *Java,* uma vez se tratar de uma das mais populares e amplamente utilizadas no desenvolvimento de *software.* A *Java* corresponde a uma linguagem orientada a objetos, o que significa que ela utiliza conceitos como classes e objetos para organizar o código *[50]*. Deste modo, inicia-se a procura por uma ferramenta viável para o caso de estudo, que deve ser o mais intuitivo, o menos complexo possível, ter obtido atualizações recentes, à data da pesquisa, ter procedimentos de utilização e, preferencialmente que permita a aplicação de uma ferramenta *MES* com base na descrição semântica dos elementos produtivos, através de módulos e *skills*.

Após uma pesquisa por sistemas com as especificações previamente estabelecidas, especialmente no sítio *GitHub*, uma plataforma que se dedica à partilha de *softwares* desenvolvidos e atualizados por milhões de desenvolvedores e organizações. [51], verifica-se que se torna complexo este processo de procura, uma vez que existem muitas soluções disponíveis, nomeadamente de código aberto, contudo correspondem a soluções maioritariamente amadoras, sem procedimentos de utilização ou com pouquíssima informação disponibilizada, o que inviabiliza uma grande parte das ferramentas encontradas. Com base nos critérios anteriores, das cerca de 20 soluções encontradas e analisadas, foi possível estabelecer um Top 5 de soluções, que se encontram demonstradas na Tabela 1. As restantes soluções foram descartadas, maioritariamente por falta de procedimentos de utilização. Cada uma das soluções presentes foram analisadas, através de um processo de hierarquia analítica (AHP), ou seja, passa pela atribuição de um grau de aptidão para as diversas restrições definidas em que o *software* que obtiver maior pontuação corresponderá à solução vencedora. Pode ser considerado um processo subjetivo. Porém pode evidenciar potenciais características relevantes das ferramentas para o projeto. Para estas soluções, encontra-se especificada a data da última atualização, no momento da última visualização, referente a abril de 2022, e respetivas linguagens de programação.

Tabela 1.: Soluções de *software* de código aberto encontradas.

ID	Nome do software	Última Atualização	Linguagens
1	WEM (WebErpMesv2)	09/04/2022	Blade (55.4%); PHP (44.1%)
2	SkillMEx	29/03/2022	TypeScripyt (77,3%); HTML (18.1%); SCSS (3.6%); JavaScript (1%)
3	MES4U	29/09/2020	Vue (39.9%); PLpgSQL (24.9%); Java (19.2%); JavaScript (10.3%)
4	qcadoo MES	02/04/2022	Java (58%); PLpgSQL (38,7%); Java Sc
5	Batur123	24/03/2022	C# (100%)

Foram analisadas as cinco soluções anteriores, com vista a definir um *software* o mais compatível possível com os objetivos definidos. A informação relativa a cada um dos programas foi obtida por análise da documentação disponível com cada um deles, já que, no tempo de execução deste trabalho, não seria possível a implementação completa de todas as hipóteses. Na Tabela 2, define-se o grau de aptidão de cada uma das soluções aparente para cada uma das restrições definidas, sendo relacionadas com a Tabela 1 pelo campo '*ID'*. O grau de aptidão encontra-se definido na Tabela 3. Tornou-se complexo o processo de atribuição do grau de aptidão a cada uma das restrições, visto que a informação disponível não permite determinar a viabilidade de determinadas funcionalidades.

Tabela 2.: Grau de aptidão de cada *software* às restrições estabelecidas.

	ID Software				
Restrições	1	2	3	4	5
Grau Aparente de Desenvolvimento	4	4	3	4	3
Possui procedimentos de utilização	NE	3	4	4	1
Intuitivo	4	2	4	3	2
Painel de status da fabricação em tempo real	3	2	0	3	NE
Gestão de Pessoas / utilizadores e funções	3	3	2	2	2
Gestão de inventário e stocks	NE	3	1	NE	NE
Sistema de relatórios	NE	3	0	NE	NE
Gestão de produtos defeituosos	NE	3	2	2	NE
Possibilidade de gestão de etiquetas dos produtos	NE	NE	4	NE	NE
Adição de clientes, qualidade, produtos, ordens de fabrico;	4	3	3	3	4
Gestão das *BOM* (Lista de Materiais)	4	3	3	3	NE
Representação ontológica relativa à estrutura das máquinas	NE	4	NE	NE	NE
Definição de capacidades produtivas, módulos e *skills*	4	5	4	4	4
Interação com *KPI's* e *Kanban*	NE	NE	NE	NE	NE
Gestão da Montagem de Produtos	NE	NE	NE	NE	NE
Gestão da Inspeção de produtos em produção (OK/NOK)	NE	NE	NE	NE	NE
Possibilidade de acoplar com um *ERP*	5	NE	5	4	NE
OPINIÃO PESSOAL	10	4	6	8	2
Pontuação	41	42	41	40	18

Tabela 3.: Grau de aptidão estabelecido.

Legenda	Grau
5	Muito Bom
4	Bom
3	Razoável
2	Insatisfatório
1	Péssimo
NE	Não Especificado

Após o processo anterior e de acordo com a Tabela 2, é possível definir o *SkillMex* como a solução vencedora e que melhor corresponderá aos objetivos previamente estabelecidos. Um fator relevante desta ferramenta é que esta é a única das analisadas que apresenta esta nova abordagem de *skills*, enquanto as restantes correspondem basicamente à execução de

um *MES* tradicional, embora seja inconclusivo se as outras ferramentas possuem também a abordagem de *skills*. Por isso, iremos optar por esta via de execução.

3.4. Caracterização do *SkillMex*

Após a definição de uma metodologia e respetivo *software*, procede-se à etapa seguinte. É crucial o conhecimento das bases da ferramenta, como decorreu a sua criação e contexto. Basicamente, deve-se conhecer todo o ambiente do *SkillMex* e demais ferramentas, tratando-se de um passo relevante e inevitável para podermos obter resultados viáveis.

3.4.1. **Desenvolvimento do SkillMex: Contexto e Ferramentas Essenciais**

O *SkillMex* corresponde a uma plataforma de gestão da produção que funciona através de módulos produtivos que possuem uma descrição semântica das suas funções, sob a forma de capacidades e *skills* (conceitos abordados no tópico 2.3). Os módulos podem ser registados com as respetivas capacidades e *skills* e posteriormente editados. As ontologias correspondem a um modo de desenvolver modelos semânticos e expressar termos e relações entre estes, de forma que possa ser interpretável pela máquina. Portanto, as ontologias permitem a implementação de meios sofisticados de interagir com informações, como, por exemplo, consultas mais complexas, verificação de consistência de modelos, incitar novas informações a partir de informações pré-existentes e aplicar regras especificas aos modelos. Na modelação de módulos, é necessária a interpretação por via de três aspetos.

- Estrutura da máquina – em que é possível a representação de informações sobre a estrutura dos equipamentos e respetivos componentes;
- Capacidades – Estas representam os processos que um módulo pode realizar. Os processos de fabricação podem ser especificados segundo os padrões da indústria *DIN* 8580 e *VDI* 2860 [52]. A norma alemã *DIN 8580*, aplicada internacionalmente aos processos produtivos, organiza os processos em seis grupos principais: a conformação primária (produção de peças a partir de materiais sem forma definida); a transformação (modificação da forma de materiais sólidos); a divisão de materiais em partes menores; a fusão de duas ou mais peças; a modificação das características internas do material; e o revestimento (aplicação de camadas de material sobre uma superfície) [53]. A norma *VDI* 2860, desenvolvida pela Associação de Engenheiros Alemães (*VDI*), permite a definição de termos e conceitos relativos à comunicação,

ao utilizar objetos e materiais e ao desenvolvimento de sistemas automatizados, como robôs industriais [54]. Os processos manuais e os automáticos podem ser expressos como capacidades.

- *Skills* – Estas descrevem os mecanismos de interação fornecidos por um módulo para usar automaticamente as suas funcionalidades. Cada *skill* fornece uma descrição de interface para que possa ser invocada e monitorizada a partir de outros sistemas, por exemplo, sistemas de execução da produção (*MES*).

Clarificando, os recursos correspondem à descrição das funções da máquina e podem ser oferecidos por várias máquinas. Uma máquina pode fornecer uma ou mais *skills* para executar uma capacidade. As *skills* são exclusivas de uma máquina. As capacidades são modeladas como operadores de processo.

Tendo em conta uma abordagem para a modelação de funções do sistema, temos o conceito de arquitetura funcional, que corresponde à modelação de sistemas através de funções. Esta abordagem possibilita uma melhoria do entendimento sobre o sistema para *stakeholders* com um *know-how* diferenciado devido à sua forma simples de representação. Este tipo de abordagem é, portanto, particularmente adequada para modelar os *CPPS* dinâmicos, uma vez que os detalhes específicos sobre os componentes dos sistemas envolvidos no *CPPS* não precisam ser conhecidos para os descrever, nomeadamente em termos de entradas e saídas. A linguagem selecionada para o efeito consiste na linguagem de modelação de sistemas *SysML*, porém esta não considera funções, o que leva à necessidade de uma extensão do sistema. O *SysML* consiste numa linguagem de modelação aplicada à engenharia de sistemas, dando suporte ao *design*, análise e verificação de sistemas complexos que podem incluir componentes de *software* e *hardware* [55]. Uma abordagem *SysML* para funções de modelação requer modelos *CPPS* para os quais os módulos deverão contribuir.

Por outro lado, a ferramenta selecionada assume uma abordagem de engenharia baseada em *skills*, vista como um tipo de descrição de mais alto nível sendo decompostas em ações menos complexas para a sua execução. Esta baseia-se num modelo de *skill* semântica na forma de uma ontologia baseada em diferentes padrões da indústria, cujas *skills* podem ser executadas através do *OPC UA* (ligação neutra) ou por via *web*. Neste contexto, surge o sistema *MES*, nomeadamente o *SkillMex*, que promete efetuar várias tarefas, nomeadamente registar e executar *skills*, bem como processos de produção completos compostos por

diversas *skills* de ordem de complexidade inferior, sob a forma de diagramas *BPMN* [52]. Por outro lado, o *SkillMex* baseia-se num método *MBSE* (*Model-Based Systems Engineering*), que se trata de uma metodologia que utiliza a modelação e simulação de modelos para apoiar o ciclo de vida de um sistema, desde a conceção e o *design* até as atividades de verificação e validação finais [56]. O *MBSE* é demonstrado na Figura 13.

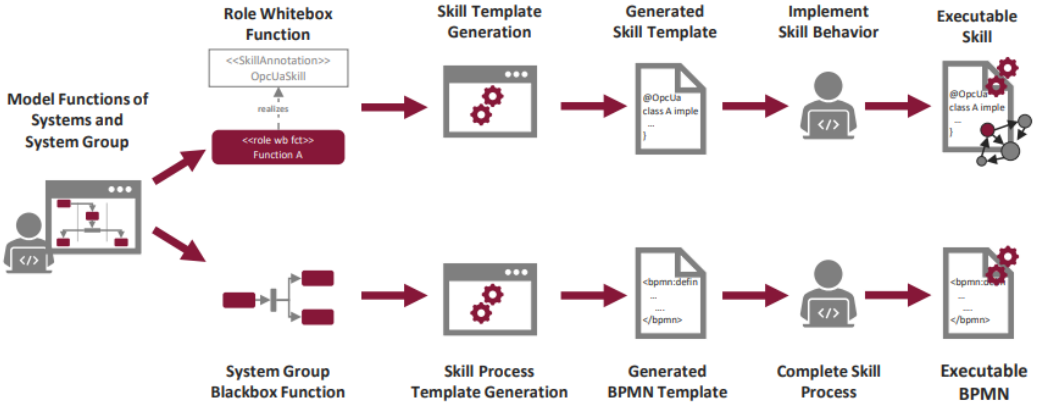

Figura 13.: Visão geral para gerar um código executável a partir do *MBSE* aplicado ao *SkillMex* [13].

Neste *software*, a modelação é conseguida a partir da conjugação de dois tipos de módulos funcionais, porém, é relevante clarificar alguns conceitos previamente, nomeadamente os conceitos de *SG blackbox e SG whitebox*. A nomenclatura *SG* refere-se a grupo de sistema. O *SG blackbox* (grupo de sistemas de caixa preta) são frequentemente utilizados para garantir a segurança, eficiência e confiabilidade em ambientes de produção automatizados, realizando tarefas específicas sem a intervenção humana direta ou a compreensão detalhada dos seus processos internos. O sistema interno não é visível ou acessível, apenas as suas entradas e saídas podem ser visualizadas. Por outro lado, o *SG whitebox* (grupo de sistemas de caixa branca) é transparente e acessível, em que os seus utilizadores podem visualizar e compreender o sistema interno, tendo acesso a dados produtivos, análises detalhadas e lógicas de controlo [57]. A abordagem de construção de um sistema *CPPS* com esta ferramenta contém os seis passos seguintes [39].

1) Modelação por objetivos – corresponde à fase de modelação dos objetivos pré-definidos, ou seja, de forma a obter uma base adequada para derivar as funções que descrevem as funcionalidades;

2) Modelação das funções *SG blackbox* – Estas funções surgem dos objetivos definidos. Como a *SysML* distingue entre elementos estruturais e comportamentais e as funções *SG blackbox* são especificações comportamentais, um elemento estrutural inicial do *SG* é criado em paralelo a esta etapa, cujas funções são atribuídas com um comportamento próprio. As metas modeladas na primeira etapa são vinculadas às funções *SG blackbox* modeladas no final da segunda etapa para garantir que todas as metas do grupo sejam consideradas pelas funções *SG blackbox*.

3) Atribuição de papéis *SG* e modelação de funções *SG whitebox* – Decorre a especificação real do comportamento das funções *SG blackbox*. É desenvolvido um diagrama de atividades para cada função, onde constam as funções *SG whitebox* que são modeladas. Estas podem ser conectadas entre si por fluxos de controlo para definir uma sequência de execução ou por fluxos de objetos para trocar informação a ser processada entre eles por meio de entradas e saídas. As funções *SG whitebox* modeladas são atribuídas a *placeholders* estruturais que podem ser assumidos por *CPSs* que são responsáveis pela execução da função *SG whitebox*. Os *placeholders* são utilizados em programação e *design* para reservar um espaço temporário que será preenchido posteriormente com dados reais [58] e são adicionados ao *SG blackbox* para expressar quais os papéis individuais que o *SG* deve consistir.

4) Descrição dos portos *Cross-Role* – Estes portos correspondem a pontos de troca de informação. Nesta fase, é crucial a descrição das interfaces (portos *Cross-Role*) dos papéis, de modo que as entradas e saídas das funções possam permitir troca de informação. Para isso, todos os pontos em que ocorre a troca de objetos entre funções são identificados através do uso dos diagramas de atividades. Para cada tipo de informação trocada, um porto separado é adicionado ao elemento de estrutura da respetiva função.

5) Ligação das funções *SG whitebox* aos portos *Cross-Role* - As conexões entre *SG whitebox* de papéis pertencentes a diferentes objetivos são substituídas por elementos para enviar ou receber sinais. Os sinais podem ser vinculados aos portos criados na quarta etapa, realizando assim a troca funcional entre vários papéis.

6) Modelação e agregação dos estados das funções – Ocorre uma descrição dos estados individuais de cada função, de modo a considerar as dependências funcionais baseadas no estado. As funções *SG whitebox* criadas na quarta etapa podem ser vinculadas diretamente aos estados modelados em diagramas de estado. Além disso, outras interfaces podem servir para trocar sinais entre funções que acionam

transições de estado, realizadas pelas funções de colaboração, e podem ser agregadas em estados de *SG* [13]. As funções de colaboração fazem a coordenação e a comunicação entre os componentes do sistema [59].

Depois deste procedimento de modelação de funcionalidades dos sistemas e *SG*, que especifica as contribuições dos diversos *CPSs*, especifica-se e gera-se o código referente às *skills* para cada funcionalidade produzida. É relevante efetuar a distinção entre o desenvolvimento de *skills* individuais de um sistema e o desenvolvimento de uma sequência de processos que consiste em *skills* individuais e é posteriormente executada de forma colaborativa pelo *SG*.

1) Geração de modelos de *skills* individuais

Após modelar as funções de sistema conforme descrito na abordagem *SysML*, estas devem ser implementadas como *skills*. Para cada uma, os autores do *SkillMex* desenvolveram um *framework Java*, ou seja, uma plataforma que facilita o desenvolvimento de aplicações em *Java* [60], denominado *SkillUp*, para o seu desenvolvimento simplificado. Com o *SkillUp*, um desenvolvedor apenas necessita de programar o comportamento real da *skill*. Trata-se de uma tarefa que também seria necessária com uma abordagem de controlo convencional. Deste modo, para chamar as *skills* e a descrição ontológica obrigatória são criados automaticamente, uma máquina de estados, um servidor *OPC UA* ou servidor web para invocar as *skills* e a descrição ontológica.

