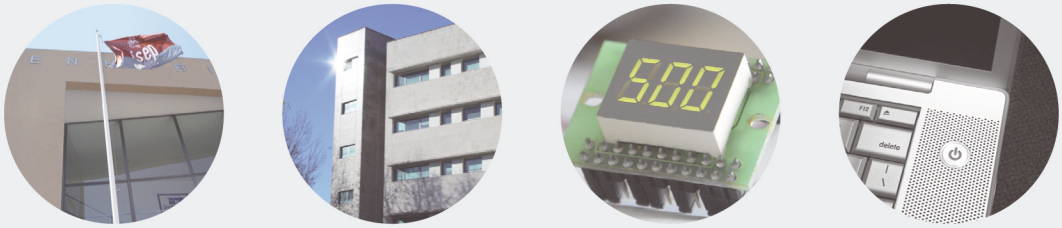
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



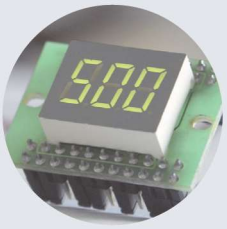
Integração Automática de Modelos de Sistemas de Sinalização Ferroviária em Sistemas Embebidos

JOSÉ MÁRIO TAVARES CORREIA
julho de 2021

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Integração Automática de Modelos de Sistemas de Sinalização Ferroviária em Sistemas Embebidos

JOSÉ MÁRIO TAVARES CORREIA
Junho de 2021

POLITÉCNICO
DO PORTO

INTEGRAÇÃO AUTOMÁTICA DE MODELOS DE SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA EM SISTEMAS EMBEBIDOS

José Mário Tavares Correia

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

2021

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: José Mário Tavares Correia, Nº 1160533, 1160533@isep.ipp.pt

Orientação científica: Manuel Gradim de Oliveira Gericota, mgg@isep.ipp.pt

Empresa: EFACEC Engenharia e Sistemas, S.A.

Supervisão: João Marques Martins, joao.martins@efacec.com

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

20 de junho de 2021

Agradecimentos

Em primeiro lugar, dirijo o meu agradecimento à EFACEC pela oportunidade de desenvolver este trabalho, em particular, à pessoa do Engenheiro João Marques Martins pela sua total disponibilidade, orientação científica, interesse e apoio.

Ao meu orientador, Professor Doutor Manuel Gradim de Oliveira Gericota, pela sua total disponibilidade, orientação científica e os seus conselhos.

Aos meus amigos, que me auxiliaram nos momentos mais difíceis e pelo apoio ao longo desta caminhada.

Com um carinho especial, quero agradecer à minha família pelo apoio incondicional ao longo desta etapa e por tudo aquilo que me proporcionaram, se esta caminhada teve um ponto de partida a eles o devo.

Por último, a todos aqueles que não foram mencionados anteriormente, que contribuíram não só, para o meu crescimento pessoal, mas também para o desenvolvimento deste trabalho.

O meu obrigado.

Resumo

Um sistema de sinalização ferroviário é um sistema que permite a gestão segura do tráfego ferroviário. Através destes sistemas é possível proteger os veículos ferroviários de eventuais colisões e descarrilamentos, fruto do estabelecimento de itinerários evitando movimentos conflituosos, permitindo assim o movimento dos veículos ferroviários em segurança. O encravamento eletrónico permite uma menor dependência do ser humano para executar determinadas tarefas, sendo possível implementar medidas de mitigação contra acidentes e alcançar níveis de integridade de segurança mais elevados.

A Unidade de Transportes, da empresa EFACEC, dedica-se ao desenvolvimento de sistemas de sinalização ferroviários e tem a necessidade de automatizar o processo de integração dos seus produtos genéricos certificados com o nível de integridade de segurança 4 de acordo com a norma EN 50128, em uma aplicação específica, o que originou o trabalho apresentado nesta tese. Este processo de automatização visa melhorar a eficiência do processo de integração e torná-lo menos propenso a erros. De forma a suportar a automatização da componente de integração deste processo, foi criada a ferramenta chamada KCG Adapter To SILWorX, a qual tem como objetivo a integração dos modelos desenvolvidos em SCADE Suite, permitindo executar esses modelos num *Programmable Logic Controller*, o HIMatrix, programado através da ferramenta SILWorX.

A ferramenta desenvolvida pode ser subdividida em três componentes principais. Um deles é o processo que diz respeito à interface gráfica da ferramenta, desenvolvido através das funcionalidades do Windows Presentation Foundation no *software* Visual Studio 2019. Os restantes processos são afetos à aquisição de informação proveniente do projeto modelado no SCADE Suite. Isto só é possível devido às funcionalidades disponibilizadas pela *Application Programming Interface* Java do Scade, dedicada para o *software* Eclipse. Os processos desenvolvidos através do Eclipse resultam em ficheiros executáveis *Java Archive*. Estes ficheiros são responsáveis por gerar ficheiros XML que contêm informação estruturada acerca do projeto SCADE Suite e das configurações realizadas autonomamente. Os ficheiros XML são interpretados pelo processo da interface gráfica o que permite apresentar a informação ao utilizador.

De um modo geral, através do projeto apresentado nesta tese foi possível desenvolver uma ferramenta capaz de interpretar o conteúdo de projetos modelados no SCADE Suite e integrar autonomamente estes modelos nos controladores, de acordo com o SILworX. A utilização desta ferramenta permite substituir o processo de integração manual e torná-lo mais eficiente.

Palavras-Chave

Ferrovia, Sistema sinalização Ferroviário, Encravamento, SIL, SCADE Suite, KCG, SILworX, Eclipse, WPF, XML.

Abstract

A railway signalling system is a system that allows to control the railway traffic safely. Through these systems, it is possible to establish routes, protecting rail vehicles from potential collisions and derailments, allowing to avoid conflicting movements and ensure the safety of the rail vehicle. The electronic interlocking relies less on humans to perform determined tasks, enabling the implementation of mitigation measures against accidents, achieving higher safety integrity levels.

The *Unidade de Transportes* of the company EFACEC develops railway signaling systems. It felt the need to automate the integration process of its generic products certified with safety integrity level 4 according to EN 50128, for a specific application, which originated the work presented in this thesis. This automation process aims to improve the efficiency of the integration process and make it less error-prone. To automate the integration process, it was developed a tool called KCG Adapter To SILWorX, which aims to integrate the models developed in SCADE Suite in a HIMatrix Programmable Logic Controller. This tool provides the possibility to run the models developed in SCADE Suite in the Programmable Logic Controllers, programmed through the SILWorX tool.

The tool developed can be subdivided into three main processes. One of them is the process that concerns the graphical interface of the tool developed through the Windows Presentation Foundation functionalities in Visual Studio 2019 software. The remaining processes regard the acquisition of information coming from the project modeled in the SCADE Suite. That is only possible due to the functionalities provided by SCADE's Java *Application Programming Interface*, dedicated for Eclipse software. The processes developed through Eclipse result in executable Java Archive files. These files are responsible for generating XML files that contain structured information about the SCADE Suite project and the configurations performed autonomously. The graphical interface process interprets the XML files, allowing to present information to the user.

In general, this thesis project made it possible to develop a tool capable of interpreting the content of projects modeled in SCADE Suite and autonomously integrate these models in

the controllers, according to SILworX. The use of this tool allows to replace the manual integration process and make it more efficient.

Keywords

Railway, Railway Signalling System, Interlocking, SIL, SCADE Suite, KCG, SILworX, Eclipse, WPF, XML.

Índice

AGRADECIMENTOS ..... **I**

RESUMO ..... **III**

ABSTRACT ..... **V**

ÍNDICE ..... **VII**

ÍNDICE DE FIGURAS ..... **IX**

ÍNDICE DE TABELAS ..... **XI**

ACRÓNIMOS ..... **XIII**

1. INTRODUÇÃO **1**

1.1. CONTEXTUALIZAÇÃO 2
1.2. OBJETIVOS 4
1.3. ORGANIZAÇÃO DO RELATÓRIO 4

2. ESTADO DA ARTE **7**

2.1. FERROVIA 7
2.2. SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA 11
2.3. SEGURANÇA NOS SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA 18
2.4. TRABALHOS REALIZADOS NA ÁREA 24

3. DESCRIÇÃO DO PROJETO **27**

3.1. SCADE SUITE 27
3.2. SILWORX 30
3.3. DESCRIÇÃO DETALHADA DO PROJETO 32

4.* PLANIFICAÇÃO DO *SOFTWARE **39**

4.1. ARQUITETURA GERAL DO *SOFTWARE* 39
4.2. PRINCÍPIO DE FUNCIONAMENTO DO *SOFTWARE* 41

5. IMPLEMENTAÇÃO **49**

5.1. MANIPULAÇÃO DO PROJETO SCADE SUITE 49
5.2. INTERFACE GRÁFICA DO KATOS 50
5.3. VERIFICAÇÕES E RECOLHA DE DADOS DOS FICHEIROS JAR 52
5.4. FICHEIRO DE INSTALAÇÃO 55