O *OPC UA* permite constituir um padrão para a permuta de dados entre o chão de fábrica e a gestão de topo. Este protocolo possui a capacidade de fornecer todas as informações de um dispositivo ou processo industrial numa única entidade semântica, bem como simplificar a troca de dados durante a operação do equipamento. Cada entidade *OPC UA* contém um modelo de informação, conhecido como *Address Space [13]*, que é construído segundo um paradigma orientado a objetos que descreve dispositivos ou componentes industriais reais e as suas propriedades, como componentes mecânicos, robôs ou células de fabrico completas em diferentes níveis de detalhe [61].

No *SkillMex*, este processo envolve a criação de uma classe *Java* com anotações que precisam ser configuradas manualmente. Uma anotação obrigatória que define uma classe como uma *skill* corresponde ao campo '@Skill'. Este é utilizado para especificar a implementação automatizada da interface da *skill*, por via *OPC UA* ou *webservice*, bem

como associar a *skill* a um módulo, ao qual pertence. O registo de *skills* e módulos é efetuado através do *runtime* do *SkillUp, que* procura classes anotadas com '@Skill' para registar *skills* assim que forem colocadas numa localização pré-definida no sistema de ficheiros do computador hospedeiro. As anotações '@SkillParameter' e '@SkillOutput' podem ser usadas para especificar os parâmetros e saídas, respetivamente, dentro da *skill.* A anotação '@StateMachine' cria automaticamente uma máquina de estados interna para cada *skill* contendo os estados iniciar, executar, abortar, entre outros. Um exemplo da implementação genérica é especificado na Figura 14.

Com os conceitos anteriores, criou-se uma ligação definida entre uma abordagem de engenharia baseada num modelo para criar funções com uma abordagem baseada em *skills* para implementá-las posteriormente. Previamente à geração de código, é criado um diagrama de classes adicional, ou seja, um *deployment diagram*, que mantém a separação da fase de engenharia da implementação real. A este diagrama são adicionadas as funções *SG whitebox* a serem implementadas.

```
@Skill(type=OpcUaSkillType.class, moduleIri="
    https://hsu-hh.de/modules#Module1")
public class AdditionSkill {

    @SkillParameter
    int a,b;

    @SkillOutput
    int result;

    @StateMachine
    Isa88StateMachine stateMachine;

    @Execute
    public void aPlusB() {
        this.result = this.a + this.b;
    }
}
```

Figura 14.: Exemplo de anotações usadas para criar uma *skill* [13]**.**

Com uma interface *UML* adicional do *SkillAnnotation,* um pacote desenvolvido pelos autores da ferramenta, toda a informação das anotações do universo *SkillMex*, como '@Skill', '@SkillParameter' e o '@SkillOutput' são reconhecidos pelo *SkillMex*. Além disso, a anotação '@StateMachine' é adicionada automaticamente através da definição da variável 'stateMachine*'*. Adicionalmente, para cada estado da máquina de estados, é produzido um método com o nome do estado. Isso resulta num modelo de classe de *skill* completo, de modo que um desenvolvedor ou utilizador apenas necessita executar os métodos necessários. Após esta implementação e a conclusão da *skill*, ela pode ser executada através do *runtime* do *SkillUp.*

2) Síntese de processos de *skills*

Os autores do *SkillMex* desenvolveram um conceito para modelar e executar processos ou sequências de *skills*, utilizando o *BPMN*. O *BPMN* pode ser utilizado para processos de negócios, mas também no âmbito de fabricação, para flexibilizar as organizações. Nos processos de *skills*, os autores usaram as *ServiceTasks* (tarefas de serviço), que correspondem a um elemento *BPMN* que permite acionar um serviço *web* ou *OPC UA* para uma classe criada. Posteriormente, é necessário correr o *SkillExecutor*, que executa as *skills* através da ferramenta *Camunda*. Por outro lado, foi desenvolvida uma abordagem automatizada para transferir os elementos de um diagrama de atividades para um processo *BPMN*. Como ambos são semelhantes em termos da semântica dos seus elementos a correspondência entre estes diagramas é bastante direta.

O gerador de código *BPMN* desenvolvido corresponde a um algoritmo codificado em linguagem de programação *C#* com classes para todos os elementos de um diagrama de atividades *UML*. Ao invocar o gerador de código, para cada elemento de um diagrama de atividades selecionado, uma instância da classe *C#* correspondente é criada. Em seguida, todo o diagrama é serializado, com todos os elementos. Estes são agrupados num processo *BPMN*. Todo este processo pode ser importado para o *SkillMex*. Posteriormente, os valores dos parâmetros de todas as skills precisam ser adicionadas previamente à execução do processo *BPMN* pelo *SkillMex* [13].

No *SkillMex*, podem ser associadas tarefas aos blocos do *BPMN*. No caso da interação humana, esta pode ser modelada através da definição de uma tarefa de utilizador ou uma tarefa manual. Enquanto a tarefa de utilizador é rastreada por um mecanismo *BPMN*, as tarefas manuais devem ser executadas sem um mecanismo. A automação de processos pode ser obtida por meio de *Script Tasks*, *Business Rule Tasks* ou *Service Tasks*. As *Script Tasks* são utilizadas para implementar funcionalidades menores que são executadas diretamente por um mecanismo *BPMN*. As *Business Rule Tasks* definem regras que são verificadas por um mecanismo (*engine*) independente e externo. As *Service Tasks* oferecem o maior potencial pois permitem atribuir uma tarefa a uma aplicação externa que se encarregará de a executar. Assim que a execução de um processo atinge uma *Service Task*, a aplicação externa referenciada pela mesma é invocada. Assim que esse mecanismo externo finalizar a sua execução, o mecanismo *BPMN* continua a execução do processo real [13].

3.4.2. **Funcionamento do ambiente *SkillMex***

Numa primeira instância, é fundamental a instalação da ferramenta *SkillMex*. O procedimento de instalação adotado no presente projeto consta em anexo com maior detalhe dada a complexidade do mesmo. Seguindo o procedimento de instalação descrito no sítio *GitHub* da ferramenta, são necessários requisitos prévios à instalação. Primeiramente instala-se uma versão *LTS (Long Term Support)* do *Node.js*, uma vez que o *SkillMex* é construído utilizando o *NestJS* e *Angular*, que dependem da instalação daquela ferramenta [62]. As versões *LTS* do *Node.js* remetem para versões que recebem suporte a longo prazo, garantindo atualizações de segurança e correções de *bugs* críticas por um período máximo de 30 meses [63]. Nesta etapa, é crucial clarificar o conceito de *npm*, que corresponde a um *software* de gestão de pacotes e ferramentas de projeto que utiliza a linguagem de programação *JavaScript*. O pacote *npm* corresponde por omissão ao pacote de gestão de ferramentas utilizado no ambiente *Node.js* [64].

Por sua vez, procedeu-se à instalação do *GraphDB*, que permite a criação de um ou mais repositórios para o projeto, permitindo a utilização de ontologias. Um repositório consiste num local de armazenamento de dados que são organizados para facilmente serem acedidos e geridos [65]. Após a instalação e iniciação, verifica-se a necessidade da criação do repositório inicial, que o *SkillMex* possa usar (No atual estado de desenvolvimento do *SkillMex*, é obrigatório usar, para este fim, o *ID* '*test-repo*'). A gestão do repositório pode ser efetuada no *SkillMex*. A Figura 15 evidencia o repositório criado por esta via.

Após a instalação dos requisitos anteriores, procedeu-se ao *download* da pasta referente ao *master* do *SkillMex* (que, numa fase mais tardia foi renomeado para *CaSkade-MES)*, para que se possa colocar a correr o *backend* e o *frontend* do *software*, caracterizados anteriormente no tópico 2.5.3. Após a abertura dos comandos *cmd*, localizados no *backend* e no *frontend*, o que implica a abertura de duas janelas, e após o procedimento de resolução de problemas e vulnerabilidades, podemos colocar ambos em funcionamento, através do comando '*npm run start:dev'* que, após a verificação e implementação do código, carrega uma página *web* referente à interface do *SkillMex*. Após este processo, a janela aberta pelo *frontend* agora encontra-se em *watch-mode*, o que indica que a ferramenta entra num ciclo em que observa o sistema de ficheiros quanto a alterações. Sempre que uma alteração é detetada, a tarefa é teoricamente executada novamente para atualizar a sua saída.

Este processo acelera o desenvolvimento porque a reconstrução ocorre automaticamente sempre que se guarda um ficheiro e a tarefa pode beneficiar do *cache* em memória uma vez que o processo nunca termina, encontrando-se sempre atualizado. Após ter o *backend* e o *frontend* a correr, numa primeira etapa, é possível a criação e registo de um utilizador daquele *software*, sendo então procedido o registo, como demonstra a Figura 15. Este registo de utilizador é, na fase de desenvolvimento em que está este *software*, irrelevante, uma vez que os dados de acesso, uma vez introduzidos, não voltam a ser solicitados em nenhum dos acessos futuros. Além disso, após entrar na aplicação, o nome de utilizador que surge corresponde ao nome do autor do *SkillMex*.

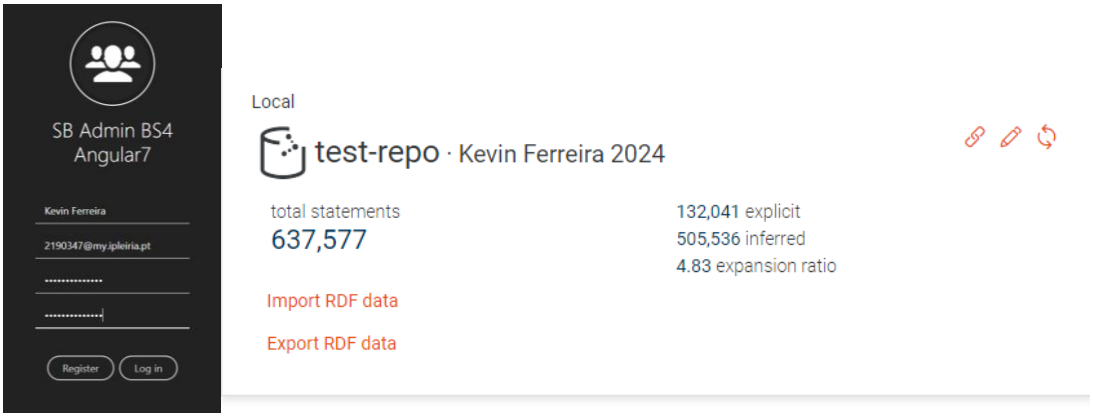

Figura 15.: Registo de utilizador no *SkillMex* (à esquerda) e repositório criado no *GraphDB* (à direita).

Após o registo, a página reencaminha para a janela que se verifica na Figura 16, como resultado do *frontend* instalado. Nesta janela *web* aberta, após a colocação do *frontend* e *backend* a correr, verificamos que o menu lateral é composto pelos seguintes campos.

- *Dashboard* – local onde surge uma visão geral dos registos efetuados no *SkillMex* e a exibição de mensagens decorrentes de alterações.
- *Module Management* – onde surgem os módulos registados e as respetivas *skills* e capacidades, podendo ser efetuada uma gestão geral.
- *Skill-Overview* – onde constam as *skills* individuais registadas, quer estejam associadas ou não a um módulo.
- *Manufacturability Check* – aparentemente inoperacional na versão atual, seria um local em que seria possível efetuar uma gestão da produção, adicionar o modelo 3D de uma determinada peça, limitar os seus processos ou parâmetros. É possível adicionar o nome do utilizador e da empresa, além de um e-mail, para gestão posterior.

- *Production Processes* – permite a criação manual de uma sequência de processos ou fazer o download de um modelo *BPMN* e colocá-lo a correr.
- *Graph Visualization* – surge como um local que possibilita a visualização mais geral dos módulos registados no *SkillMex*, através de balões devidamente identificados, permitindo mover esses balões.
- *SkillMex Configuration* – corresponde a um local em que é possível alterar o utilizador e selecionar ou alterar o devido repositório. O repositório é definido automaticamente quando o *Graph DB* se encontra a correr.

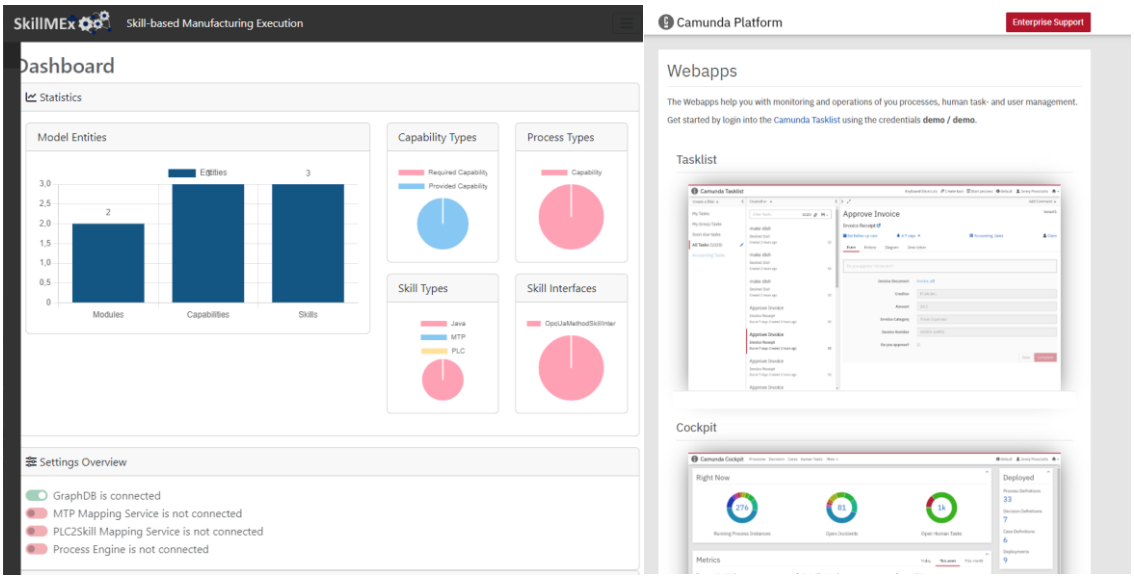

Figura 16.: Página principal do *SkillMex* que surge após a introdução dos dados do utilizador (à esquerda) e página de indicação de abertura do *camunda* (à direita).

Um sistema paralelo, o *SkillUp*, deve ser utilizado, para a criação e edição de *skills* de uma forma mais cómoda e prática. Esta ferramenta minimiza o trabalho referente à criação de uma *skill*, como mencionado previamente. Pela via manual, é necessário fornecer uma tecnologia de interface para invocar o comportamento de uma *skill* (quer seja um servidor *OPC UA* ou um servidor *web*) e ainda criar uma ontologia bastante extensa e complexa para descrever tudo isto. Porém, nunca se lida diretamente com o *SkillUp*, mas sim com o *runtime* do *SkillUp* que, com a colocação dos ficheiros de módulos e/ou *skills* para o interior da pasta 'include' do *runtime*, permite ao *SkillUp* automaticamente registar os ficheiros incluidos nesta página, surgindo posteriormente na página do *SkillMex*. Por fim, de modo a possibilitar a utilização e manipulação de diagramas do tipo *BPMN*, é necessária a instalação do

camunda-bpmn-tomcat[5], que posteriormente deve ser ativado com o a colocação do ficheiro '*start-camunda.bat*' a correr. O *SkillMex* deve detetar o *camunda-run-tomcat* e ativar automaticamente (é visível no *dashboard* do *SkillMex*, surgindo o elemento '*Process Engine'* como conectado e a verde). Após a abertura deve surgir uma página *web* da plataforma, como demonstrado na Figura 16.

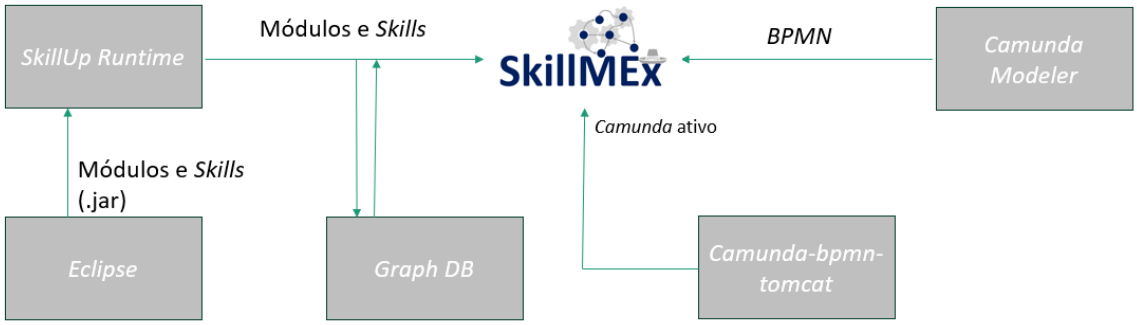

Figura 17.: Esquema geral referente à interação entre o SkillMex e as ferramentas necessárias para o seu funcionamento.

Na Figura 17, encontram-se interligadas todas as ferramentas, associadas ao funcionamento do *SkillMex*, incluindo as interações com as ferramentas *SkillUp*, *GraphDB* e *Camunda Modeler*. As versões das ferramentas são uma questão crucial para colocar em funcionamento a aplicação. As versões adotadas no projeto podem ser observadas no Anexo A.