6. RESULTADOS 57

7. CONCLUSÕES 63

REFERÊNCIAS DOCUMENTAIS 66

Índice de Figuras

Figura 1 – Comprimento da ferrovia, em km, de diversos países da união europeia em 2019 [10]. . 8
Figura 2 – Modelos operacionais para o tráfego ferroviário de alta velocidade [12]..... 9
Figura 3 – Número de acidentes ferroviários na União Europeia [16]..... 10
Figura 4 – Cabine sinalizadora, construída em 1881 na estação *King's Cross* [19]..... 12
Figura 5 – Armário de controlo implementado na solução XSafe, desenvolvido pela EFACEC [20]. 12
Figura 6 – Exemplos de acidentes ferroviários e métodos de prevenção [17]. 13
Figura 7 – Sequência de blocos orientados entre duas estações [17]. 14
Figura 8 – Exemplo de um itinerário e os seus constituintes [21]..... 15
Figura 9 – Troço de uma infraestrutura ferroviária [21]. 16
Figura 10 – Princípio geográfico [22]. 17
Figura 11 – Comparação entre o esforço técnico na implementação do princípio centralizado e geográfico, em sistemas de encravamento baseados em relés [17]. 18
Figura 12 – Principais normas aplicadas no desenvolvimento de sistemas ferroviárias [31]. 20
Figura 13 – Estruturas organizacionais para o desenvolvimento de *software*, de acordo com a norma EN 50128 [8]. 21
Figura 14 – Modelo de projeto cascata [8]. 22
Figura 15 – Modelo de projeto em V [8]. 23
Figura 16 – Troca de informação recorrendo ao railML[34]. 25
Figura 17 – Ferramentas do SCADE Suite [38]. 29
Figura 18 – Exemplo de meta modelo do SCADE Suite [39]..... 30
Figura 19 – Diagrama de Blocos do HIMatrix F35 03 [47]..... 31
Figura 20 – Processo de desenvolvimento de um SSF. 32
Figura 21 – Estrutura de um Projeto SCADE Suite 33
Figura 22 – *Inputs* e *Outputs* de uma função..... 33
Figura 23 – Estrutura de dados. 34
Figura 24 – Subdiretórios dos vários modelos que contêm estruturas. 35
Figura 25 – Arquitetura geral do *Software* do KATOS..... 40
Figura 26 – Fluxograma do processo GUI. 41
Figura 27 – Fluxograma do processo pré-preenchimento. 42
Figura 28 – Fluxograma do processo manipulação de dados. 44

Figura 29 – Fluxograma da função desfragmentação dos IO 45
Figura 30 – Fluxograma da função processamento do tipo composto. 46
Figura 31 – Interface gráfica do KATOS. 57
Figura 32 – Exemplo de seleção de ficheiro ou diretório. 58
Figura 33 – Ficheiro XML gerado pelo processo de pré-preenchimento. 58
Figura 34 – Demonstração da aquisição dos dados utilizados na geração do KCG. 59
Figura 35 – Demonstração dos operadores identificados pelo KATOS. 59
Figura 36 – Demonstração de prefixo incompatível. 60
Figura 37 – Demonstração da verificação do conteúdo do Ficheiro do SILworX. 60
Figura 38 – Demonstração do aviso acerca dos ficheiros expectáveis. 60
Figura 39 – Relatório da geração do KATOS. 61
Figura 40 – Exemplo das tabelas criadas pelo KATOS. 61

Índice de Tabelas

Tabela 1 – Tabela de controlo de itinerários para o troço exposto na Figura 9 [21]. 16
Tabela 2 - Níveis SIL [23]..... 19
Tabela 3 – Requisitos do *software* [8]..... 23
Tabela 4 – IO utilizados no exemplo. 35
Tabela 5 – Estrutura utilizada no exemplo..... 35
Tabela 6 – Exemplo de Desfragmentação IO..... 36

Acrónimos

API	–	*Application Programming Interface*
AWT	–	*Abstract Window Toolkit*
CMD	–	*Command Shell*
CSV	–	*Comma-separet Values*
EPS	–	*EFACEC Power Solutions*
GUI	–	*Graphical User Interface*
HTML	–	*HyperText Markup Language*
IO	–	*Inputs/Outputs*
JAR	–	*Java Archive*
JFC	–	*Java Foundation Classes*
JRE	–	*Java Runtime Environment*
KATOS	–	*KCG adapter to SILworX*
KCG	–	*Qualified Code Generator*
OS	–	Sistema Operativo
PLC	–	*Programmable Logic Controllers*
railML	–	*railway Markup Language*
RAMS	–	*Reliability, Availability, Maintainability and Safety*
RAS	–	*Railway Automation Suite*

SCADE – *Safety Critical Application Development Environment*

SIL – *Safety Integrity Level*

SSF – Sistema de sinalização ferroviário

SSI – *Solid State Interlocking*

TCL – *Tool Command Language*

UE – União Europeia

UML – *Unified Modeling Language*

UNIFE – *The European Rail Industry*

uTRP – Unidade de Transportes

WPF – *Windows Presentation Foundation*

XAML – *eXtensible Application Markup Language*

XML – *eXtensible Markup Language*

1. INTRODUÇÃO

Um sistema de sinalização ferroviário (SSF) é um sistema que permite a gestão segura do tráfego ferroviário. Através destes sistemas é possível proteger os veículos ferroviários de eventuais colisões e descarrilamentos, o que permite estabelecer itinerários de modo a proporcionar o movimento dos veículos ferroviários em segurança. Atualmente os SSF utilizam mecanismos de encravamento eletrónico. O encravamento eletrónico permite uma menor dependência do ser humano para executar determinadas tarefas, sendo possível implementar medidas de mitigação contra acidentes e alcançar níveis de integridade de segurança (SIL) mais elevados. Existem empresas no mercado dedicadas a desenvolver soluções para este tipo de sistemas ferroviários, sendo a EFACEC uma empresa com grande potencial nesta área.

A EFACEC é uma empresa Portuguesa, fundada em 1948, à data com o nome de Empresa Fabril de Máquinas Eléctricas, SARL. Contudo, a sua origem remota a duas outras empresas. A Moderna, fundada em 1905, que por sua vez, deu origem à Electro-Moderna, Lda. em 1921. Esta última dedicava-se à produção de motores, geradores, transformadores e acessórios elétricos. Estes equipamentos viriam também a ser produzidos pela EFACEC os quais, inicialmente, alavancaram o seu crescimento. Atualmente a EFACEC é um grande exportador e marca presença em mais de 65 países. O crescimento da empresa levou a que a

mesma fosse restruturada num grupo empresarial [1]. A empresa detentora desse grupo é a EFACEC Power Solutions (EPS), sediada na Arroteia, concelho de Matosinhos. Este grupo possui várias filiais e sucursais, em quatro continentes. No entanto, a maior parte das operações é realizada em Portugal, onde existem oito unidades de negócio divididas por três empresas do grupo [2]:

- EFACEC Electric Mobility, S.A. responsável pela unidade de negócio:
    - Mobilidade Elétrica.
- EFACEC Energia, Máquinas e Equipamentos Eléctricos, S.A. responsável pelas unidades de negócio:
    - Transformadores;
    - Aparelhagem;
    - *Servicing*;
    - Automação.
- EFACEC Engenharia e Sistemas, S.A. responsável pelas unidades de negócio:
    - Energia;
    - Ambiente;
    - Transportes.

Neste capítulo é contextualizada a necessidade, por parte da unidade de Transportes (uTRP), que originou o trabalho apresentado nesta tese. Em seguida são expostos os objetivos principais, de modo a cumprir os requisitos apresentados. Por fim, é apresentada a estrutura do documento.

1.1. CONTEXTUALIZAÇÃO

A uTRP foca-se no desenvolvimento, conceção, comercialização e implementação, de sistemas e produtos para as áreas de Rodovias, Ferrovias e Metros Ligeiros, oferecendo uma variedade de soluções para os setores da Energia, Telecomunicações, Sistemas de controlo Ferroviário e Comando e Controlo [3]. Alguns exemplos dos sistemas desenvolvidos pelos setores supracitados são[4]:

- Sistemas de tração e infraestruturas de distribuição de energia;
- Soluções de catenária para metro ligeiro e ferrovia;
- Sistemas de informação ao público e videovigilância inteligente;
- Sistemas de transmissão digital e comunicações sem fios;

- Sistemas de sinalização ferroviária (AEGIS [5]);
- Sistemas de passagens de nível automáticas (XSafe[6]);
- Plataformas integradas de comando, controlo e operação (EFARAIL e INOSS);
- Controlo Central de Tráfego (AEGIS [5]).