3.5. Elaboração do ciclo exemplificativo

O processo de criação de código para cada módulo e *skill* do procedimento corresponde a um processo complexo. Para o desenvolvimento de *skills* e módulos será utilizada a ferramenta *Eclipse*, que corresponde a uma ferramenta de desenvolvimento de *software* em diversas linguagens de programação, conhecido pela sua capacidade de ser estendida através de *plugins* [66]. A criação do corpo de código do módulo baseou-se na metodologia disponibilizada na página *GitHub* da ferramenta *SkillUp* e pode ser consultado no Anexo B com detalhes aprofundados. Prosseguindo, na sua conceção, vai ser produzido um ficheiro do tipo 'pom.xml', onde constam as dependências, *plugins* e outras especificidades relevantes do pacote desenvolvido. Posteriormente, é criado um ficheiro de classe do módulo, onde vai constar todo o corpo desse módulo. Por fim, após a colocação do módulo

[5] O autor recomenda correr o *camunda-bpmn-run*, mas apenas consegui correr o *camunda-bpmn-tomcat*, que produz o mesmo efeito.

a correr, o *Eclipse* produz uma pasta do tipo '.jar'. Para registar este módulo, basta a deslocação do ficheiro para a pasta 'include' do *runtime* do *SkillUp*.

No *SkillMex*, cada módulo é identificado por uma nomenclatura especifica, 'moduleIRI', que identifica o módulo no contexto da ferramenta. Deste modo, nas *skills* referentes a esse mesmo módulo, deve-se colocar essa mesma nomenclatura no campo respetivo. Relativamente a importações, basta importar o campo 'Module' do pacote de anotações do *SkillUp*. A nomenclatura '@Module' identifica ao *SkillMex* de que se trata de um módulo, onde deve constar o campo 'moduleIRI', podendo adicionalmente associar-se uma descrição ao critério do utilizador (Figura 18). Os módulos encontram-se especificados no Anexo D.

ModuleT.java

```
1 package modulet;
2 import skillup.annotations.Module;
3 @Module(moduleIri = "https://my-example-module.com/#moduleT", description = "MyModuleT")
4 public class ModuleT {
5
6 }
```

Figura 18.: Corpo de código do módulo correspondente ao equipamento de transporte circular (tapete).

A criação de uma *skill* acarreta mais especificidades. Tal como na criação do módulo, é desenvolvido um ficheiro '*pom.xml*' e, no final, obtêm-se um ficheiro do tipo '.jar'. A criação do pacote e da classe possui critérios específicos quanto à nomeação de parâmetros. Em termos de importações é muito mais complexo e depende das funcionalidades adicionadas a cada *skill* em específico, embora devam constar sempre os pacotes do *SkillUp* e eventualmente o pacote da máquina de estados desenvolvidos pelos mesmos autores do *SkillMex*, ('ISA88StateMachine'). Na Figura 19, surge, como exemplo, a *skill* 'Skill-Ir-Para' do tapete circular do processo modelado, que serve para deslocar a peça entre as estações. Na 'Skill-Ir-Para', existe uma menção à máquina de estados, que já se encontra incorporada no ficheiro carregado para o repositório e que é facultativo, visto que apenas serve para garantir que não ocorrem erros inesperados por parte da máquina de estados (sugerido pelo autor da ferramenta). A máquina de estados é ativada automaticamente com o *SkillUp*. Por outro lado, nas *skills* desenvolvidas, são necessárias outras importações relevantes, além do pacote 'skillup.annotations', que contém os termos relativos ao universo *SkillMex*, nomeadamente o pacote relativo aos termos utilizados na comunicação *OPC UA*, pertencente à biblioteca *Java* de código aberto *Eclipse Milo*, elemento desenvolvido pela fundação *Eclipse* (Figura 19) [67].

Após as importações, o campo '@Skill' identifica ao *SkillMex* de que se trata de uma *skill*. Nesta estrutura, temos 4 campos a identificar: o campo 'skillIri' contém o identificador da própria *skill*; o campo 'capabilityIri' serviria para identificar a capacidade, ainda que esta característica não esteja, na versão corrente, completamente implementada no *SkillMex*; o 'moduleIri' identifica o módulo ao qual a *skill* está associada; e, por último, o tipo, *OPC UA* ou serviços web *RESTful*, um modo de comunicação entre sistemas que seguem os métodos *HTTP (Hypertext Transfer Protocol)* [68]. Neste projeto, o objetivo focou-se no uso do protocolo *OPC UA*. Para cada *skill*, é necessária a identificação dos parâmetros a utilizar, através do parâmetro '@SkillParameter' e das saídas, através do '@SkillOutput'. Após estas etapas, podemos começar a moldar o corpo da *skill*, iniciado pelo desenvolvimento do comando '@Starting', que implementa a ligação ao servidor *OPC UA* do tapete na sua localização *IP*. De modo a facilitar o processo, não foram induzidas nenhumas seguranças de sistema. Posteriormente, no campo do '@Execute', temos a definição do *Address Space*, caracterizado no tópico 3.4.1. Implementou-se uma subscrição, o que permite que o cliente receba notificações quando os valores dos nós são alterados no servidor *OPC UA*, definindo o campo *subscription* do método.

```java
package skilld;
import skillup.annotations.*;
import java.util.concurrent.*;
import statemachine.Isa88StateMachine;
import org.eclipse.milo.opcua.sdk.client.AddressSpace;
import org.eclipse.milo.opcua.sdk.client.OpcUaClient;
import org.eclipse.milo.opcua.sdk.client.api.subscriptions.UaSubscription;
import org.eclipse.milo.opcua.sdk.client.nodes.UaObjectNode;
import org.eclipse.milo.opcua.sdk.client.methods.UaMethod;
import org.eclipse.milo.opcua.stack.core.Identifiers;
import org.eclipse.milo.opcua.stack.core.types.builtin.Variant;
import org.eclipse.milo.opcua.stack.core.security.SecurityPolicy;

@Skill(skillIri = "https://my-example-skill.com/#Skill-D", capabilityIri = "https://my-example-capability.com/#Cap-d", mo

//Habilidade para MOVER a peça (move) - ROBOT

public class Skilld {
    public OpcUaClient client;
    @SkillParameter(isRequired = true, option = { "1", "2", "3" })
    int station;

    @SkillParameter(isRequired = true, option = { "0", "1"})
    int Navete;

    @StateMachine // It assures that unexpected problems on SkillMex won't occur.
    Isa88StateMachine stateMachine;

    @Starting
    void run() throws Exception {
    client = OpcUaClient.create( // OPC UA instance creation to access the server's machine.
            "opc.tcp://192.168.228.179", //@Tapete
            endpoints -> // Valid endpoint of the robot
                endpoints.stream()
                    .filter(e -> e.getSecurityPolicyUri().equals(SecurityPolicy.None.getUri()))
                .findFirst(),
                configBuilder ->
                    configBuilder.build()
                );
    client.connect().get(); // C) Trying to automatically connect to the robot
    }

    // D) Application of the method

        //D.1 Access to the Method

    @Execute
    void executa() throws Exception {
        AddressSpace addressSpace = client.getAddressSpace();

        UaObjectNode serverNode = addressSpace.getObjectNode(Identifiers.Server);

        //create a subscription @ 1000ms

        UaSubscription subscription = client.getSubscriptionManager().createSubscription(1000.0).get();

        //Method calling

        UaMethod getMonitoredItems = serverNode.getMethod("Ir_Para"); //Method Name

        getMonitoredItems.call(
            new Variant[]{
                new Variant(subscription.getSubscriptionId())
            }
        );
    }

    @Completing
    void completa(OpcUaClient client, CompletableFuture<OpcUaClient> future) throws Exception {
        client.disconnect().get();
    }
}
```

Figura 19.: Corpo de código da *skill* 'Skill-Ir-Para' do tapete.

Segue-se a criação de um método, designado por 'Ir_Para'. Um método corresponde a uma função ou procedimento que pode ser chamado num servidor *OPC UA* para realizar uma operação específica. Estes são definidos nos nós do servidor e podem ser invocados por clientes para executar ações, como cálculos, controlo de dispositivos ou outras operações específicas da aplicação [67]. Por último, é necessário definir o campo '@Completing' da *skill*, ou seja, as ações a executar ao finalizar a execução da *skill*, neste caso desligar-se do cliente *OPC UA*. A metodologia anterior, embora com algumas divergências, seria aplicada às *skills* presentes neste projeto, em que cada uma remete para uma funcionalidade distinta.

Com a *skill* e os módulos desenvolvidos, mas não testados, procedeu-se então ao aprofundamento do diagrama *BPMN*, através do *Camunda Modeler*. Nesta ferramenta, modelou-se um *BPMN* mais profundo do que o desenvolvido no tópico 3.2. Na Figura 20, podemos visualizar o diagrama principal (do lado esquerdo) e três *skills* compostas (do lado direito), modeladas através de subprocessos.

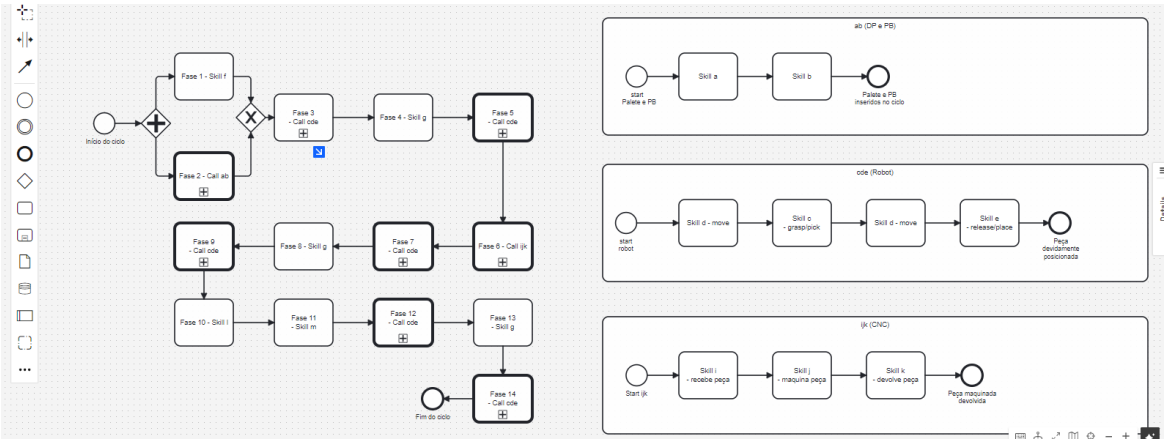

Figura 20.: Objetivo de diagrama final do nosso projeto.

No *Camunda Modeler*, a modelação da totalidade do BPMN descrito na Figura 20 é possível através de um único ficheiro do tipo '.bpmn', em que agrupam os processos e subprocessos. Porém, na versão atual do *SkillMex*, tal tarefa não é concretizável (confirmado pelo autor da ferramenta). Portanto, devido a essa limitação, procedeu-se ao desenho de diagramas de *BPMN* alternativos, de modo a contornar o problema.

- Versão 1 – Uma versão bastante extensa de todo o procedimento, não existindo blocos de subprocessos em que se agrupam todas as tarefas do ciclo exemplificativo, demonstrada na Figura 21.

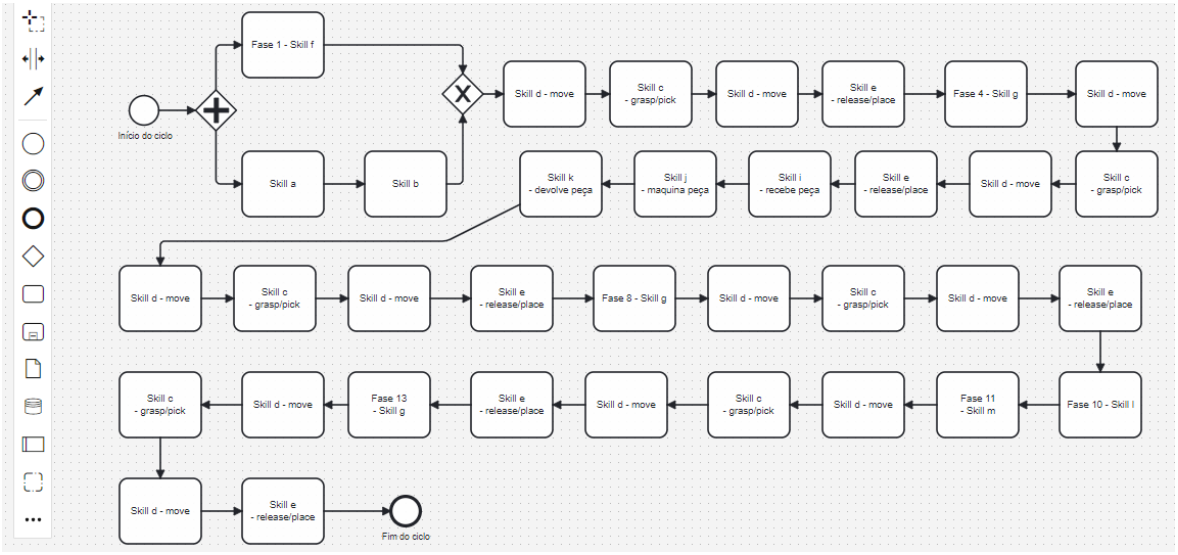

Figura 21.: Versão estendida do *BPMN* alternativo (Versão 1).

- Versão 2 – Uma separação em diversos diagramas a serem integrados e interligados no *SkillMex*. Portanto, foi desenvolvido um diagrama principal (Figura 24) e três outros, referentes a subprocessos, nomeadamente o 'SC ab' (Figura 22), o 'SC cde' (Figura 23) e o 'SC ijk' (Figura 22).

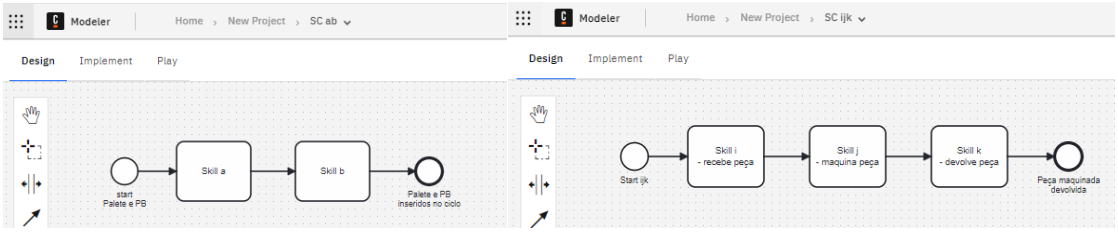

Figura 22.: Subprocessos que contém as *skills* a e b (à esquerda) e as *skills* i,j e k (à direita).

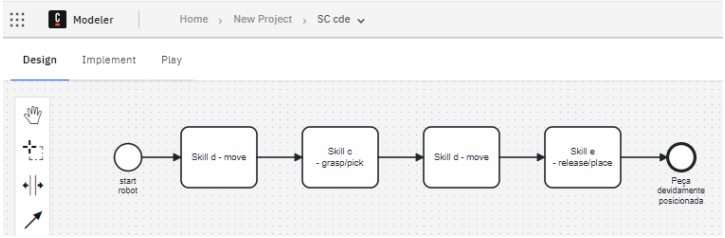

Figura 23.: Subprocesso que contém as *skill* c, d e e.

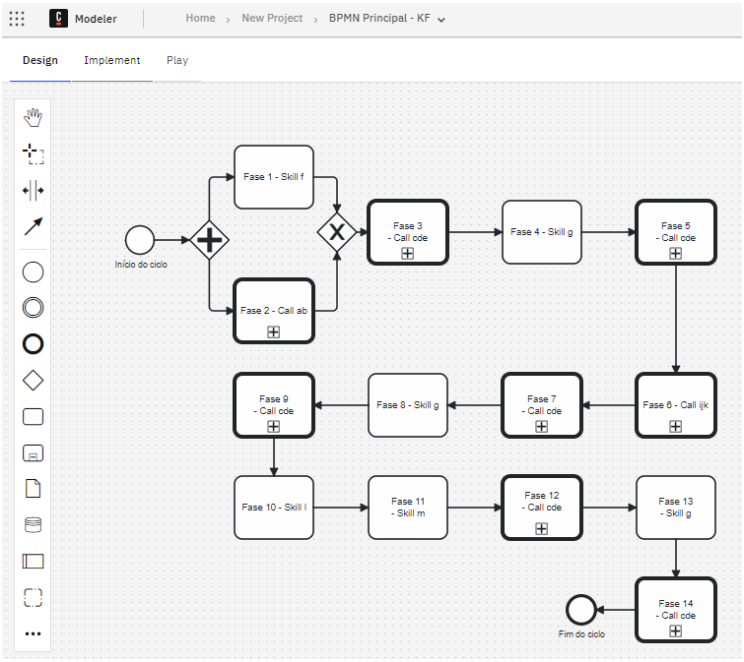

Figura 24.: Versão separada do *BPMN* - processo principal.

4. Resultados e Discussão

No presente capítulo serão apresentadas as sucessivas versões da aplicação do caso de estudo no *SkillMex*, a fim de verificar a viabilidade desta ferramenta na gestão da produção. Nesta fase, é crucial a verificação dos seguintes pontos chave:

- Compatibilidades dos módulos e *skills* no contexto do *SkillMex*;
- Verificação da viabilidade da aplicação direta do BPMN.