O setor de Sistemas de controlo Ferroviário é responsável por desenvolver SSF. Os SSF são responsáveis por assegurar que os veículos ferroviários se deslocam em segurança, prevenindo todo o tipo de colisões, com veículos da ferrovia ou de outras vias, descarrilamentos, ou outras adversidades que possam ocorrer. É, para isso, necessário que as ferrovias possuam equipamentos de sinalização e encravamento, tais como agulhas, sinais luminosos, passagens de nível, entre outros. De agora em diante, os equipamentos mencionados anteriormente são designados por elementos ferroviários e os dispositivos responsáveis por controlar estes elementos ferroviários são designados por controladores. Qualquer falha inerente a um SSF pode provocar situações catastróficas, que podem originar lesões graves ou até mesmo a morte de uma ou várias pessoas. Desta forma, os SSF acarretam um elevado grau de criticidade, pelo que é fundamental que todas as fases de desenvolvimento de um SSF obedeçam às normas e critérios impostos a este tipo de sistemas.

Sempre que a uTRP desenvolve um SSF é necessário modelar o sistema e integrá-lo nos controladores de acordo com os requisitos impostos na fase de projeto desse sistema. A integração manual destes sistemas num projeto concreto é uma tarefa morosa e propensa a erros por parte do programador, erros estes que se vão repercutir ao longo do processo de integração, o que origina um processo de verificação mais prolongado.

Esta tese surge da necessidade de automatizar o processo de integração dos controladores, independentemente do *design* do projeto. Neste contexto, é necessário abordar os critérios de segurança associados ao desenvolvimento de um SSF. Os equipamentos e *software* utilizados nos SSF são elementos críticos para a segurança, pelo que têm de ser certificados de acordo com a norma internacional IEC 61508, *Functional Safety of Electrical/Electronic/Programmable Eletronic Safety-related Systems* [7]. Mais especificamente, todo o software utilizado no desenvolvimento de um SSF deve estar em conformidade com a norma EN 50128 - *Railway applications: Software for railway control and protection systems*[8].

1.2. OBJETIVOS

O principal objetivo desta tese é desenvolver a ferramenta KCG Adapter To SILWorX (KATOS) capaz de suportar a integração autónoma dos modelos desenvolvidos em *Safety Critical Application Development Environment* (SCADE) Suite num controlador, neste caso, um *Programmable Logic Controller* (PLC) HIMatrix, permitindo assim executar os modelos desenvolvidos em SCADE num PLC programado através da ferramenta SILWorX. A integração autónoma destes modelos vai agilizar o processo de integração e torná-lo mais eficiente. É pretendido que o KATOS seja implementado na ferramenta *Railway Automation Suite* (RAS), uma ferramenta agregadora responsável por vários processos autónomos que será desenvolvida pela EFACEC.

Com o intuito de desenvolver uma solução robusta e fiável para o problema em causa, existe a necessidade de traçar os seguintes objetivos intermédios:

1. Compreender os princípios impostos a um SSF;
2. Compreender os requisitos impostos pelas normas ferroviárias, tais como a norma EN 50128;
3. Compreender a interação entre a fase de projeto e implementação de um SSF;
4. Analisar o software utilizado na fase de projeto;
5. Analisar a melhor solução para desenvolver a ferramenta pretendida;
6. Identificar que informação é necessária para gerar as configurações autonomamente;
7. Estabelecer especificações para a geração das configurações autónomas;
8. Desenvolver uma solução *user-friendly* e intuitiva;
9. Restringir a introdução de informação imprevista, por parte do utilizador;
10. Apresentar toda a informação relativa à geração das configurações, assim como possíveis erros ou avisos;
11. Apresentar um registo associado à geração de todas as configurações realizadas pela ferramenta;
12. Desenvolver um *software* que possa ser integrado na ferramenta RAS.

1.3. ORGANIZAÇÃO DO RELATÓRIO

No capítulo 1, é feita uma introdução ao projeto, contextualizando a necessidade que originou o seu desenvolvimento e apresentando os seus objetivos principais, e a estrutura do documento.

No capítulo 2, é apresentado o estado da arte, sendo abordados conceitos ferroviários acerca do desenvolvimento deste setor, dos princípios e métodos por detrás de um SSF e as normas aplicadas ao seu desenvolvimento, bem como alguns projetos com objetivos similares aos propostos nesta tese.

No capítulo 3, é realizada uma descrição do projeto, é apresentado o *software* utilizado para modelar os SFF e para programar as unidades de controlo utilizadas pela EFACEC, sendo definidos os requisitos específicos do projeto.

Em seguida, no capítulo 4, é descrita a planificação do *software*, sendo apresentado de uma maneira geral a sua arquitetura e, posteriormente, os vários fluxogramas através dos quais é possível compreender a ferramenta desenvolvida.

O capítulo 5, diz respeito à implementação da ferramenta, sendo abordados os *software* utilizados, a interação dos vários processos inerentes à ferramenta e explicados os processos de verificação implementados, bem como algumas adversidades encontradas.

No Capítulo 6, é demonstrado o resultado da ferramenta desenvolvida, sendo apresentadas algumas comparações que permitem demonstrar o funcionamento da ferramenta e são explicadas algumas funcionalidades descritas no capítulo 5.

Por último, no capítulo 7, são apresentadas as conclusões e ainda possíveis melhorias ao projeto.

2. ESTADO DA ARTE

Neste capítulo pretende-se que o leitor adquira conhecimentos sobre conceitos teóricos acerca da evolução do setor ferroviário, de modo a enquadrar a necessidade apresentada pela uTRP. Inicialmente é apresentada uma breve introdução histórica à ferrovia, assim como alguns paradigmas associados ao desenvolvimento de infraestruturas ferroviárias. É realçada a tendência de crescimento deste setor, o que impulsiona a aposta da EFACEC no desenvolvimento de sistemas ferroviários. Posteriormente são abordados os SSF, sendo apresentados métodos de prevenção contra descarrilamentos e colisões. É também explicado como é garantida a segurança dos veículos ferroviários durante a circulação e quais os princípios lógicos que são utilizados para isso. Em seguida, são apresentadas as normas a que deve obedecer o desenvolvimento de sistemas ferroviários de segurança crítica. Por último, são descritos alguns projetos com objetivos similares aos propostos nesta tese, de modo a perceber o que atualmente é feito na área e perspetivar o desenvolvimento de ferramentas que possam auxiliar a modelação de SSF.

2.1. FERROVIA

As infraestruturas ferroviárias e os comboios foram alvo de muitas mudanças ao longo dos anos, a par da evolução tecnológica associada. Desde os sistemas ferroviários mais rudimentares, que possuíam carris de madeira para transportar cargas pesadas através da

força de animais, ou até mesmo dos escravos, tal como era feito, na antiguidade, no transporte de vagões em minas, até aos sistemas ferroviários mais atuais e complexos que acarretam um elevado grau de criticidade. Sem nunca descurar os paradigmas ferroviários que podem contribuir para a descarbonização.

Em 1830 foi inaugurada a *Liverpool and Manchester Railway*. É a primeira ferrovia interurbana, no mundo, idealizada para transportar bens e pessoas através de locomotivas movidas a vapor [9]. Esta ferrovia possuía cerca de 50 km e era possível realizar este itinerário em cerca de uma hora e quarenta e seis minutos, sendo uma mais valia para a comunidade e para as unidades fabris das duas localidades, que até então utilizavam barcos e carroças para viajar e transpor cargas, demorando respetivamente cerca de doze e três horas a completar o trajeto [9]. Naturalmente, com o decorrer dos anos, foram implementadas ferrovias por todo o mundo. Na Figura 1 é possível observar o comprimento da ferrovia em vários países da União Europeia (UE) em 2019, o que torna mais percetível o crescimento das infraestruturas da ferrovia até então.

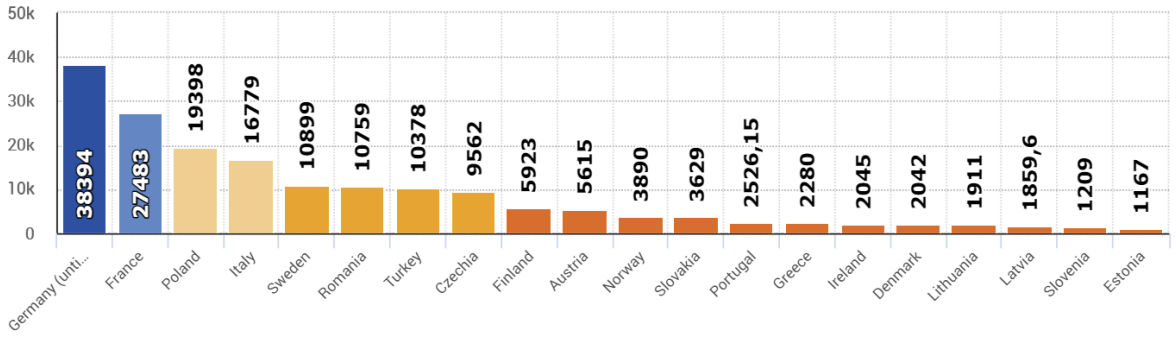


Figura 1 – Comprimento da ferrovia, em km, de diversos países da união europeia em 2019 [10].

Em 1844 Espanha teve a necessidade de se prevenir contra possíveis invasões francesas, pelo que optou por construir a sua rede ferroviária com uma bitola diferente da internacional. A bitola espanhola veio também a ser utilizada na construção da ferrovia portuguesa, atualmente designada por bitola ibérica [11]. Se em 1844 foi necessário construir uma ferrovia com uma bitola diferente da europeia com o intuito de separar/afastar a Espanha da França, atualmente é necessário aproximar não só esses dois países, mas todos os outros. Essa aproximação tem o intuito de referir-se aos benefícios económicos que podem ser gerados, mas também ao encurtamento dos tempos de viagem. Existe um grande caminho a percorrer para implementar uma única rede europeia de transporte ferroviário. Atualmente, estão implementados modelos operacionais distintos em vários países. Na Figura 2 estão

expostos os vários modelos. Na França, Espanha e Itália é utilizado o modelo de alta velocidade mista, enquanto que na Alemanha, Áustria e em duas ferrovias Italianas é utilizado o modelo totalmente misto [12]. As disparidades técnicas entre as redes ferroviárias dos vários países, como, por exemplo, a incompatibilidade das normas aplicáveis ao tráfego ferroviário, têm de ser ultrapassadas. Contudo, isso pode implicar restruturar as estruturas e os sistemas em vigor.

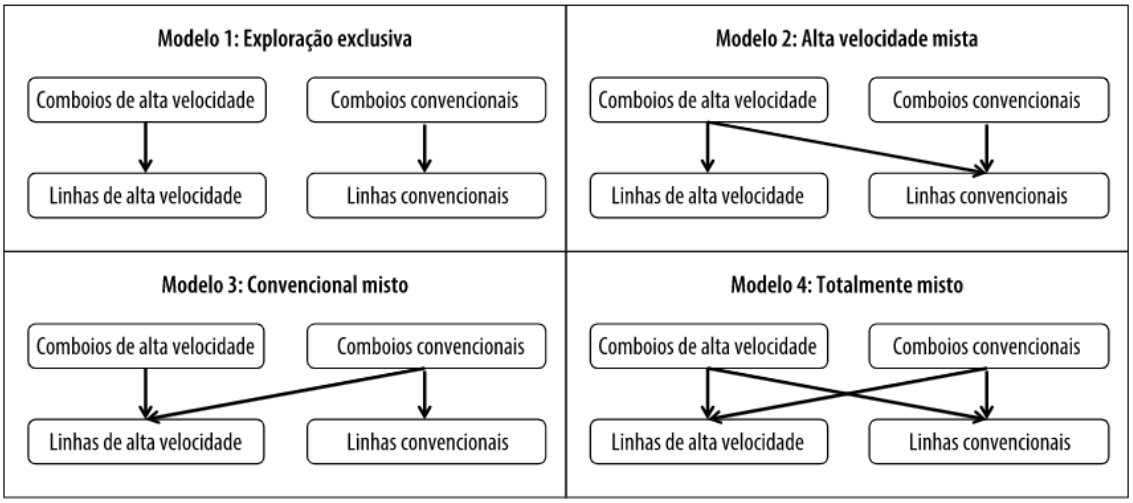


Figura 2 – Modelos operacionais para o tráfego ferroviário de alta velocidade [12]

O relatório elaborado pelo Tribunal de Contas Europeu indica que "O transporte ferroviário de alta velocidade é um modo de transporte confortável, seguro, flexível e ambientalmente sustentável. Proporciona desempenho ambiental e benefícios socioeconómicos que podem apoiar os objetivos da política de coesão e de transportes da UE. Desde 2000, a UE disponibilizou 23,7 mil milhões de euros em cofinanciamento para apoiar os investimentos em infraestruturas ferroviárias de alta velocidade" [12]. A implementação de linhas de alta velocidade gera grande controvérsia relativamente à rentabilidade do investimento, desde o aumento dos encargos relacionados com a manutenção da ferrovia à implantação de sistemas ferroviários atuais.

Relativamente às grandes cidades nas quais o tráfego automóvel é muito intenso, existe a necessidade de reduzir as emissões de carbono. Optar por desenvolver novas linhas de metro e expandir as existentes, de modo a responder às necessidades da comunidade, é uma possível solução para a redução deste tipo de emissões. Em *The Effects of Subway Expansion on Traffic Conditions: Evidence from Beijing* perspetivou-se a construção de 279 estações e cerca de 561km de ferrovia até 2015, de maneira a dar resposta à densidade populacional de

Pequim [13]. Noutro estudo é apresentada uma estimativa do impacto desta expansão. Numa perspetiva temporal de 10 anos, serão obtidos benefícios a nível da saúde pública e da diminuição do congestionamento, benefícios que podem representar um retorno de 3,57 % e 95,3 %, respetivamente, do investimento inicial de cerca de 46 mil milhões de euros [14].

Segundo a previsão *World Rail Market Study: Forecast 2020 to 2025*, realizada pela empresa de consultadoria Roland Berger a pedido da associação *The European Rail Industry* (UNIFE), o mercado ferroviário mundial sofreu uma descida de 8% em 2020, devido à crise económica mundial gerada pela pandemia do COVID-19. Contudo, está previsto um crescimento estável até 2025, seguindo a tendência apresentada por este sector que cresce, anualmente, 3,6% desde 2017 [15].

O decréscimo de acidentes na ferrovia ao longo dos anos, Figura 3, estará certamente relacionado com a reabilitação da ferrovia e com a utilização de sistemas de controlo inovadores. É, portanto, crucial continuar a apostar no desenvolvimento de soluções para o setor ferroviário, tais como os SSF.

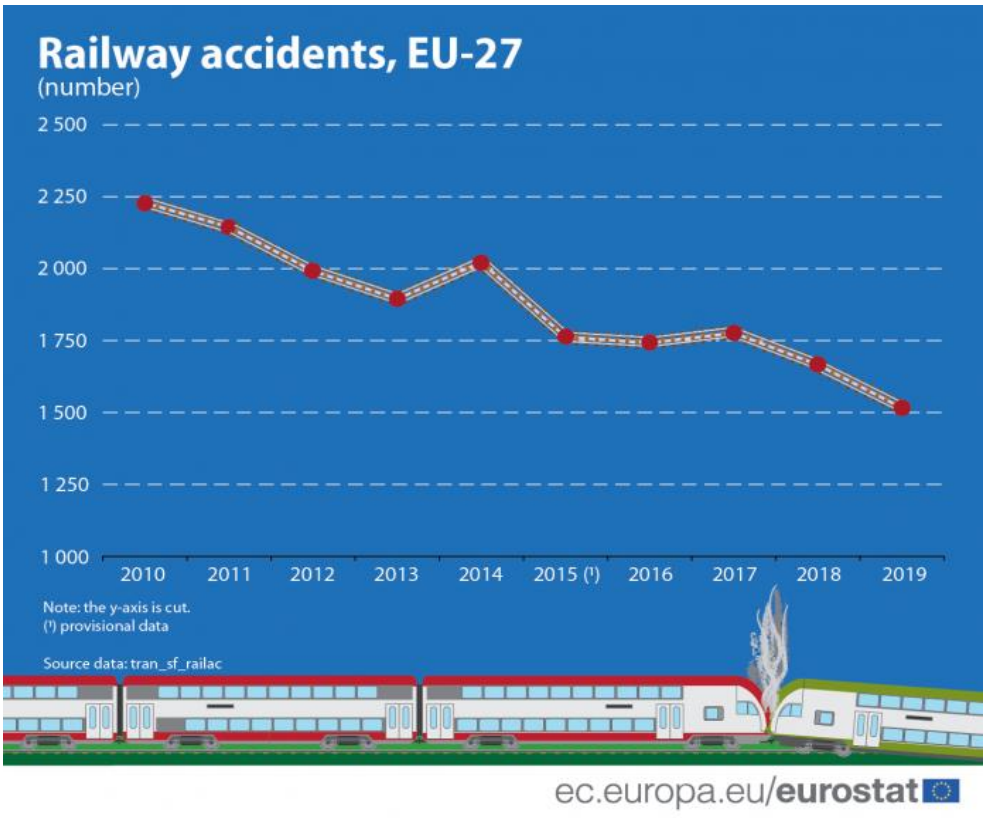


Figura 3 – Número de acidentes ferroviários na União Europeia [16].

2.2. SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA

Um SSF é responsável por gerir o tráfego de uma linha ferroviária e garantir que todos os veículos ferroviários se deslocam em segurança. Para isso, é necessário que o sistema possua toda a informação acerca da ocupação da ferrovia. Através do processamento desta informação é possível sinalizar as secções de via, impedindo ou permitindo a entrada de veículos ferroviários nessa secção [17]. As secções de via são troços ferroviários delimitados por sinais luminosos. A sua alocação a um determinado itinerário permite garantir que estes não são ocupados por mais do que um veículo ferroviário ao mesmo tempo. Existe uma relação entre os sinais luminosos e todos os elementos ferroviários que fazem parte de um itinerário, como as agulhas, uma vez que, por exemplo, o aspeto de autorização de movimento apenas é mostrado se todos os elementos ferroviários estiverem em condições de permitir a entrada de um veículo num dado itinerário. É percetível que um SSF é responsável pelo encravamento de todos os elementos ferroviários e através deste é possível estabelecer itinerários para os veículos ferroviários de forma segura.