4.1. Verificação da viabilidade de aplicação

Neste estágio, será verificada a viabilidade do registo de todos os módulos e *skills* do projeto para verificar a compatibilidade com a ferramenta, assim como a viabilidade da submissão do *BPMN*. A colocação dos ficheiros '.jar' referentes a todos os módulos do projeto na pasta 'include' do *runtime*, a fim de serem registados, resulta numa mensagem que se verifica na área do *runtime*. Note-se que todas as *skills* e módulos foram desenvolvidos no *Eclipse* e não apresentam quaisquer erros de sintaxe ou de qualquer outra origem, já que, se fosse esse o caso, o próprio *Eclipse* não permitiria a criação de ficheiros *.jar*. Portanto, o presente objetivo é ver se se registam corretamente os elementos anteriores. Comecemos pelos módulos, uma vez que estes possuem um código muito mais simples.

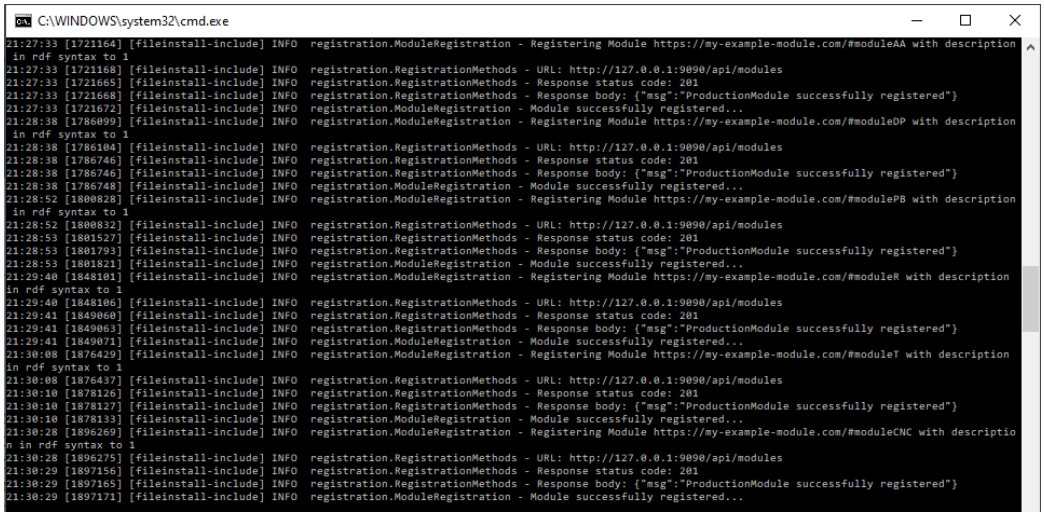

Figura 25.: Registo de módulos no *runtime*.

O sucesso do registo dos módulos é evidente: tanto no *SkillUp*, pela via do *runtime*, onde consta a indicação do registo dos módulos com sucesso (Figura 25), como no *SkillMex*, em

cuja página de módulos de produção, os módulos passaram a constar, como se verifica na Figura 26. Após a verificação do registo com sucesso dos módulos na ferramenta, procedeu-se à tentativa de registo da *skill* referente ao tapete, presente na secção 3.5. A colocação do ficheiro *.jar* referente à 'Skill-Ir-Para' na pasta *include* do *runtime* resultou numa não reação por parte do *SkillMex* e do *runtime*. Posteriormente, em conversações com um dos autores da ferramenta, conclui-se que os pacotes que usem o pacote *OPC UA Milo*, de momento, não são compatíveis com o *SkillMex*.

De facto, o conceito que se pretendia implementar, o de usar um cliente *OPC UA* associado a uma capacidade, nunca foi aplicado desta forma nesta ferramenta. Portanto, encontramos aqui uma limitação grave ao seguimento dos objetivos originalmente estabelecidos. Apesar destas limitações, e ainda em fase de teste, pode testar-se a aplicabilidade do *BPMN* com a versão 1 evidenciada no final da secção 3.5 correspondente à versão bastante extensa de todo o procedimento produtivo.

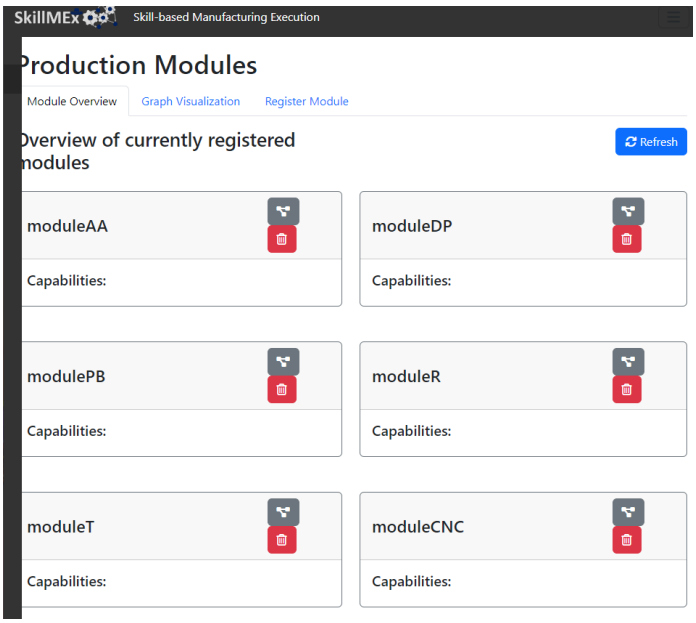

Figura 26.: Registo com sucesso dos módulos do projeto.

O modelo desta versão, assim como os outros modelos implementados, foi desenvolvido no *Camunda Modeler*, o que indica que teremos de efetuar o download do modelo em formato '.bpmn' e carregá-lo para o contexto dos *SkillMex/SkillUp*. Para isso, na página do *SkillMex*, basta ir ao separador '*Production Processes'* e no campo '*Model New Process'*, basta carregar no campo 'Load from File' e selecionar o ficheiro do BPMN na sua localização. Após isto, basta indicar para cada bloco o tipo de tarefa, o que, neste caso, é uniforme, uma vez que a totalidade dos blocos são do tipo '*Service task'*. Posteriormente,

basta indicar o tipo de elemento, neste caso são todos do tipo '*skill'* e, posteriormente, selecionar e associar a *skill* referente a cada um destes blocos. Neste caso, para fim de teste, foram usadas *skills* sem o pacote *Eclipse Milo*. Logo após, efetua-se o *deploy* do processo através do comando com essa mesma designação, podendo ser carregado com (Figura 27).

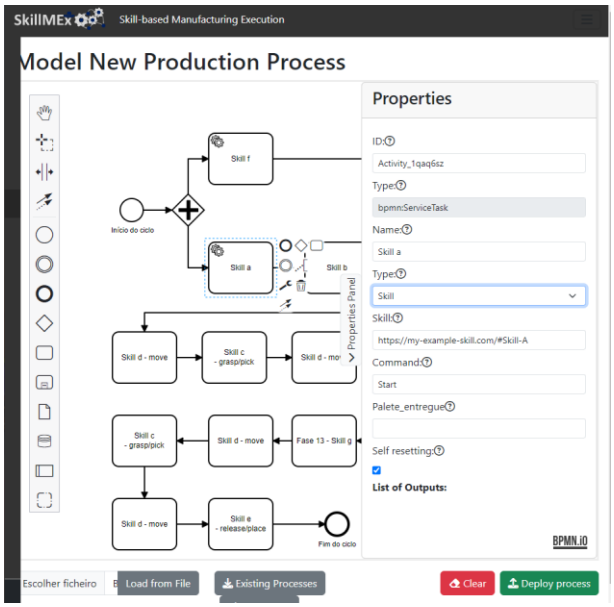

Figura 27.: *BPMN* da versão 1 no *SkillMex*.

Como resultado do carregamento do *BPMN* para o *SkillMex*, deve-se verificar a presença do modelo, no separador <*Process Definitions*> da página '*Process Control'* (Figura 28).

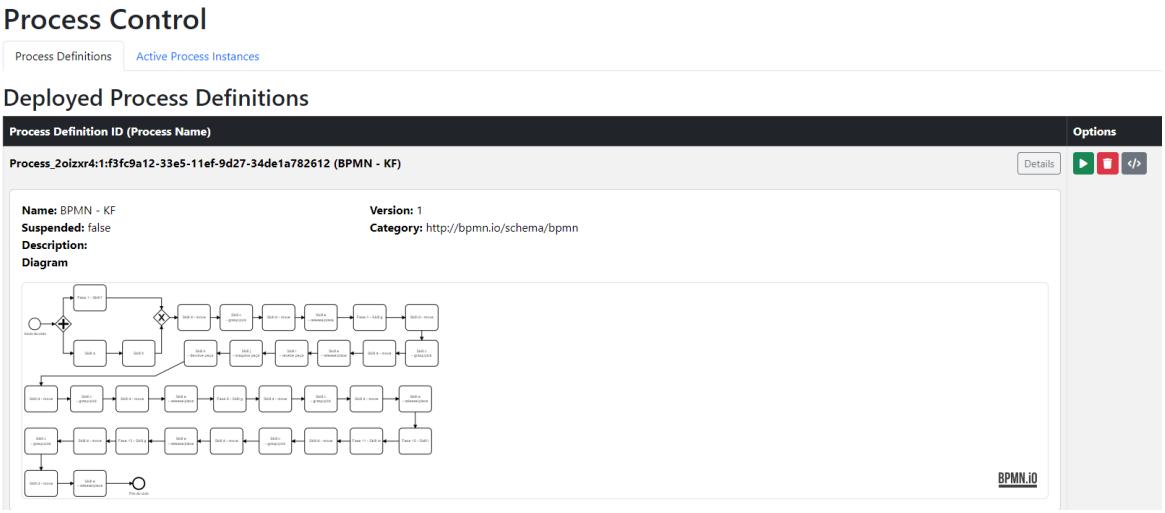

Figura 28.: *BPMN* versão 1 carregado no SkillMex.

Numa fase inicial, testou-se o *BPMN* apenas com blocos vazios, de modo a averiguar o sucesso do carregamento, o que se confirmou. Numa outra tentativa, foram associadas *skills*, porém com o corpo do código mais simples, o que resultou numa tentativa bem-sucedida.

No teste da segunda solução para o problema, que inclui três diagramas *BPMN*, um principal e três outros secundários há que testar o carregamento de cada um e a sua interligação. Neste caso, tanto o carregamento dos diagramas *BPMN* secundário quanto do principal, ainda que com *skills* 'vazias' associadas, decorreu sem problemas, tanto no diagrama principal como nos subprocessos (Figura 29 e Figura 30).

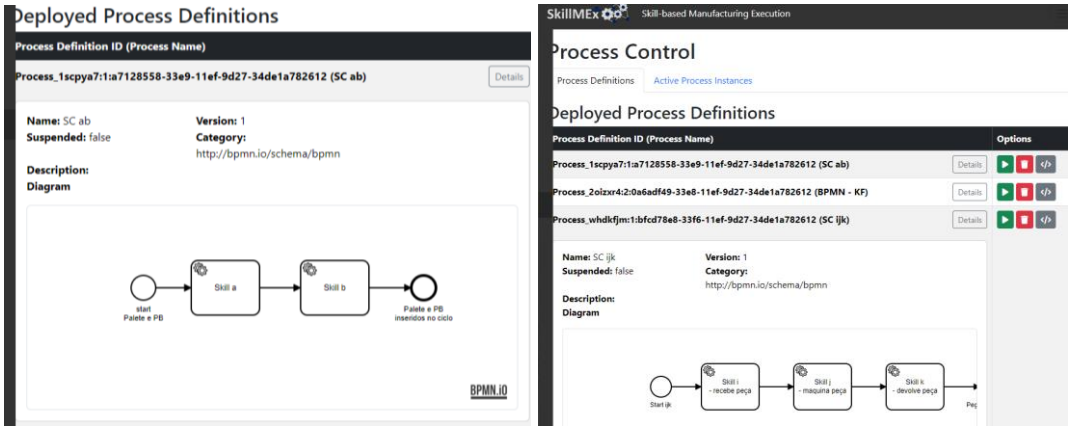

Figura 29.: Subprocessos da versão 2 do *BPMN* ab (à esquerda) e ijk (à direita).

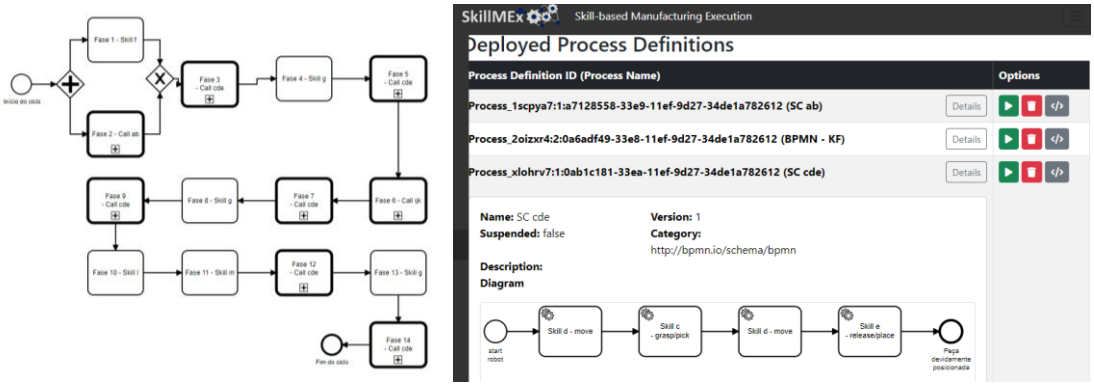

Figura 30.: *BPMN* principal da versão 2 (à esquerda) e subprocesso do *BPMN* cde da versão 2 (à direita).

Porém encontrou-se uma limitação importante do *SkillMex*, já que, ao contrário do que acontece com o *Camunda Modeler*, que lhe serve de base, não é possível a associação de subprocessos a um diagrama *BPMN* principal. Como resultado, verifica-se que não é possível, no atual estado de desenvolvimento do *SkillMex*, usar esta possibilidade, ficando a solução limitada ao uso de um único diagrama mais complexo e difícil de reproduzir e gerir.

Por outro lado, encontrou-se um outro problema. Quando é iniciada uma nova instância para qualquer um dos *BPMN* carregados (Figura 31, à esquerda), verifica-se um erro que impede a viabilização da operação (Figura 31, à direita). De acordo com o autor da ferramenta, quando são utilizados blocos do tipo '*service task'*, que permitem a associação

de *skills*, módulos ou capacidades (teoricamente), surge o erro que consta na Figura 31, embora se desconheça a sua origem.

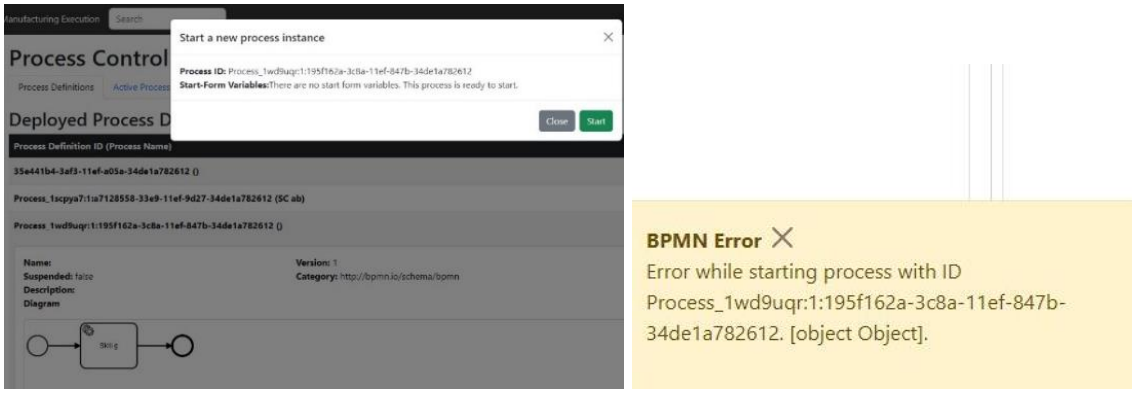

Figura 31.: Processo de início de uma nova instância para o(s) *BPMN* carregado(s) (à esquerda) e a mensagem de erro resultante (à direita).

Em suma, nesta fase, não obstante o carregamento com sucesso de módulos e *skills* (desde que não utilizem o pacote *OPC UA Milo*), a versão atual do *SkillMex*, não permite de facto, ao contrário do anunciado, utilizar os diagramas *BPMN* como ferramenta de trabalho, o que se traduz numa limitação grave.

4.2. Potenciais Soluções Alternativas

Os problemas e limitações encontrados na versão atual do *SkillMex* inviabilizam a ferramenta quanto à configuração que se afigurava ideal, ou seja, não é possível estabelecer uma interação direta entre as *skills* desenvolvidas, pela via do *BPMN* modelado, com os servidores de cada equipamento do chão de fábrica modelado.