O único método de prevenção contra colisões de veículos ferroviários, utilizado nas primeiras ferrovias, era garantir que os veículos chegam ao seu destino no intervalo de tempo estabelecido [17]. Quando isso não acontecia, assumia-se que teria ocorrido alguma avaria, embora não existisse nenhuma informação sobre o que realmente tinha ocorrido. Contudo, com o aumento da velocidade dos veículos ferroviários e do tráfego ferroviário surgiu a necessidade de implementar e regulamentar os sistemas de segurança ferroviária, pelo que foram equacionados novos métodos de prevenção contra acidentes. As primeiras alterações originaram o que atualmente designamos de cabines sinalizadoras, Figura 4. Através deste sistema era possível controlar sinais, ao invés de bandeiras, e agulhas de uma determinada área, recorrendo a alavancas, sem existir a necessidade dos trabalhadores ferroviários se deslocarem até ao local [18]. Foi também implementada a comunicação entre cabines sinalizadoras adjacentes, através de telégrafos, permitindo dividir a via em secções, prevenindo que não existisse mais do que um veículo ferroviário na mesma secção de via ao mesmo tempo [17]. Todavia, devido à complexidade das tarefas encarregadas aos trabalhadores da ferrovia, propicias ao erro humano, continuavam a ocorrer muitos acidentes na ferrovia.

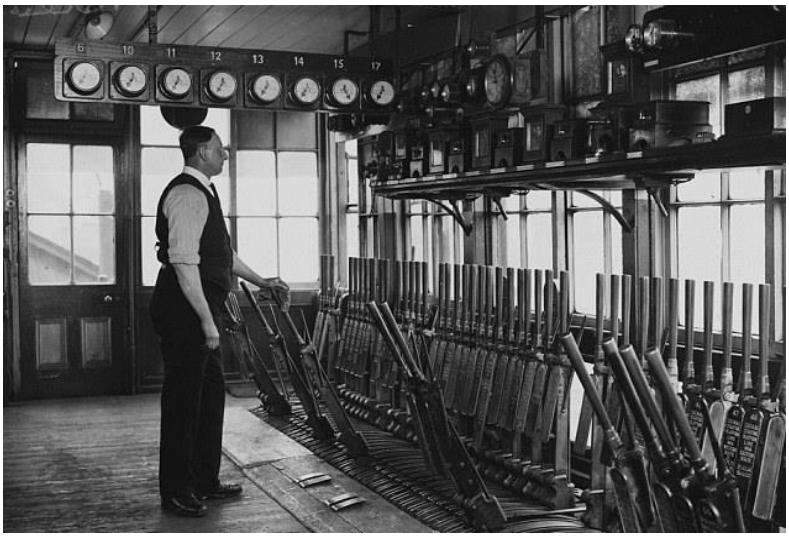


Figura 4 – Cabine sinalizadora, construída em 1881 na estação *King's Cross* [19].

Embora estes princípios não tenham sofrido alterações significativas ao longo dos anos, os sistemas que controlam os elementos ferroviários sofreram. Desde os sistemas de encravamento mecânico, até aos eletromecânicos, baseados em relés e eletrónicos/*Solid State Interlocking* (SSI), estes últimos permitindo que o controlo do encravamento seja efetuado ao nível lógico através de unidades de controlo que possuem processadores, como os PLC, a evolução tecnológica permitiu substituir as cabines sinalizadoras pelos atuais armários de controlo, Figura 5 [17].

Figura 5 – Armário de controlo implementado na solução XSafe, desenvolvido pela EFACEC [20].

2.2.1. REQUISITOS DE UM SISTEMA DE SINALIZAÇÃO FERROVIÁRIA

As características da ferrovia impõem o risco de colisão e descarrilamento durante a deslocação dos veículos ferroviários, pelo que existe a necessidade de implementar vários métodos de prevenção, Figura 6. Em caso de travagem os veículos ferroviários necessitam de uma grande distância para parar completamente, devido ao reduzido atrito entre os carris e as rodas dos veículos ferroviários, o que pode originar colisões. Como a posição das agulhas é que define o itinerário que o veículo irá tomar o maquinista não tem a possibilidade de se desviar de obstáculos nem de optar por outros percursos, em caso de alguma anomalia ou erro humano, o que pode originar descarrilamentos [17].

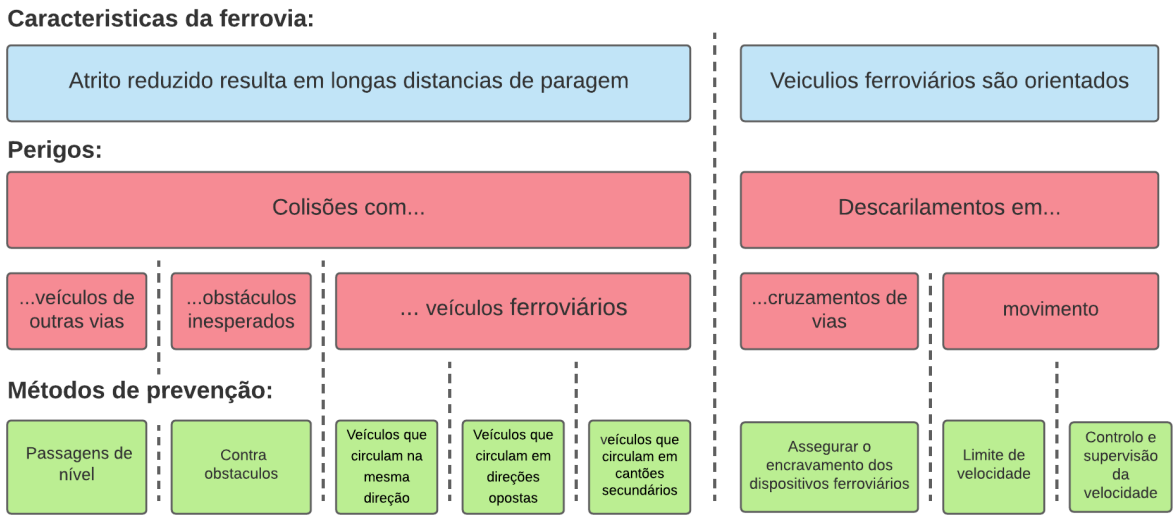


Figura 6 – Exemplos de acidentes ferroviários e métodos de prevenção [17].

Os métodos de prevenção contra colisões e descarrilamentos são requisitos básicos de qualquer ferrovia, sendo eles [17]:

- Proteção contra veículos que circulam na mesma direção;
- Proteção contra veículos que circulam em direções opostas;
- Proteção contra veículos que circulam em secções de via secundários/paralelas;
- Proteção nas passagens de nível;
- Proteção contra objetos externos à ferrovia;
- Garantir o bom funcionamento dos elementos ferroviários que permitem estabelecer rotas;
- Definir limites de velocidade;
- Controlar e supervisionar a velocidade dos veículos.

2.2.2. MÉTODOS DE SALVAGUARDAR OS PERCURSOS DOS VEÍCULOS FERROVIÁRIOS

Existem dois métodos que permitem salvaguardar os percursos dos veículos ferroviários. Estes métodos são designados por itinerários/rotas e blocos orientados. Embora estes métodos possuam características distintas, como iremos verificar de seguida, por vezes é possível tirar partido da implementação de ambos na mesma ferrovia. Contudo ambos devem obedecer às seguintes regras [17]:

- Um sinal só pode permitir a entrada de um veículo ferroviário, num determinado itinerário, se todos os elementos ferroviários estiverem encravados na posição correta e devem permanecer encravados enquanto estiverem a ser utilizados pelo veículo;
- Exceto em situações especiais, não pode existir mais do que um veículo ferroviário a realizar a mesma secção do bloco.

No método blocos orientados, Figura 7, assim que um veículo desocupa uma determinada secção de via é enviada uma mensagem, com essa informação, para o sistema que controla essa mesma secção. A receção desta mensagem permite autorizar a entrada de veículos à secção de via em causa. No entanto, através deste método, só é possível prevenir colisões contra veículos que circulam na mesma direção ou em direções opostas, pelo que só é utilizado em secções de via sem interceções/plena via [17].

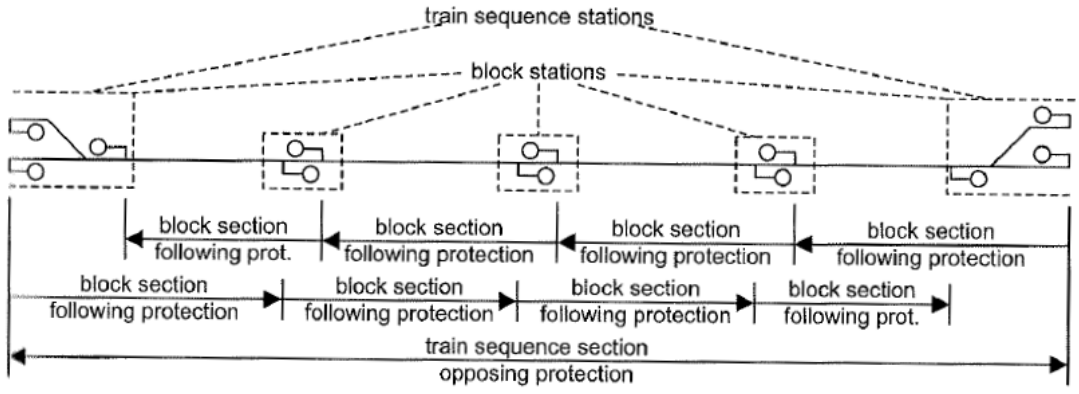


Figura 7 – Sequência de blocos orientados entre duas estações [17].

Para originar a criação de um itinerário, Figura 8, deve ser feito um pedido para a deslocação entre a posição A, inicial, e a posição B, final. De seguida, é verificado se todas as secções de via necessárias a essa deslocação se encontram desocupados. Se as secções de via estiverem desocupadas, a deslocação será permitida, embora seja necessário que todos os elementos ferroviários sejam encravados na posição correta. Assim que o encravamento for

realizado, os sinais luminosos indicam que o itinerário foi estabelecido. As secções de via utilizadas para completar este itinerário podem ser libertadas à medida que o veículo ferroviário as percorre, ainda que este não tenha completado o itinerário. Enquanto um itinerário estiver estabelecido, estão implementados mecanismos que aplicam os métodos de prevenção contra colisões e descarrilamentos, apresentados em 2.2.1 [17]. No tópico 2.2.3 são abordados dois principios que permitem definir as dependências entre os elementos ferroviários de forma a estabelecer um itinerário.

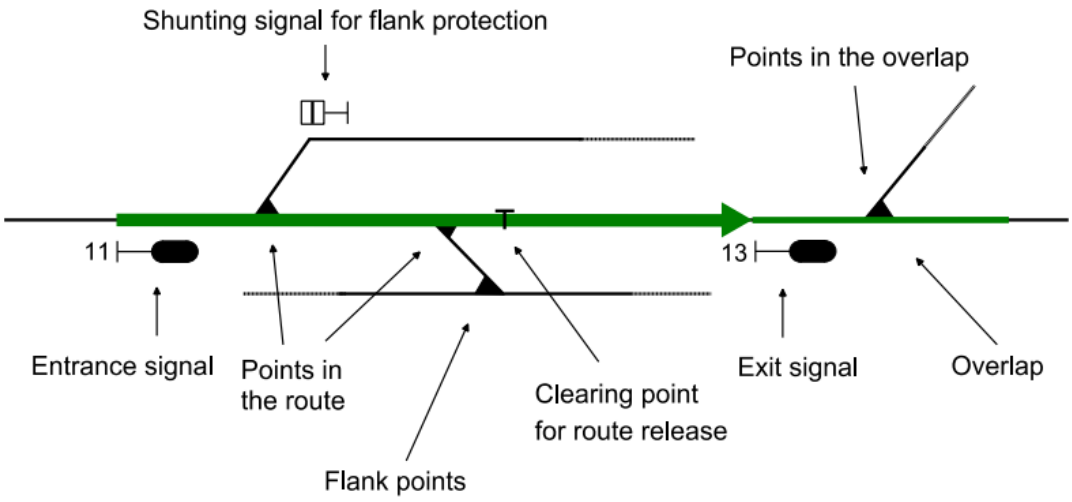


Figura 8 – Exemplo de um itinerário e os seus constituintes [21].

2.2.3. PRINCÍPIOS LÓGICOS UTILIZADOS NO ENCRAVAMENTO

Como foi mencionado anteriormente, um veículo ferroviário só pode entrar numa determinada secção de via se todos os elementos ferroviários estiverem encravados. Atualmente, nos sistemas ferroviários é aplicado o princípio centralizado ou o geográfico [17]. Existe também o princípio cascata que não será abordado neste documento. Através destes princípios, é possível estabelecer quais são os elementos ferroviários a encravar de forma a estabelecer um itinerário.

O princípio centralizado é utilizado desde os sistemas de encravamento mecânicos e continua a ser utilizado no encravamento eletrónico. Neste princípio, as condições de encravamento são definidas numa tabela de controlo. Nesta tabela estão expostos todos os itinerários possíveis para a ferrovia em questão. Para cada itinerário estão indicados os elementos ferroviários que o compõem, assim como as suas posições de encravamento. Esta

tabela possui também informação acerca de quais os itinerários interditos quando um determinado itinerário está a ser utilizado [17], [21]. Na Figura 9 é exposto um troço de uma infraestrutura ferroviária e na Tabela 1 a tabela de controlo associada a esse troço.

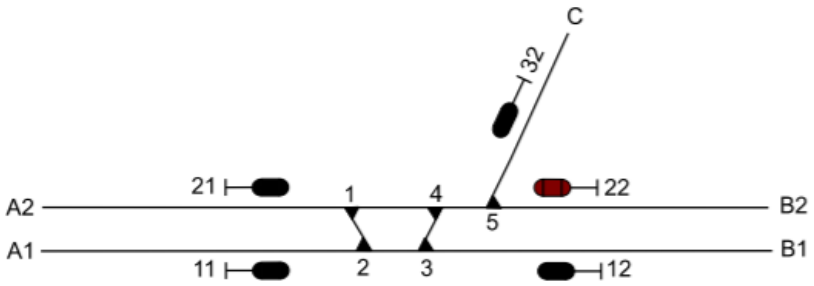


Figura 9 – Troço de uma infraestrutura ferroviária [21].

Tabela 1 – Tabela de controlo de itinerários para o troço exposto na Figura 9 [21].

ROUTE	ROUTES LOCKED OUT	POINTS	
		NORMAL	REVERSE
11-B1	11-B2, 11-C, 21-B1, 12-A1, 12-A2, 22-A1, 32-A1	1, 2, 3, 4	
11-B2	11-B1, 11-C, 21-B1, 21-B2, 21-C, 12-A1, 12-A2, 22-A1, 22-A2, 32-A1, 32-A2	1, 2, 5	3, 4
11-C	11-B1, 11-B2, 21-B1, 21-B2, 21-C, 12-A1, 12-A2, 22-A1, 22-A2, 32-A1, 32-A2	1, 2	3, 4, 5
21-B1	11-B1, 11-B2, 11-C, 21-B2, 21-C, 12-A1, 12-A2, 22-A1, 22-A2, 32-A1, 32-A2	3, 4	1, 2
21-B2	11-B2, 11-C, 21-B1, 21-C, 12-A2, 22-A1, 22-A2, 32-A1, 32-A2	1, 2, 3, 4, 5	
21-C	11-B2, 11-C, 21-B1, 21-B2, 12-A2, 22-A1, 22-A2, 32-A1, 32-A2	1, 2, 3, 4	5
12-A1	11-A1, 11-B2, 11-C, 21-B1, 12-A2, 22-A1, 32-A1	1, 2, 3, 4	
12-A2	11-B1, 11-B2, 11-C, 21-B1, 21-B2, 21-C, 12-A1, 22-A1, 22-A2, 32-A1, 32-A2	3, 4	1, 2
22-A1	11-B1, 11-B2, 11-C, 21-B1, 21-B2, 21-C, 12-A1, 12-A2, 22-A2, 32-A1, 32-A2	1, 2, 5	3, 4
22-A2	11-B2, 11-C, 21-B1, 21-B2, 21-C, 12-A2, 22-A1, 32-A1, 32-A2	1, 2, 3, 4, 5	
32-A1	11-B1, 11-B2, 11-C, 21-B1, 21-B2, 21-C, 12-A1, 12-A2, 22-A1, 22-A2, 32-A2	1, 2	3, 4, 5
32-A2	11-B2, 11-C, 21-B1, 21-B2, 21-C, 12-A2, 22-A1, 22-A2, 32-A1	1, 2, 3, 4	5

O princípio geográfico foi desenvolvido para os sistemas de encravamento baseados em relés e continua a ser utilizado no encravamento eletrónico. Neste princípio, todos os elementos ferroviários possuem dependências com todos os elementos adjacentes. Cada itinerário possui elementos ferroviários que delimitam a entrada e a saída desse itinerário.

Quando se pretende estabelecer um itinerário sabe-se qual é o seu início e fim. A partir dai são estabelecidas as dependências entre os vários elementos ferroviários adjacentes, se estes não estiverem a ser utilizados. Quando são estabelecidas as dependências dos elementos ferroviários, todas as rotas que possam gerar conflitos são bloqueadas e os sistemas de proteção que possam ser ativados são também assegurados [17], [21]. Na Figura 10 é exposto um troço de uma infraestrutura ferroviária e a correspondente disposição dos blocos lógicos de cada elemento ferroviário, através destes são estabelecidas as dependências com os elementos adjacentes.