Para verificar o modo de funcionamento do *SkillUp*, recorre-se ao *UaExpert*, que é um cliente *OPC UA* genérico, utilizado para aceder a dados, estabelecer alarmes e condições, acesso a históricos e chamar métodos pela via *OPC UA*, ferramenta disponibilizada pela empresa *Unified Automation* [69] e cujo interface se pode visualizar no exemplo da Figura 32. Com o acesso ao cliente *OPC UA* do *SkillUp*, foi possível verificar que os parâmetros de cada *skill* estão acessíveis e podem ser sujeitos a leitura e edição. Com isto, pode ser idealizada uma nova estratégia para estabelecer a comunicação entre as *skills* e os equipamentos.

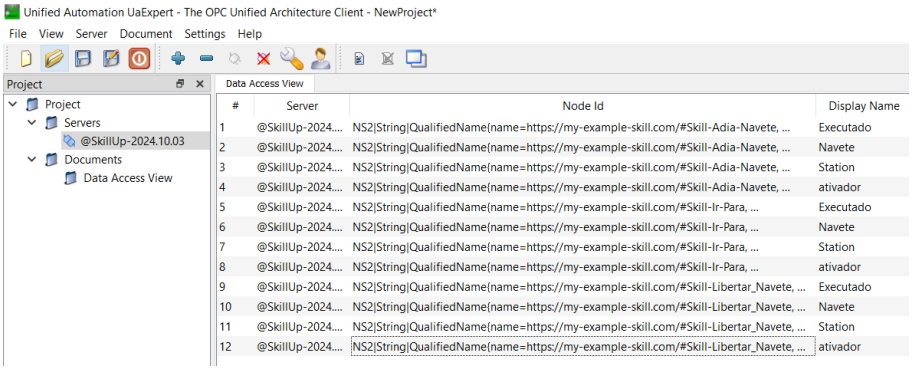

Figura 32.: Interface do *UaExpert*.

Tendo em conta que não é possível utilizar diagramas do tipo *BPMN*, vamos pelo menos efetuar a comunicação entre as *skills* individuais e o tapete. No equipamento, neste caso o tapete, cuja lógica é aplicável às restantes máquinas, existe um cliente *OPC UA* que possui métodos idealizados previamente ao presente projeto, que podem ser visualizados na Figura 33. Estes métodos já assumem uma lógica idêntica às *skills* idealizadas para o nosso ciclo produtivo, inclusivamente os mesmos parâmetros 'station' e 'Navete'. Através deste cliente *OPC UA*, é possível executar os métodos incorporados, permitindo a edição dos parâmetros 'station' e 'Navete' para os valores que pretendemos.

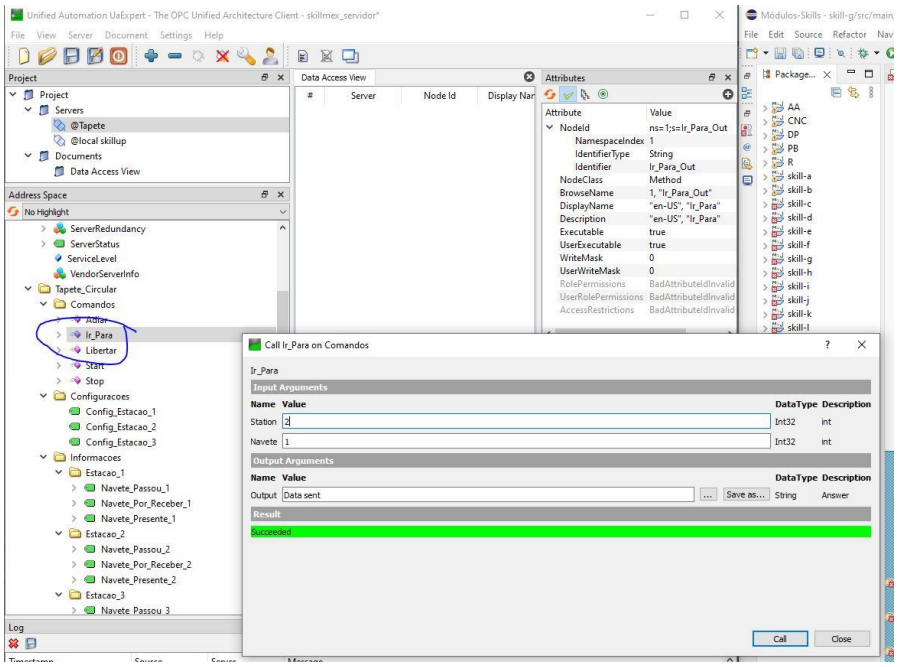

Figura 33.: Cliente *OPC UA* do tapete e potencial meio de gestão do mesmo.

A comunicação entre o cliente *OPC UA SkillUp* e o cliente *OPC UA* do equipamento não pode ser estabelecida, já que esta apenas é possível entre um servidor e um cliente. Para concretizar a comunicação entre dois clientes *OPC UA*, um deles tem de atuar como servidor

ou então é necessária a existência de um intermediário no processo [70]. Deste modo, podemos converter um dos clientes *OPC UA* em servidor. Com isto, procede-se à clonagem do atual cliente *OPC UA* do tapete, a fim de convertê-lo em servidor, visto que este é mais moldável no seu estado atual para o efeito.

A transformação do cliente do tapete em servidor, remete para o desenvolvimento de um ficheiro, do tipo *python*, que estabeleça a comunicação com o cliente *OPC UA SkillUp*, denominado por 'Duplo_Cliente_Kevin.py'. Neste ficheiro, desenvolvido previamente ao presente projeto, é caracterizado o dispositivo que vai aceder (neste caso, o meu computador pessoal), o pórtico do *SkillUp* e a rede local. Para proceder à execução de ordens, é necessária a instalação do ficheiro 'python-39.13', cuja versão foi a única compatível para o efeito. Além disso, na linha de comandos e na localização do utilizador da rede, neste caso, da instituição de ensino, digitou-se o seguinte código, 'pp install asyncua'. Após este procedimento, o ficheiro 'Duplo_Cliente_Kevin.py' está pronto a ser utilizado, permitindo manipular o tapete. Foram necessárias várias iterações e resoluções de problemas até chegar ao objetivo final. Posteriormente, para executar o ficheiro anterior, basta na linha de comandos e na mesma localização anterior, executar o seguinte comando 'python Duplo_Cliente_Kevin.py'. Após este procedimento, no meu computador pessoal, com ligação à mesma rede e com o mesmo *IP* que o equipamento, é possível manipular as ações físicas do tapete.

Mas primeiro, para estabelecer a ligação entre o cliente *OPC UA* do *SkillUp* com o servidor do tapete requer que sejam constituídas novas *skills* teoricamente sem ligação via *OPC UA* direta entre a *skill* e o equipamento. Portanto, a seguir caracteriza-se a nova *skill* do tapete 'Skill-Ir-Para' com este novo conceito, será a única que será detalhada, as restantes *skills* podem ser encontradas no Anexo E. Dentro do corpo da *skill* (Figura 34), inicialmente constam as importações, que neste caso correspondem ao pacote das anotações do *SkillUp* e da máquina de estados. Sucede-se a descrição da *skill* para que o *SkillMex* consiga reconhecer do que se trata. Dentro do campo '@Skill', seguindo a mesma direção que a *skill* desenvolvida no tópico 3.5, consta o campo 'skillIri', uma nomenclatura para a *skill*, '#Skill-Ir-Para', seguindo-se o campo 'CapabilityIri' correspondente a '#Cap-Ir-Para', seguindo-se do campo 'moduleIri', que neste caso pertence ao módulo do tapete, correspondente a '#moduleT' e por fim, o tipo de *skill*, do tipo *OPC UA*. Após esta descrição inicial, o corpo da *skill* é continuado pela definição dos parâmetros 'Station' e 'Navete', que podem ser

manipulados de acordo com o pretendido, além da definição da saída 'Executado', que deve indicar ao servidor do tapete se existe ordem para executar a *skill* (em que o 'Executado' assume o valor de 1). Após a execução da tarefa, o valor de saída deve retomar a zero, mantendo-se neste modo até ser solicitada uma nova ordem. Tendo em conta que a saída 'Executado' não permite a manipulação direta por parte do tapete, visto que se trata de uma saída, existe a necessidade de estabelecer um novo parâmetro, cuja denominação será 'ativador' que efetuará a manipulação indireta da saída 'Executado'.

Progredindo no corpo da *skill*, foi invocada a máquina de estados. No campo 'Starting', é atribuído à variável 'Executado' o valor de 0, uma vez que deve estar deste modo antes da execução de um método. Esta variável é colocada a 1 de seguida no campo '@Execute' para implementar o método. E, por fim, no campo '@Completing' a saída 'Executado' retoma à nulidade (Figura 34).

```
1  package skill_Ir_Para;
2
3  import skillup.annotations.*;
5
6  @Skill(skillIri = "https://my-example-skill.com/#Skill-Ir-Para", capabilityIri = "https://
7
8  //Skill para mover peça de x para y - TAPETE
9
10 public class Skill_Ir_Para {
11     @SkillParameter(isRequired = true, option = { "1", "2", "3" })
12     int Station;
13
14     @SkillParameter(isRequired = true, option = { "1", "2", "3" })
15     int Navete;
16
17     @SkillParameter(isRequired = true)
18     boolean ativador;
19
20     @SkillOutput(isRequired = true)
21     int Executado;
22
23     @StateMachine // It assures that unexpected problems on SkillMex won't occur.
24     Isa88StateMachine stateMachine;
25
26     @Starting
27     public void inicio() {
28         Executado = 0;
29     }
30
31     @Execute
32     public void execute() {
33         if (!ativador) {
34             Executado = 1;
35         }
36         else {
37         }
38     }
39
40     @Completing
41     public void complete() {
42         Executado = 0;
43         }
44     }
```

Figura 34.: Novo modelo da 'skill-Ir-Para'.

Na aplicação do *SkillMex*, no separador *skills*, é possível a manipulação dos parâmetros, através dos campos 'Start', 'Stop', 'Abort' e 'Reset', presentes no campo <*Executable transitions*>, associados a cada *skill* registada, demonstrados na Figura 35.

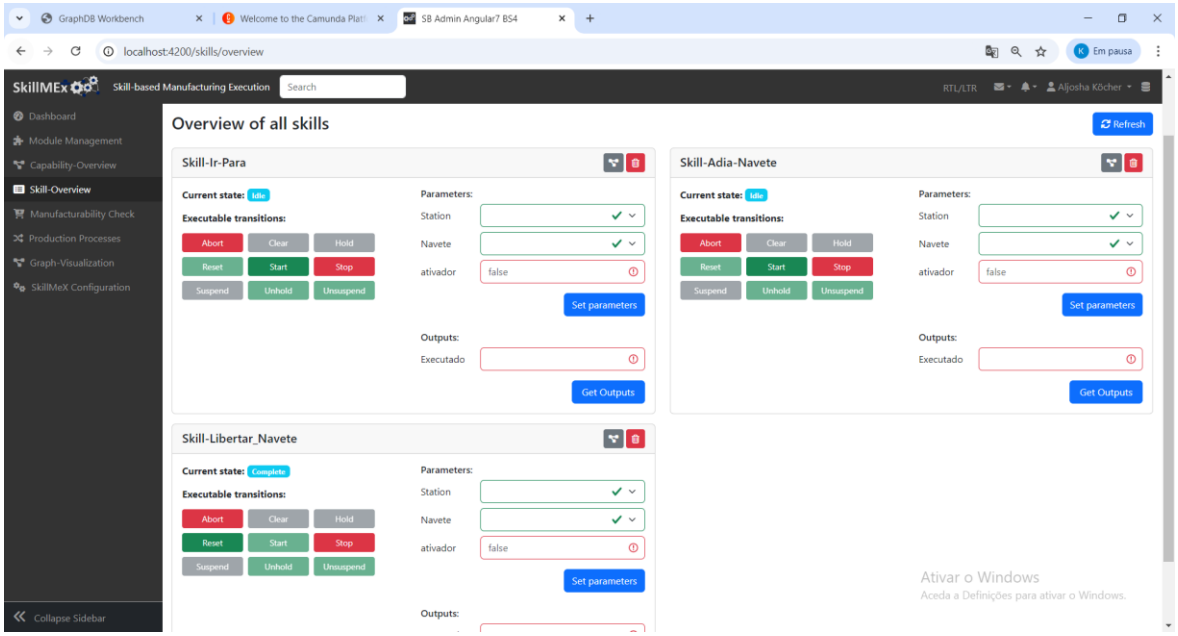

Figura 35.: Separador das *skills* no *SkillMex*

Na mesma página, a manipulação dos parâmetros previamente definidos ‘Station’, ‘navete’ e ‘ativador’ revelou-se inexequível, visto que o separador da aplicação ‘Parameters’, que se encontra na Figura 35 se encontra inoperacional (foi confirmado pelo autor da ferramenta que este separador nunca foi devidamente desenvolvido). Deste modo, para se proceder à manipulação dos parâmetros, é necessária a utilização do *UaExpert* para o efeito, através do servidor do *SkillUp* (Figura 36).

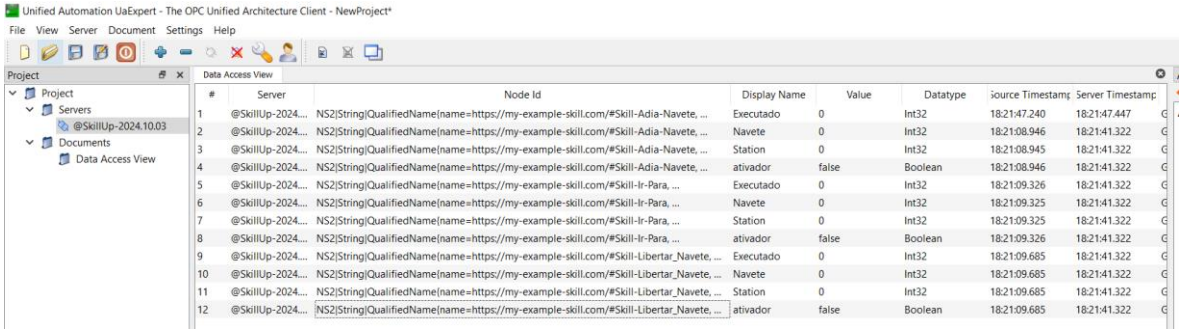

Figura 36.: Janela do *UaExpert* com o servidor do *SkillUp*

Esta manipulação é acessível e foi efetuada com êxito nas instalações do LRAFI (Figura 37). Com o acionamento do campo ‘start’ da Figura 35, é possível a execução de uma *skill* individual, em que a variável ‘Executado’ altera de estado, de zero para um, regressando novamente a zero no fim de concluir a ação. Com vista a uma correta operação de todo o universo *SkillMex*, recomendo a totalidade da instalação dos elementos constituintes, *Graph DB*, *camunda-tomcat* e demais ferramentas, com acesso a uma mesma rede local (com um

mesmo endereço *IP*). A rede local em que ocorre a instalação deve ser a mesma que o equipamento. A mudança de rede remete para o não funcionamento da ferramenta. Portanto, a inflexibilidade da ferramenta é verificada, uma vez que não é possível (sem a reinstalação de todas as ferramentas) instalar as ferramentas num outro local com outra rede, para depois executar no local do equipamento.

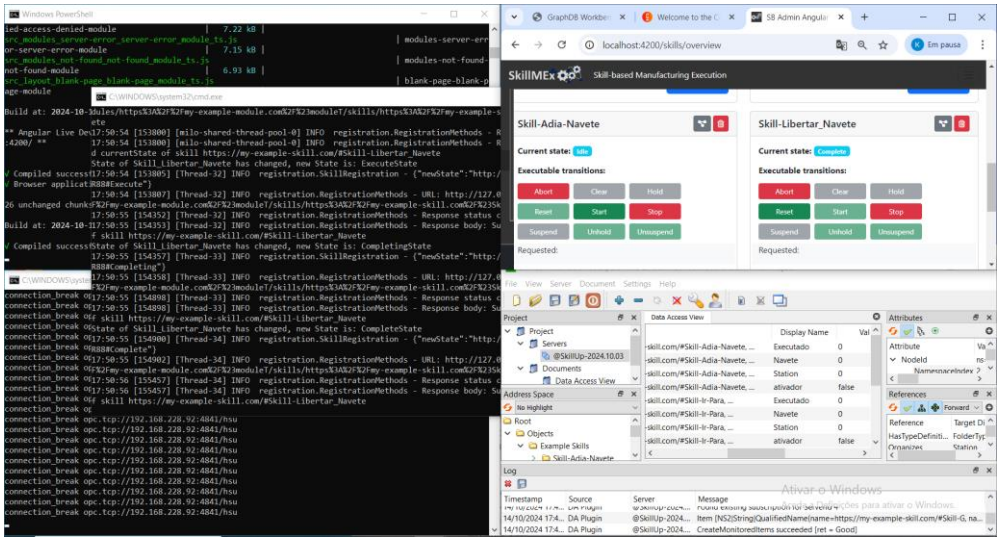

Figura 37.: Panorama global das ferramentas aquando manipulação de *skills*.