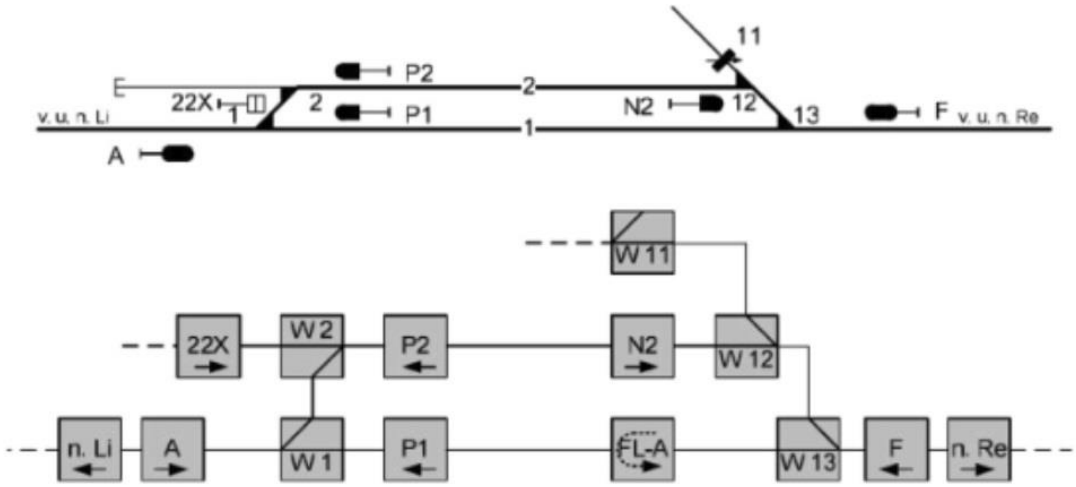


Figura 10 – Princípio geográfico [22].

A utilização destes dois princípios acarreta diferentes vantagens e desvantagens consoante a utilização do encravamento baseado em relés e o encravamento eletrónico. Relativamente ao encravamento baseado em relés, na Figura 11 encontra-se exposto um gráfico através do qual é percetível que quanto mais complexo for o SSF, mais vantajoso é implementar o princípio geográfico.

A implementação do princípio geográfico através de sistemas baseados em relés implica que para cada configuração de um conjunto de elementos ferroviários, exista um grupo de relés que irá controlar o relé associado a cada elemento. Estas configurações podem ser testadas e validadas antes da implementação da infraestrutura, todavia o processo de validação deste princípio é mais complexo que o processo de validação do princípio centralizado. No princípio centralizado, todos os elementos são controlados individualmente, o que é uma mais valia em infraestruturas simples. No entanto, quanto mais complexa for a infraestrutura e, subsequentemente, maior o número de elementos ferroviários, mais adequada será a

implementação do princípio geográfico. Se existir a necessidade de efetuar alterações à infraestrutura, incorporando novos elementos ferroviários, a utilização do princípio geográfico também é benéfica.

Nos sistemas de encravamento eletrónicos, o encravamento de cada elemento é individual e é realizado ao nível lógico, pelo que as dificuldades técnicas para a implementação de ambos os princípios são similares. Ambos continuam a ser utilizados consoante a infraestrutura em questão [17]. Para além de ser essencial utilizar o princípio lógico mais adequado para o SSF que se pretende desenvolver, é essencial cumprir todas as normas impostas ao desenvolvimento destes sistemas, como se descreve no tópico 2.3.

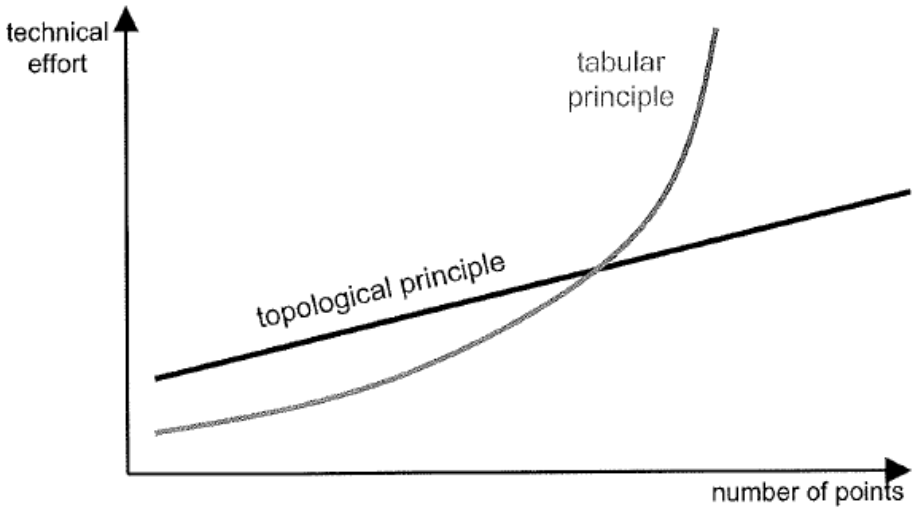


Figura 11 – Comparação entre o esforço técnico na implementação do princípio centralizado e geográfico, em sistemas de encravamento baseados em relés [17].

2.3. SEGURANÇA NOS SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA

A norma internacional IEC 61508 impõem métodos sobre como desenvolver e implementar equipamentos e sistemas eletrónicos, de segurança crítica, que possam causar situações de catástrofe na indústria em geral. O principal objetivo desta norma é reduzir a probabilidade de ocorrência de situações anómalas, e, em caso de ocorrência, minimizar o seu impacto. Esta norma estabeleceu 4 SIL, Tabela 2, atribuídos de acordo com a probabilidade de ocorrência de situações perigosas. No setor ferroviário é necessária uma monitorização constante dos sistemas afetos, pelo que o cálculo do SIL é efetuado tendo em conta a probabilidade de ocorrência de uma falha por hora, coluna mais à direita na Tabela 2.

Tabela 2 - Níveis SIL [23].

SIL Safety Integrity Level	**PFDavg** Average probability of failure on demand per year (low demand mode)	**RRF** Risk Reduction Factor	**PFDavg** Average probability of failure on demand per hour (high demand or continuous mode)
SIL 4	$\geq 10^{-5}$ and $< 10^{-4}$	100000 to 10000	$\geq 10^{-9}$ and $< 10^{-8}$
SIL 3	$\geq 10^{-4}$ and $< 10^{-3}$	10000 to 1000	$\geq 10^{-8}$ and $< 10^{-7}$
SIL 2	$\geq 10^{-3}$ and $< 10^{-2}$	1000 to 100	$\geq 10^{-7}$ and $< 10^{-6}$
SIL 1	$\geq 10^{-2}$ and $< 10^{-1}$	100 to 10	$\geq 10^{-6}$ and $< 10^{-5}$

Quanto maior for a previsível gravidade de um acidente causado por uma determinada anomalia, maior deve ser o SIL a usar, de forma a reduzir ao mínimo a possibilidade de ocorrência dessa anomalia. Determinar e calcular o nível SIL associado a cada sistema não será abordado neste documento, mas é possível encontrar informação a esse respeito em [23], [24].

A partir da norma internacional IEC 61508 foram estabelecidas normas mais específicas para as várias indústrias. No setor ferroviário europeu são aplicadas as seguintes normas [25]:

- EN 50126 - *Railway applications: The specification and demonstration of Reliability, Availability, Maintainability and Safety* (RAMS) [26];
- EN 50128 - *Railway applications: Software for railway control and protection systems* [8];
- EN 50129 - *Railway applications: Safety related electronic systems for signalling* [27];
- EN 50155 - *Railway applications: Electronic equipment used on rolling stock* [28];
- EN 50159 - *Railway applications: Safety-related communication in transmission systems* [29].

A norma EN 50126 aborda conceitos de base que especificam como é que se deve comportar uma determinada solução ferroviária, durante o seu normal funcionamento, através da RAMS dessa solução. A EN 50128 diz respeito aos métodos de desenvolvimento e validação de *software* de um SFF [30]. Em EN 50129 são estabelecidas normas com o intuito de

garantir a segurança do *hardware* utilizado nos SSF. A EN 50155 diz respeito a todos os sistemas eletrónicos, *hardware*, *software* afetos a um veículo ferroviário, pelo que não é tida em conta no desenvolvimento de um SSF. Por último, com a norma EN 50159 pretende-se garantir a segurança em todos os tipos de comunicação entre os equipamentos que controlam os sistemas de segurança [31]. Através da Figura 12 é possível ter uma ideia geral da aplicação das normas nas etapas do desenvolvimento de um sistema ferroviário. A norma EN 50128 será abordada com mais detalhe, pois é crucial ao desenvolvimento de software para SSF.