Na Figura 37, constam todas as ferramentas implicadas na manipulação da trajetória de uma peça. Quando ocorre a execução de uma tarefa com sucesso, surgem mensagens na janela do *SkillUp runtime* (janela central da figura) que indicam o êxito de execução. Portanto, foi possível a utilização do *SkillMex* com sucesso para proceder à aplicação de *skills* de uma forma individual, após várias iterações de teste.

5. Conclusões e Trabalhos Futuros

Os trabalhos de aplicação das ferramentas anteriores conduziram ao presente capítulo, que se divide nas conclusões resultantes dos trabalhos experimentais desenvolvidos e nas sugestões de desenvolvimentos futuros.

5.1. Conclusões

Dentro dos objetivos estabelecidos, pode concluir-se que a ferramenta em estudo permite o registo de módulos e de *skills*. No entanto, e conforme anteriormente discutido, esta ferramenta apresenta algumas limitações face às necessidades identificadas. De modo particular, importa destacar que o *SkillMex* não permite registar *skills* que contenham determinados pacotes associados, nomeadamente o *OPC UA Milo*, o que permitiria efetuar a ligação direta entre o equipamento e o próprio *SkillMex*, via *OPC UA*. Adicionalmente, verificou-se que a ferramenta não permite a aplicação de diagramas *BPMN* com blocos que sejam do tipo 'service task', ou seja, que estejam associadas *skills*. Assim, embora seja permitido o registo do diagrama *BPMN*, este não realiza a sua execução. Neste contexto, verificou-se ainda que a ferramenta também não permite criar ou carregar subprocessos associados a um diagrama principal. Tudo o que fomente alguma complexidade de execução, inibe totalmente a sua concretização, uma vez que é utilizado um *plug-in* do *Camunda Modeler* que, quando usado diretamente, na respetiva página *web*, funciona corretamente.

Numa nota mais positiva, através do *runtime*, o *SkillMex* está em constante atualização, o que permite, com flexibilidade o registo de novos módulos e *skills* para o seu universo. Ainda assim, esta característica apresenta limitações, uma vez que quando uma *skill* é alterada, após se inserir o ficheiro '.jar' da *skill* na pasta 'include' do *runtime*, é devolvido um erro que é solucionado apenas com o reinício da aplicação. Portanto, a aplicação da ferramenta selecionada para efetuar a gestão da produção revelou-se impossível, com o seu atual estado de desenvolvimento. Esta conclusão, não era, de todo, aparente na fase de procura e seleção, tendo em conta a informação acessível nessa fase do trabalho. Os processos de instalação da ferramenta foram bastante complexos e trabalhosos, o que revelou a sua inoperabilidade já numa fase bastante tardia, em que problemas de instalação, de dependências e compatibilidade dos diversos módulos e respetivas versões, mascararam o problema. No entanto, a fragilidade da ferramenta, não inviabiliza o potencial do uso da ideia

base pela aplicação de modelos associados a módulos e *skills* no âmbito da gestão da produção. Apesar destes contratempos, fica patente que a aplicação destes conceitos é uma via interessante para gerir de forma automática os sistemas produtivos e que, uma vez que é construída sobre módulos de elevada abstração, de alguma forma agnósticos face aos equipamentos específicos que constituem o chão de fábrica, o que promove a modularidade e a possibilidade de reutilização em diferentes processos, podendo ainda beneficiar do contributo da inteligência artificial. É patente, todavia, que a ferramenta está ainda num estado de desenvolvimento muito mais atrasado do que é aparente na informação que é fornecida publicamente.

5.2. Trabalhos Futuros

Seguindo a mesma linha de raciocínio, a ferramenta *SkillMex* revelou-se uma ferramenta de aplicabilidade limitada face aos problemas que ainda apresenta, pelo menos no seu estado atual de desenvolvimento. Numa fase inicial da pesquisa por este tipo de ferramentas, esta destacou-se pela utilização de conceitos relativos a modelos que se baseiam em *skills* e módulos e, realmente, entende-se este como um caminho de desenvolvimento. Este tipo de conceitos pode, efetivamente, ser a solução de problemas de gestão da produção na indústria e, portanto, não deve ser descartada a melhoria do *SkillMex* e demais ferramentas nas suas diversas limitações e incompatibilidades.

É de referir ainda que a potencial melhoria do *SkillMex* pode passar pelo desenvolvimento do seu código, através da adaptação do mesmo quanto às compatibilidades com o pacote *Milo* e talvez outros pacotes relevantes neste contexto. Por outro lado, a execução dos diagramas *BPMN* também requer uma resolução quanto às incompatibilidades com as *service tasks*. Numa outra abordagem, pode-se descartar a ideia de utilização do *SkillMex* no presente projeto e focar numa solução criteriosa de uma outra ferramenta que se encontre disponível e aplicar os mesmos conceitos. Para além disso, é possível a interligação de uma outra ferramenta *MES* com a inteligência artificial (*IA*), o que pode constituir um bom tópico de discussão, uma vez integrar o progresso inevitável de desenvolvimento tecnológico de diversos mercados, nomeadamente o da produção industrial. A *IA* é uma ferramenta para o futuro e certamente será uma realidade bastante próxima da indústria, o que incrementará a evolução de processos industriais e de decisão nas empresas.

Bibliografia

[1] Almiro de Oliveira, “A importância dos sistemas de informação,” 1998. Accessed: Feb. 19, 2025. [Online]. Available: https://ejms.iseg.ulisboa.pt/files/1998-A_importancia_dos_sistemas_de_informacao_para_a_industria.pdf

[2] I. T. C. de Novaes Bastos, “Desenvolvimento de Atividades de Suporte à Implementação de um Sistema Informático de Controlo da Produção,” Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2017. Accessed: Mar. 24, 2022. [Online]. Available: https://repositorio-aberto.up.pt/bitstream/10216/105577/2/201702.pdf

[3] S. Mantravadi, C. Li, and C. Møller, “Multi-agent Manufacturing Execution System (MES): Concept, Architecture & ML Algorithm for a Smart Factory Case,” in *Proceedings of the 21st International Conference on Enterprise Information Systems*, SCITEPRESS - Science and Technology Publications, 2019, pp. 477–482. doi: 10.5220/0007768904770482.

[4] R. and K. P. Govindaraju, “A methodology for Manufacturing Execution Systems (MES) implementation.,” *IOP Conference Series: Materials Science and Engineering. Vol. 114. No. 1. IOP Publishing*, 2016.

[5] Accevo Systems, “How to integrate MES with ERP system (SAP, Oracle, JDE, LN),” How to integrate MES with ERP system (SAP, Oracle, JDE, LN).

[6] R. F. Elliott, “Manufacturing Execution System (MES) An Examination of Implementation Strategy,” California Polytechnic State University, San Luis Obispo, California, 2013. doi: 10.15368/theses.2013.144.

[7] B. K. Choi and B. H. Kim, “MES (manufacturing execution system) architecture for FMS compatible to ERP (enterprise planning system),” *Int J Comput Integr Manuf*, vol. 15, no. 3, pp. 274–284, Jan. 2002, doi: 10.1080/09511920110059106.

[8] X. Chen, C. Nophut, and T. Voigt, “Manufacturing execution systems for the food and beverage industry: A model-driven approach,” *Electronics (Switzerland)*, vol. 9, no. 12, pp. 1–21, Dec. 2020, doi: 10.3390/electronics9122040.

[9] A. Perzylo, I. Kessler, S. Profanter, and M. Rickert, “Toward a Knowledge-Based Data Backbone for Seamless Digital Engineering in Smart Factories,” in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2020, pp. 164–171. doi: 10.1109/ETFA46521.2020.9211943.

[10] A. P. Sheth and J. A. Larson, “Federated database systems for managing distributed, heterogeneous, and autonomous databases,” *ACM Comput Surv*, vol. 22, no. 3, pp. 183–236, Sep. 1990, doi: 10.1145/96602.96604.

[11] S. P. Gardner, “Ontologies and semantic data integration,” *Drug Discov Today*, vol. 10, no. 14, pp. 1001–1007, Jul. 2005, doi: 10.1016/S1359-6446(05)03504-X.

[12] A. Busboom, “Automated generation of OPC UA information models — A review and outlook,” *J Ind Inf Integr*, vol. 39, p. 100602, May 2024, doi: 10.1016/j.jii.2024.100602.

[13] A. ; F. , A. Köcher, “Model-Based Engineering of CPPS Functions and Code Generation for Skills,” Hamburg, Germany, 2022. doi: https://doi.org/10.48550/arXiv.2201.13290.

[14] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, “The development of an ontology for describing the capabilities of manufacturing resources,” *J Intell Manuf*, vol. 30, no. 2, pp. 959–978, 2019, doi: 10.1007/s10845-018-1427-6.

[15] Zhongli Ding and Yun Peng, “A probabilistic extension to ontology language OWL,” in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, IEEE, 2004, p. 10 pp. doi: 10.1109/HICSS.2004.1265290.

[16] Ontotext GraphDB, “Knowledge Hub,” What Is an RDF Triplestore? Accessed: Mar. 06, 2025. [Online]. Available: https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/

[17] MESA International, “MESA is where Manufacturing Meets IT,” Chandler, USA, 2025. Accessed: Mar. 07, 2025. [Online]. Available: https://mesa.org/

[18] S. Jaskó, A. Skrop, T. Holczinger, T. Chován, and J. Abonyi, “Development of manufacturing execution systems in accordance with Industry 4.0 requirements: A review of standard- and ontology-based methodologies and tools,” Dec. 01, 2020, *Elsevier B.V.* doi: 10.1016/j.compind.2020.103300.

[19] Vector Blue Hub, “Knowledge Zone,” A Guide to Engineering Change Order (ECO). Accessed: Feb. 26, 2025. [Online]. Available: https://vectorbluehub.com/a-guide-to-engineering-change-order-eco

[20] L. Prades, F. Romero, A. Estruch, A. García-Dominguez, and J. Serrano, “Defining a Methodology to Design and Implement Business Process Models in BPMN According to the Standard ANSI/ISA-95 in a Manufacturing Enterprise,” *Procedia Eng*, vol. 63, pp. 115–122, 2013, doi: 10.1016/j.proeng.2013.08.283.

[21] V. Serifi, P. Dašić, R. Ječmenica, and D.Labović, “Functional and information modeling of production using IDEF methods,” *Strojniski Vestnik*, vol. 55, pp. 131–140, Feb. 2013, Accessed: Mar. 08, 2025. [Online]. Available: https://www.sv-jme.eu/?ns_articles_pdf=/ns_articles/files/ojs3/1563/submission/1563-1-1898-1-2-20171103.pdf&id=4931

[22] Weihenstephan Standards, “The communication interface for your machines,” The communication interface for your machines. Accessed: Mar. 08, 2025. [Online]. Available: https://www.weihenstephan-standards.com/

[23] M. Schleipen, R. Drath, and O. Sauer, “The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system,” in *2008 IEEE International Symposium on Industrial Electronics*, IEEE, Jun. 2008, pp. 1786–1791. doi: 10.1109/ISIE.2008.4676932.

[24] M. Witsch and B. Vogel-Heuser, “Towards a Formal Specification Framework for Manufacturing Execution Systems,” *IEEE Trans Industr Inform*, vol. 8, no. 2, pp. 311–320, May 2012, doi: 10.1109/TII.2012.2186585.

[25] N. E. Cagiltay, G. Tokdemir, O. Kilic, and D. Topalli, “Performing and analyzing non-formal inspections of entity relationship diagram (ERD),” *Journal of Systems and Software*, vol. 86, no. 8, pp. 2184–2195, Aug. 2013, doi: 10.1016/j.jss.2013.03.106.

[26] H. M. Abdullah and A. M. Zeki, “Frontend and Backend Web Technologies in Social Networking Sites: Facebook as an Example,” in *2014 3rd International Conference on Advanced Computer Science Applications and Technologies*, IEEE, Dec. 2014, pp. 85–89. doi: 10.1109/ACSAT.2014.22.

[27] B. Zhou, “Lean principles, practices, and impacts: a study on small and medium-sized enterprises (SMEs),” *Ann Oper Res*, vol. 241, no. 1–2, pp. 457–474, Jun. 2016, doi: 10.1007/s10479-012-1177-3.

[28] P. Perico, E. Arica, D. J. Powell, and P. Gaiardelli, “MES as an Enabler of Lean Manufacturing,” *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 48–53, 2019, doi: 10.1016/j.ifacol.2019.11.306.

[29] A. M. Kholif, D. S. Abou El Hassan, M. A. Khorshid, E. A. Elsherpieny, and O. A. Olafadehan, “Implementation of model for improvement (PDCA-cycle) in dairy laboratories,” *J Food Saf*, vol. 38, no. 3, Jun. 2018, doi: 10.1111/jfs.12451.

[30] H. Zhou and W. C. Benton, “Supply chain practice and information sharing,” *Journal of Operations Management*, vol. 25, no. 6, pp. 1348–1365, Nov. 2007, doi: 10.1016/j.jom.2007.01.009.

[31] S. Preradovic, N. Karmakar, and I. Balbin, “RFID Transponders,” *IEEE Microw Mag*, vol. 9, no. 5, pp. 90–103, Oct. 2008, doi: 10.1109/MMM.2008.927637.

[32] A. Jain, R. Bhatti, and H. Singh, “Total productive maintenance (TPM) implementation practice,” *International Journal of Lean Six Sigma*, vol. 5, no. 3, pp. 293–323, Jul. 2014, doi: 10.1108/IJLSS-06-2013-0032.

[33] A. Chmielowiec, P. Żurawski, S. Sikorska-Czupryna, L. Klich, and P. Organiściak, “Application of Neural Networks for Defect Detection in Rotationally Symmetric

Components," *Advances in Science and Technology Research Journal*, vol. 18, no. 8, pp. 403–415, Dec. 2024, doi: 10.12913/22998624/194891.

[34] P. Perico, E. Arica, D. J. Powell, and P. Gaiardelli, "MES as an Enabler of Lean Manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 48–53, 2019, doi: 10.1016/j.ifacol.2019.11.306.

[35] P. Muchiri and L. Pintelon, "Performance measurement using overall equipment effectiveness (OEE): literature review and practical application discussion," *Int J Prod Res*, vol. 46, no. 13, pp. 3517–3535, Jul. 2008, doi: 10.1080/00207540601142645.

[36] M. Talha, "Total quality management (TQM): an overview," *The Bottom Line*, vol. 17, no. 1, pp. 15–19, Mar. 2004, doi: 10.1108/08880450410519656.

[37] G. D'Antonio, J. Sauza Bedolla, A. Rustamov, F. Lombardi, and P. Chiabert, "The Role of Manufacturing Execution Systems in Supporting Lean Manufacturing," 2016, pp. 206–214. doi: 10.1007/978-3-319-54660-5_19.

[38] acatech Forschungsunion, ""Recommendations for implementing the strategic initiative INDUSTRIE 4.0," Frankfurt, Apr. 2013.

[39] A. Hayward, M. Rappl, and A. Fay, "A SysML-based Function-Centered Approach for the Modeling of System Groups for Collaborative Cyber-Physical Systems," in *2022 IEEE International Systems Conference (SysCon)*, IEEE, Apr. 2022, pp. 1–8. doi: 10.1109/SysCon53536.2022.9773806.

[40] F. Almada-Lobo, "The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES)," *Journal of Innovation Management*, vol. 3, no. 4, pp. 16–21, Jan. 2016, doi: 10.24840/2183-0606_003.004_0003.

[41] S. Mantravadi and C. Møller, "An Overview of Next-generation Manufacturing Execution Systems: How important is MES for Industry 4.0?," *Procedia Manuf*, vol. 30, pp. 588–595, 2019, doi: 10.1016/j.promfg.2019.02.083.

[42] C. Kühnel, *Embedded Linux mit dem Raspberry Pi: für Ein- und Umsteiger*. Skript Verlag Kühnel, 2013. [Online]. Available: https://books.google.pt/books?id=VdqZAAAAQBAJ

[43] E. O'Connell, D. Moore, and T. Newe, "Challenges Associated with Implementing 5G in Manufacturing," *Telecom*, vol. 1, no. 1, pp. 48–67, Jun. 2020, doi: 10.3390/telecom1010005.

[44] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nat Nanotechnol*, vol. 15, no. 7, pp. 529–544, Jul. 2020, doi: 10.1038/s41565-020-0655-z.

[45] A. T. Rizvi, A. Haleem, S. Bahl, and M. Javaid, “Artificial Intelligence (AI) and Its Applications in Indian Manufacturing: A Review,” 2021, pp. 825–835. doi: 10.1007/978-981-33-4795-3_76.

[46] i4 verse inc., “Future of Manufacturing Execution Systems (MES),” Dover, Mar. 2024.

[47] Camunda, “Camunda Modeler.” Accessed: Jun. 24, 2024. [Online]. Available: https://camunda.com/platform/modeler/

[48] Open Source, “What is open source?” Accessed: Jun. 16, 2024. [Online]. Available: https://opensource.com/resources/what-open-source

[49] The Nukon Team, “Is it right for you?: The pros and cons of open source manufacturing software,” Nukon .

[50] G. J. Badros, “JavaML: a markup language for Java source code,” *Computer Networks*, vol. 33, no. 1–6, pp. 159–177, Jun. 2000, doi: 10.1016/S1389-1286(00)00037-2.

[51] Github, “Where the world builds software.” Accessed: Apr. 12, 2022. [Online]. Available: https://github.com/about

[52] A. Kocher, L. M. V. Da Silva, and A. Fay, “Modeling and Executing Production Processes with Capabilities and Skills using Ontologies and BPMN,” in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2022, pp. 1–8. doi: 10.1109/ETFA52439.2022.9921564.

[53] DIN Media GmbH, “DIN 8580:2022-12,” Dec. 2022, *DIN Media GmbH, Berlin*. doi: 10.31030/3217872.

[54] A. Mayr *et al.*, “Machine Learning in Production – Potentials, Challenges and Exemplary Applications,” *Procedia CIRP*, vol. 86, pp. 49–54, 2019, doi: 10.1016/j.procir.2020.01.035.

[55] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer, “Thirteen years of SysML: a systematic mapping study,” *Softw Syst Model*, vol. 19, no. 1, pp. 111–169, Jan. 2020, doi: 10.1007/s10270-019-00735-y.

[56] A. W. Wymore, *Model-Based Systems Engineering*. CRC Press, 2018. doi: 10.1201/9780203746936.

[57] O. Loyola-Gonzalez, “Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View,” *IEEE Access*, vol. 7, pp. 154096–154113, 2019, doi: 10.1109/ACCESS.2019.2949286.

[58] L. E. de S. Amorim, S. Erdweg, G. Wachsmuth, and E. Visser, “Principled syntactic code completion using placeholders,” in *Proceedings of the 2016 ACM SIGPLAN*

International Conference on Software Language Engineering, New York, NY, USA: ACM, Oct. 2016, pp. 163–175. doi: 10.1145/2997364.2997374.

[59] A. Hayward, M. Rappl, and A. Fay, "A SysML-based Function-Centered Approach for the Modeling of System Groups for Collaborative Cyber-Physical Systems," in *2022 IEEE International Systems Conference (SysCon)*, IEEE, Apr. 2022, pp. 1–8. doi: 10.1109/SysCon53536.2022.9773806.

[60] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás, "JCLEC: a Java framework for evolutionary computation," *Soft comput*, vol. 12, no. 4, pp. 381–392, 2008, doi: 10.1007/s00500-007-0172-0.

[61] D. Wing, "Network Address Translation: Extending the Internet Address Space," *IEEE Internet Comput*, vol. 14, no. 4, pp. 66–70, Jul. 2010, doi: 10.1109/MIC.2010.96.

[62] A. Kocher *et al.*, "Automating the Development of Machine Skills and their Semantic Description," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2020, pp. 1013–1018. doi: 10.1109/ETFA46521.2020.9211933.

[63] Node.js, "Node.js Releases," Node.js. Accessed: Mar. 12, 2025. [Online]. Available: https://nodejs.org/en/about/previous-releases#nodejs-releases

[64] Josh Collinsworth, "What the Heck Does 'npm' Mean?" Accessed: May 03, 2022. [Online]. Available: https://css-tricks.com/a-clear-definition-of-npm-and-what-it-does/

[65] A. Abane, A. Battou, M. Merzouki, and T. Zhang, "A Smart Network Repository Based on Graph Database," 2024, pp. 75–86. doi: 10.1007/978-3-031-52426-4_6.

[66] G. C. Murphy, M. Kersten, and L. Findlater, "How are Java software developers using the Eclipse IDE?," *IEEE Softw*, vol. 23, no. 4, pp. 76–83, Jul. 2006, doi: 10.1109/MS.2006.105.

[67] S. Profanter, K. Dorofeev, A. Zoitl, and A. Knoll, "OPC UA for plug & produce: Automatic device discovery using LDS-ME," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2017, pp. 1–8. doi: 10.1109/ETFA.2017.8247569.

[68] A. Dey, A. Fekete, and U. Rohm, "REST+T: Scalable Transactions over HTTP," in *2015 IEEE International Conference on Cloud Engineering*, IEEE, Mar. 2015, pp. 36–41. doi: 10.1109/IC2E.2015.11.

[69] Unified Automation, "UaExpert—A Full-Featured OPC UA Client." Accessed: Mar. 13, 2025. [Online]. Available: https://www.unified-automation.com/products/development-tools/uaexpert.html?gad_source=1&cHash=b272c9b8ea259f82a8f1948d3063a68b

[70] Z. Nakutis, V. Deksnys, I. Jarusevicius, V. Dambrauskas, G. Cincikas, and A. Kriauceliunas, “Round-Trip Delay Estimation in OPC UA Server-Client Communication Channel,” *Elektronika ir Elektrotechnika*, vol. 22, no. 6, Dec. 2016, doi: 10.5755/j01.eie.22.6.17229.

[71] Eclipse Foundation, “OSGI Working Group.” Accessed: Oct. 11, 2022. [Online]. Available: https://www.osgi.org/about/

[72] L. Pacioli, *Su[m]ma de arithmetica geometria proportioni [et] proportionalita*. [Venice: Paganinus de Paganinis, 1494. doi: 10.5479/sil.440357.39088007406663.

[73] C. Liao and C. Shyu, “An Analytical Determination of Lead Time with Normal Demand,” *International Journal of Operations & Production Management*, vol. 11, no. 9, pp. 72–78, Sep. 1991, doi: 10.1108/EUM0000000001287.

Anexos

Anexo A - Instalação do *SkillMex* e demais ferramentas

Numa fase inicial, ainda antes da instalação do próprio *SkillMex*, é necessária a instalação da ferramenta node.js, numa versão superior à 14. A versão que instalada foi a 16.14.0.

Adicionalmente, é necessária a instalação de uma ferramenta de criação de repositórios para guardar a informação produzida. O *SkillMex* utiliza o *GraphDB* que pode ser descarregado a partir da ligação, https://www.ontotext.com/products/graphdb/graphdb-free/. No *GraphDB*, deve ser criado um repositório que o *SkillMex* possa utilizar, nesta fase obrigatoriamente com o nome 'test-repo'. Qualquer designação diferente da anterior, retornará erros posteriores. Porém, um repositório vazio de pouco servirá, uma vez que este não reconhece a ontologia inerente ao ambiente *SkillMex*. Portanto, deve ser descarregado um ficheiro denominado por 'CaSkMan_merged_v4.6.0.ttl' no seguinte sítio: https://github.com/CaSkade-Automation/CaSkMan/releases/tag/v4.6.0. Este ficheiro deve ser importado no *GraphDB* no campo 'Import (left sidebar)'>'RDF'.

Após os procedimentos anteriores, é possível fazer o *download* do *SkillMex* (instalou-se a versão 2.0.2), através da seguinte ligação: https://github.com/CaSkade-Automation/CaSkade-MES/releases/tag/v2.0.2. Após este *download*, devem ser abertas duas janelas através do comando cmd: numa será instalado o *frontend* e noutra o *backend*. Dentro de cada uma das janelas anteriores, deve navegar-se , através do comando 'cd' (*change directory*) até chegar à localização do frontend descarregado e no *backend* procede-se da mesma forma. Dentro de cada janela, após estar na localização especificada, deve ser efetuada a instalação, através da introdução do comando 'npm i'. Após a execução, este comando pede para resolver potenciais conflitos. Aqui deve ser apenas efetuado o comando 'npm audit fix', nunca devendo utilizar a extensão '-force'. Vão surgir erros no final, mas para o que se pretende, não é necessário resolver estes problemas. Deste modo, em ambas as janelas já é possível executar o comando 'npm run start:dev'. A operação anterior vai devolver uma janela referente ao próprio *SkillMex*, aberta pelo *frontend*. Nesta janela, é possível averiguar se abriu corretamente. Na janela aberta do *backend*, também devem surgir mensagens referentes à correta abertura do mesmo. A janela do *SkillMex* deve reconhecer automaticamente o repositório criado. Na pasta 'SkillMex-Configuration' do menu

principal, existe o local de seleção de repositório. Porém esta ferramenta surge inoperacional, assim como a colocação do nome do utilizador e respetiva palavra-passe, sendo campos totalmente inoperacionais de momento.

Por outro lado, é necessária a abertura do *SkillUp-runtime* para possibilitar a introdução de *skills* e módulos criados via *Eclipse* para o âmbito *SkillMex*. Para isto, deve ser descarregado o ficheiro 'skillup-runtime_v2.0.3.zip' através da seguinte ligação: https://github.com/CaSkade-Automation/SkillUp/releases/tag/v2.0.3. Para colocar a correr, basta copiar e colar os ficheiros.jar referentes às *skills* e módulos para a pasta 'include' do runtime e correr o ficheiro 'startFelix.bat'. O *SkillUp* deve fazer a introdução dos ficheiros automaticamente, não é necessário fazer nenhum passo adicional.

Para ser possível lidar com diagramas *BPMN*, embora este campo careça de bastantes desenvolvimentos, é necessário ativar um 'BPMN engine' que pode ser descarregado na seguinte ligação: https://camunda.com/download/#download-other-menu. Apesar de a pasta do *GitHub* do *SkillMex*, indicar o uso da ferramenta 'camunda-bpmn-run', não foi possível estabelecer ligação. Como solução conseguiu-se colocar em funcionamento o 'camunda-bpmn-tomcat', uma ferramenta idêntica. Em termos de versões, apenas foi possível utilizar a v7.18.0.

Anexo B - Metodologia de síntese de uma *skill* ou módulo

Após a instalação da ferramenta *SkillMex*, é necessária a instalação do software *Eclipse*, de modo a possibilitar a síntese de um módulo ou *skill*. Primeiramente, foi criado um projeto do tipo *Maven* do qual se obtém um ficheiro do tipo *pom.xml*, com código apresentado na Figura 38. De um modo geral, é possível acompanhar a metodologia que se encontra no sítio do *GitHub* do *SkillUp*.

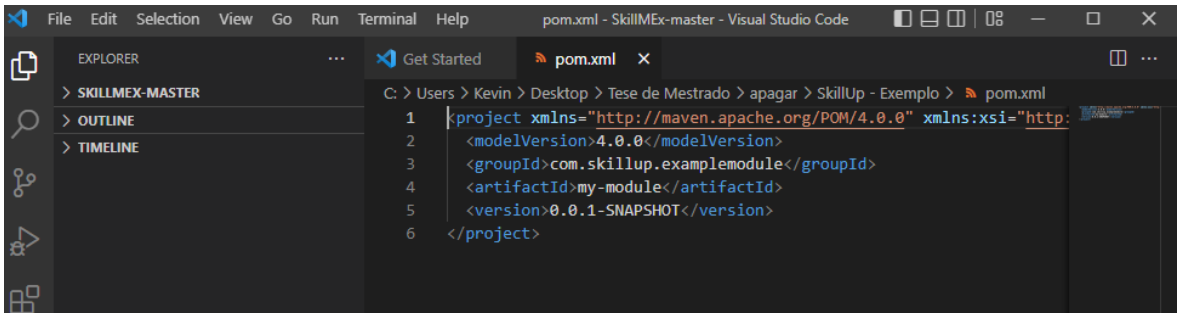

Figura 38.: Exemplo de um ficheiro resultante de um novo projeto Maven.

Porém, é necessária uma dependência do pacote do *SkillUp* que permite a utilização de anotações, necessárias para reconhecer as nomenclaturas dos módulos e afins. Deste modo, foi necessária a adição do código constante da *Figura 39*.

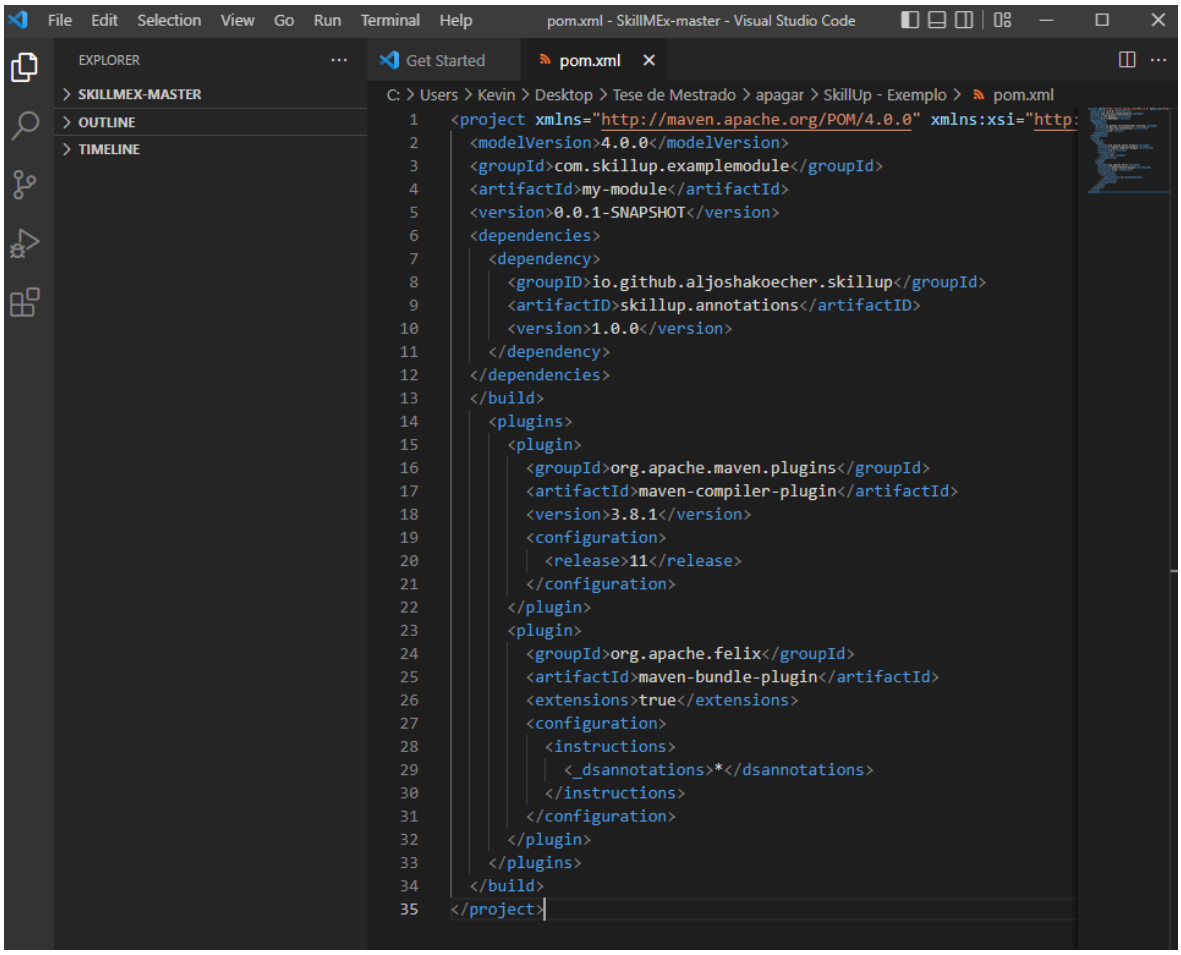

Figura 39.: Código resultante da adição das dependências necessárias.

Posteriormente, é possível a síntese de um módulo, a criação de um novo pacote denominado ‘modulo_experimental’, o qual se associou uma classe ‘ModuloExperimental_Class’ (Figura 43). Estas nomenclaturas servem apenas o propósito de exemplificação.

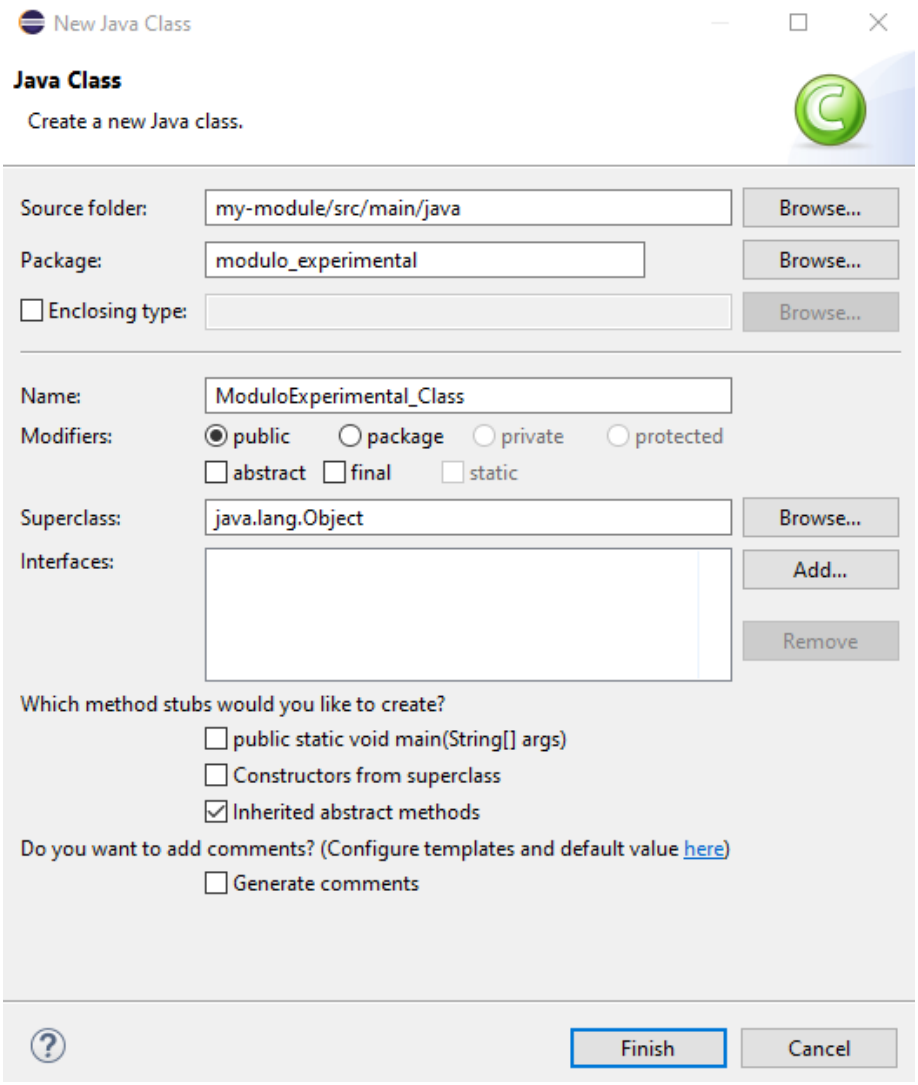

Figura 40.: Exemplo da classe *Java*.

Na criação da classe, é necessária a adição do código seguinte (Figura 41) ao ficheiro da classe, utilizando o termo característico '@Module', o qual pode ser adaptado de acordo com as nomenclaturas desejadas.

```
@Module(moduleIri = "https://my-example-module.com/#ExampleModule", description = "MyModule")
public class ExampleModule {
}
```

Figura 41.: Exemplo do campo '@Module' na produção de um módulo.

Após este processo, é necessário construir o módulo propriamente dito, seguindo-se o processo de colocar a correr o módulo. No separador 'Run configurations', é necessário

colocar no campo ‘Goals’ a opção ‘clean install’, correndo o campo ‘Run’ de seguida (Figura 42).

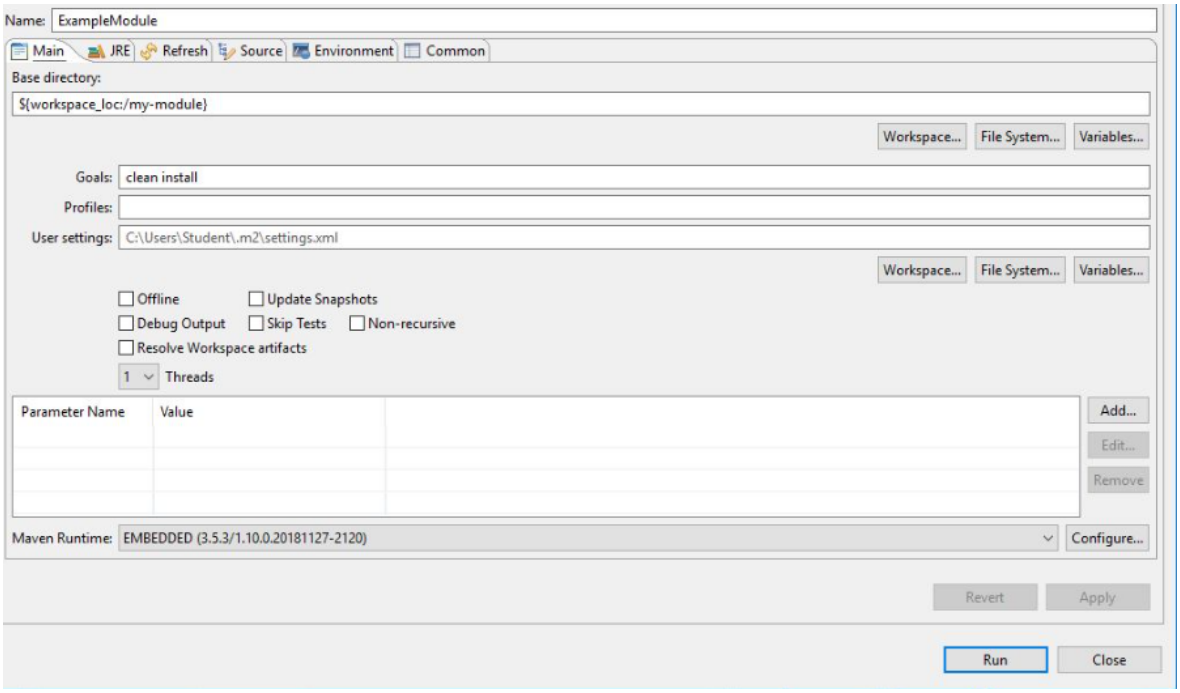

Figura 42.: Instalação de um módulo no *Eclipse*.

Seguindo o procedimento, este processo deve devolver um ficheiro to tipo ‘*.jar’* a utilizar futuramente no *SkillUp*. Este ficheiro surge somente quando ocorre a mensagem no *Eclipse* de ‘Build Success’ (Figura 43).

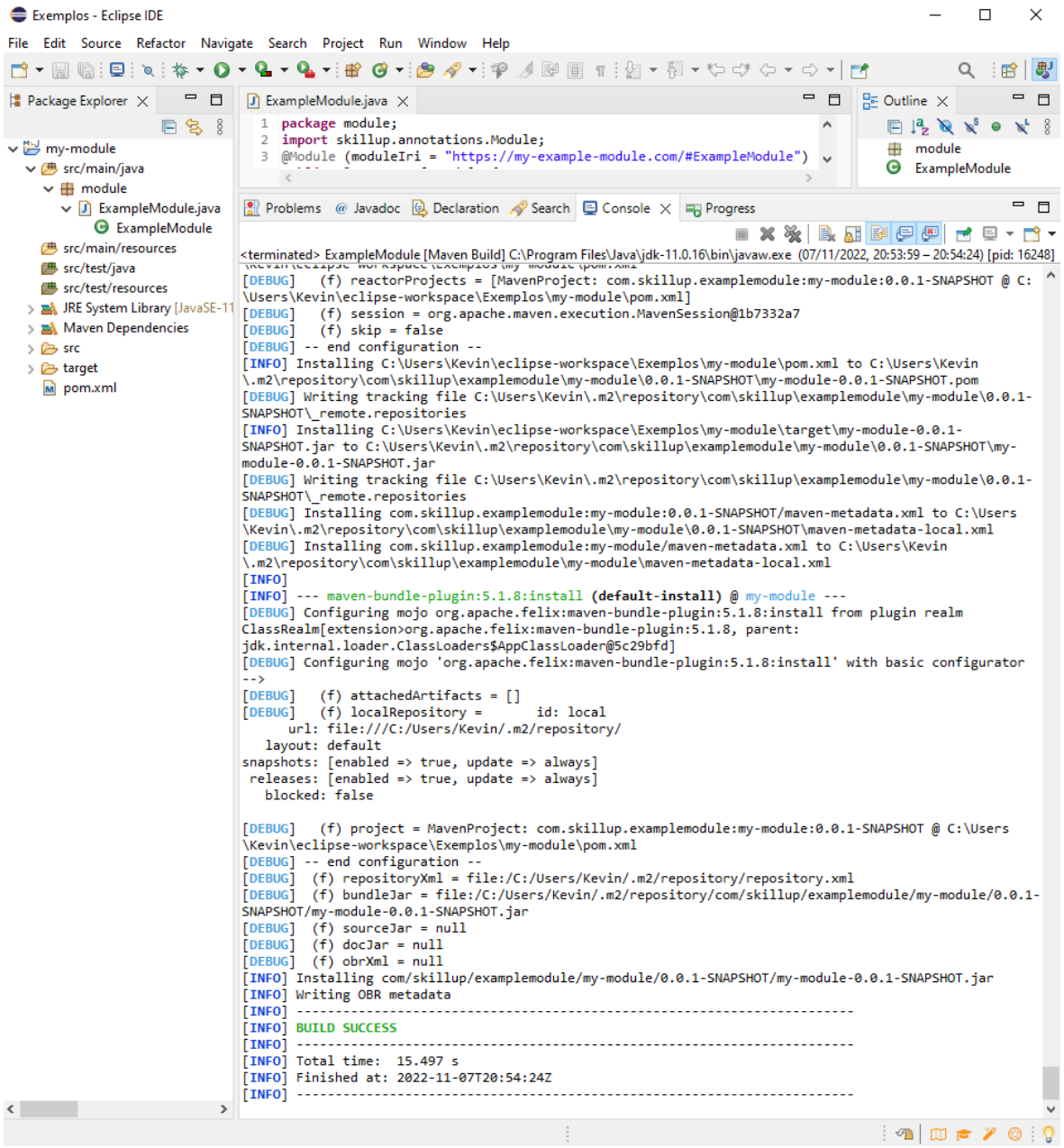

Figura 43.: Janela do *Eclipse* após a resolução dos erros.

Após a obtenção dos ficheiros anteriores, é necessária a utilização do *SkillUp Runtime*, uma plataforma que após a inclusão dos ficheiros criados no Eclipse na pasta 'Include' do runtime, ativam os módulos e as *skills* previamente introduzidas nesse local. Em termos teóricos, isto ocorreria perfeitamente, mas podem surgir diversos erros após a colocação a correr do 'startFelix.bat'. Este conceito está inerentemente relacionado com o conceito de *OSGI* que corresponde a uma iniciativa em código aberto para produzir especificações de software, o que permite o desenvolvimento e gestão dos elementos de integração do servidor com aplicações que com este interagem. [71]

Anexo C - Potenciais etapas de resolução de problemas

A janela devolvida pelo *backend*, após a abertura deste pela primeira vez, indica que existe algum problema com a origem dos parâmetros *HttpModule* e *HttpService* presentes na pasta do '@nestjs'. Para solucionar este problema, procede-se à instalação do '@nestjs/axios' e, por outro lado, alterou-se a localização desses parâmetros, de '@nestjs/common' para '@nestjs/axios', nos ficheiros das seguintes localizações.

- C:\Users\(…)\SkillMEx-master\backend\dist\util -> opc-ua-state-monitor.service.d.ts
- C:\Users\(…)\SkillMEx-master\backend\dist\util -> LocalHttpRequestConfigService.d.ts
- C:\Users\(…)\SkillMEx-master\backend\src\util -> opc-ua-state-monitor.module.ts
- C:\Users\(…)\SkillMEx-master\backend\src\util -> LocalHttpRequestConfigService.ts
- C:\Users\(…)\SkillMEx-master\backend\src\util -> opc-ua-state-monitor.module.ts.

Obviamente que as localizações anteriores devem ser adaptadas à localização da instalação em particular. Após este procedimento, o processo de instalação ainda se encontra incompleto. No *frontend*, ocorrem também erros relativos a dependências do *Angular* e do *CommonJS*. Após uma pesquisa acerca de problemas idênticos na internet verifica-se que a sua origem remete para a falta de algumas dependências No ficheiro 'angular.json' do frontend deveriam constar a totalidade dos nomes dos pacotes de dependências. Por isso, a totalidade destas dependências têm de ser adicionados manualmente. Por conseguinte, verifica-se que com a colocação dos nomes dos pacotes do frontend aos quais está associada aquela mensagem a amarelo no campo do 'angular.json', denominado

'AllowedCommonJsDependencies', soluciona-se o problema.

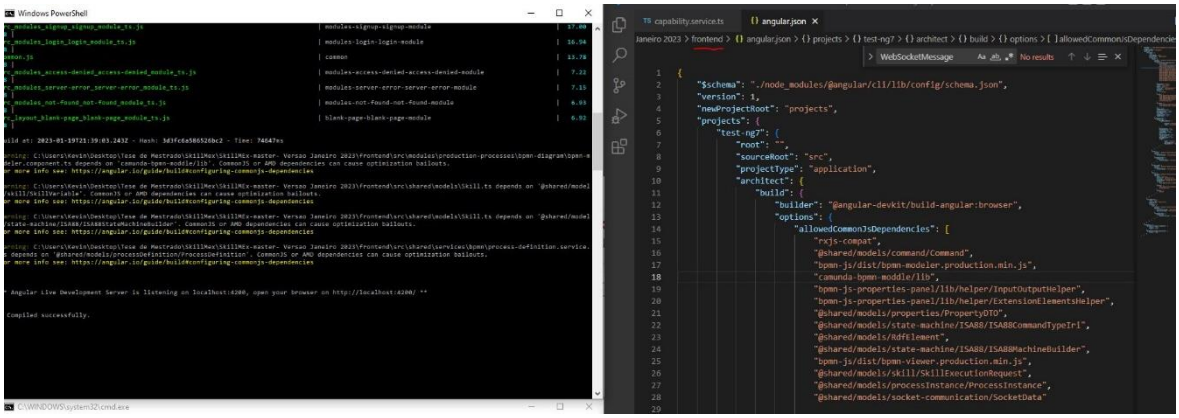

Figura 44.: Frontend com os problemas que surgiram (à esquerda) e o ficheiro 'angular.json' (à direita).

Anexo D - Especificação dos módulos do projeto

Módulo do Armazém Automático ('AA-0.0.1-SNAPSHOT.jar')

ModuleAA.java

```
1  package moduleaa;
2  import skillup.annotations.Module;
3  @Module(moduleIri = "https://my-example-module.com/#moduleAA", description = "MyModuleAA")
4  public class ModuleAA {
5
6  }
7
```

Figura 45.: Corpo do módulo do Armazém Automático ('AA-0.0.1-SNAPSHOT.jar')

Módulo da máquina CNC ('CNC-0.0.1-SNAPSHOT.jar')

ModuleCNC_.java

```
1  package modulecnc_;
2  import skillup.annotations.Module;
3  @Module(moduleIri = "https://my-example-module.com/#moduleCNC", description = "MyModuleCNC")
4  public class ModuleCNC_ {
5
6  }
```

Figura 46.: Corpo do módulo do Módulo da máquina CNC ('CNC-0.0.1-SNAPSHOT.jar')

Moduledp.java

```java
package moduledp;
import skillup.annotations.Module;
@Module(moduleIri = "https://my-example-module.com/#moduleDP", description = "MyModule")
public class Moduledp {

}
```

Figura 47.: Corpo do módulo do dispensador de paletes

Módulo do dispensador de produto bruto (‘PB-0.0.1-SNAPSHOT.jar’)

Modulepb.java

```java
package modulepb;
import skillup.annotations.Module;
@Module(moduleIri = "https://my-example-module.com/#modulePB", description = "MyModule-pb")
public class Modulepb {

}
```

Figura 48.: Corpo do módulo do dispensador de produto

Módulo do robô (‘R-0.0.1-SNAPSHOT.jar’)

ModuleR.java

```java
package moduler;
import skillup.annotations.Module;
@Module(moduleIri = "https://my-example-module.com/#moduleR", description = "MyModuleR")
public class ModuleR {

}
```

Figura 49.: Corpo do módulo do robô

Anexo E - Especificação das *skills* do projeto

Skill do tapete - ‘skill-Libertar-Navete-0.0.1-SNAPSHOT.jar

ModuleT.java | Skill_Libertar_Navete.java

```java
package skill_Libertar_Navete;

import skillup.annotations.*;

@Skill(skillIri = "https://my-example-skill.com/#Skill-Libertar_Navete", capabilityIri = "https://my-example-

public class Skill_Libertar_Navete {
    @SkillParameter(isRequired = true, option = { "1", "2", "3" })
    int Station;

    @SkillParameter(isRequired = true, option = { "1", "2", "3" })
    int Navete;

    @SkillParameter(isRequired = true)
    boolean ativador;

    @SkillOutput(isRequired = true)
    int Executado;

    @StateMachine // It assures that unexpected problems on SkillMex won't occur.
    Isa88StateMachine stateMachine;

    @Starting
    public void inicio() {
        Executado = 0;
    }

    @Execute
    public void execute() {
        if (!ativador) {
            Executado = 1;
        }
        else {
        }
    }

    @Completing
    public void complete() {
        Executado = 0;
    }
}
```

Figura 50.: Corpo da skill-Adia-Navete.

Skill do tapete - ‘skill-Adia_Navete-0.0.1-SNAPSHOT.jar’

ModuleT.java | skill_adia_navete.java

```java
package skill_Adia_Navete;

import skillup.annotations.*;

@Skill(skillIri = "https://my-example-skill.com/#Skill-Adia-Navete", capabilityIri = "https://my-example

//Skill para mover peça de x para y - TAPETE

public class skill_adia_navete {
    @SkillParameter(isRequired = true, option = { "1", "2", "3" })
    int Station;

    @SkillParameter(isRequired = true, option = { "1", "2", "3" })
    int Navete;

    @SkillParameter(isRequired = true)
    boolean ativador;

    @SkillOutput(isRequired = true)
    int Executado;

    @StateMachine // It assures that unexpected problems on SkillMex won't occur.
    Isa88StateMachine stateMachine;

    @Starting
    public void inicio() {
        Executado = 0;
    }

    @Execute
    public void execute() {
        if (!ativador) {
            Executado = 1;
        }
        else {
        }
    }

    @Completing
    public void complete() {
        Executado = 0;
    }
}
```

Figura 51.: Corpo da skill- Libertar-Navete.