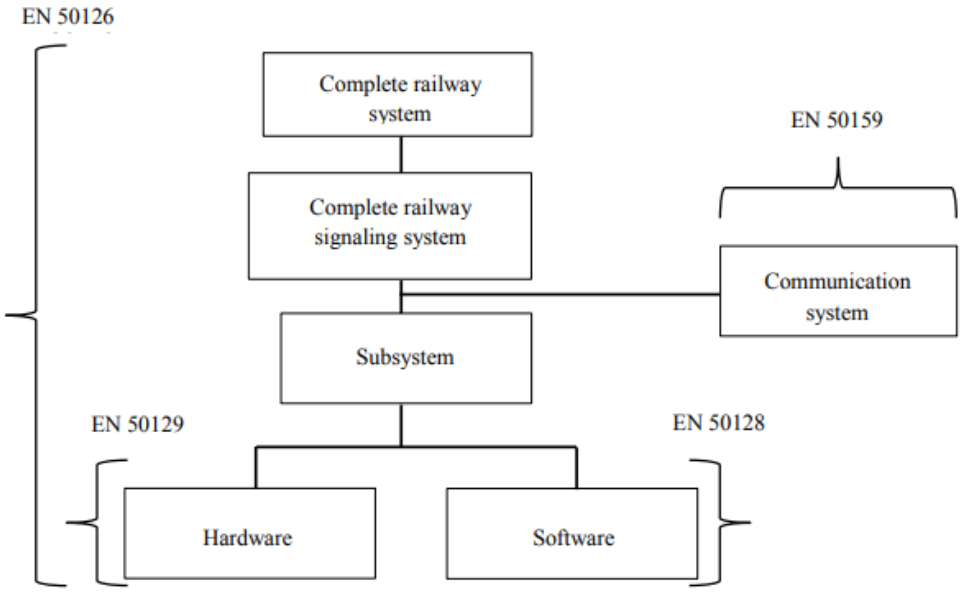


Figura 12 – Principais normas aplicadas no desenvolvimento de sistemas ferroviárias [31].

2.3.1. NORMA EN 50128

Como já foi referido anteriormente, a norma europeia EN 50128 específica o processo e os requisitos técnicos a aplicar no desenvolvimento de software para sistemas de controlo e proteção ferroviários. Estes sistemas podem ser desenvolvidos para microprocessadores, PLC, sistemas de controlo centralizados, entre outros. Esta norma não se aplica a *software* que não tem impacto em garantir a segurança do sistema, mais concretamente a *software* que em caso de falha não afete qualquer função de segurança [8]. Nesta norma é também aplicado o conceito SIL, introduzido na norma IEC 61508. Consoante o SIL são estabelecidos diferentes requisitos ao desenvolvimento do software.

A norma EN 50128 estabelece que deve existir uma estrutura organizacional na qual cada elemento, responsável pelo projeto do software, tem um papel definido de acordo com o SIL, Figura 13 [8].

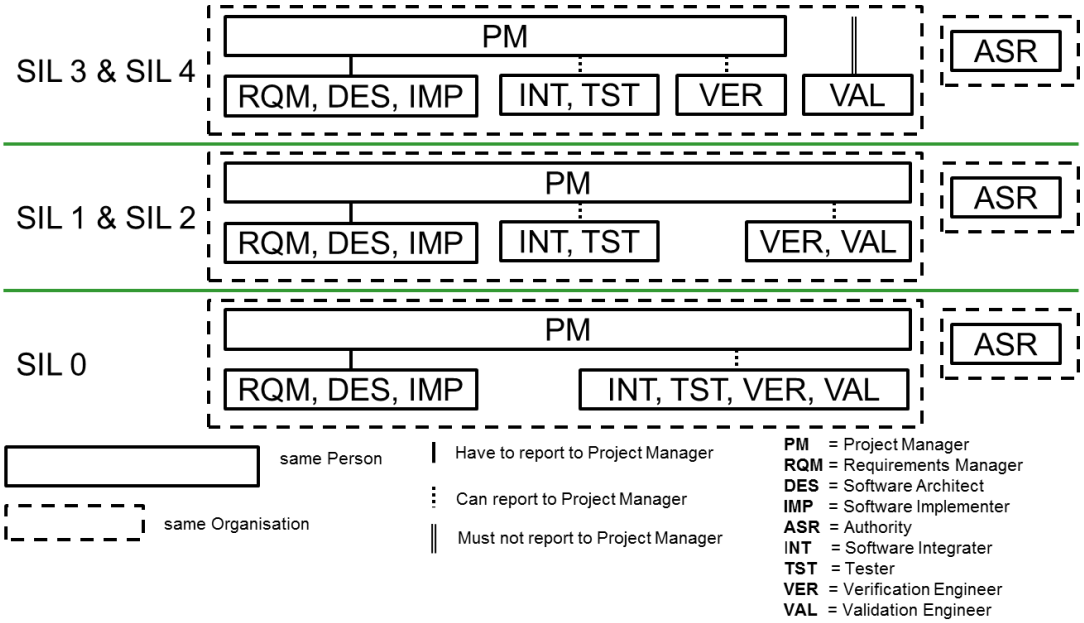


Figura 13 – Estruturas organizacionais para o desenvolvimento de *software*, de acordo com a norma EN 50128 [8].

O projeto do *software* deve ser estruturado em etapas definidas e deve estar de acordo com Plano de Garantia de Qualidade de *software,* documentado com mais detalhe nesta norma [8]. A norma apresenta dois modelos com tarefas demarcadas para cada etapa, sendo eles o modelo em cascata e em V, representados respetivamente na Figura 14 e Figura 15.

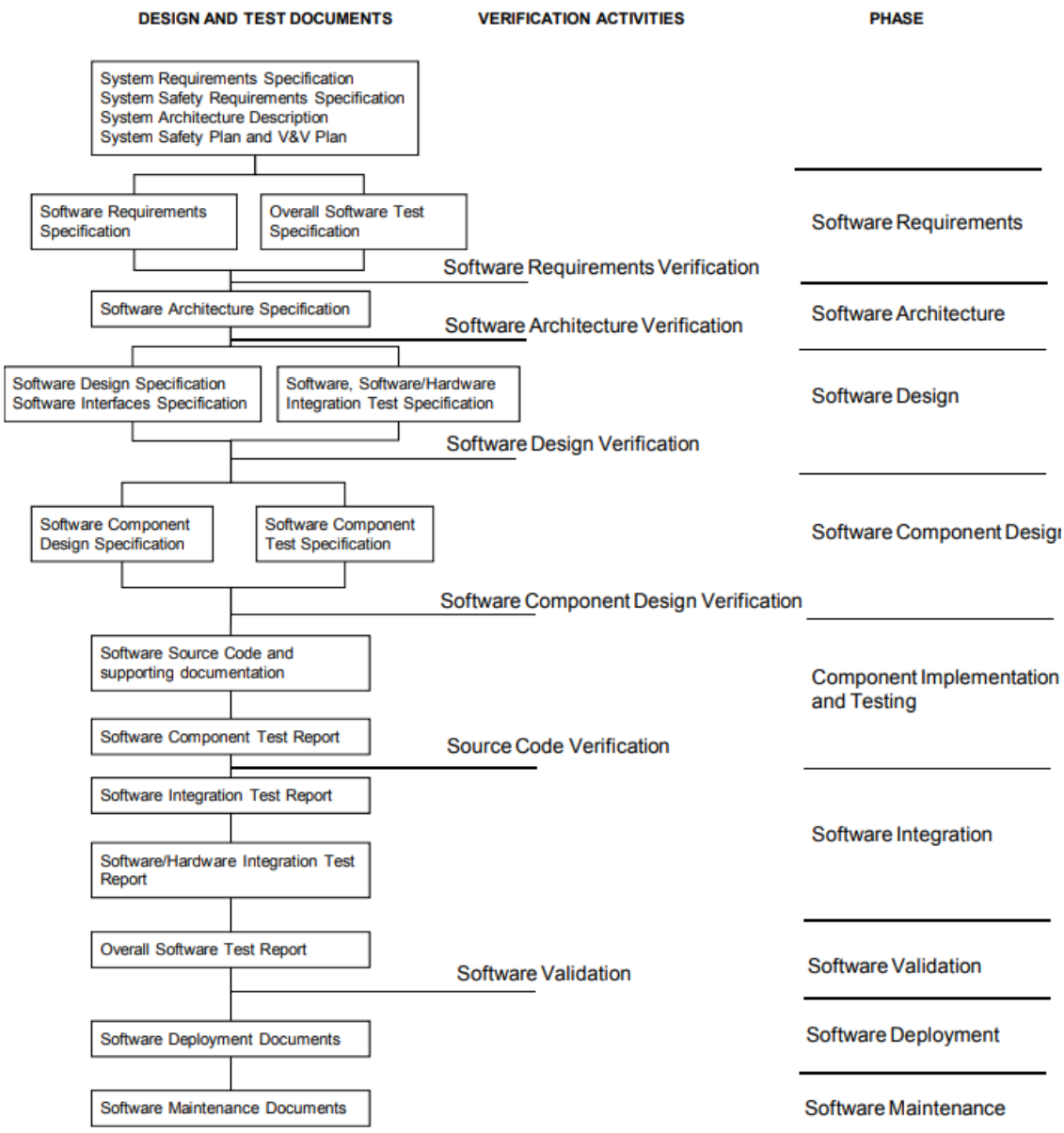


Figura 14 – Modelo de projeto cascata [8].

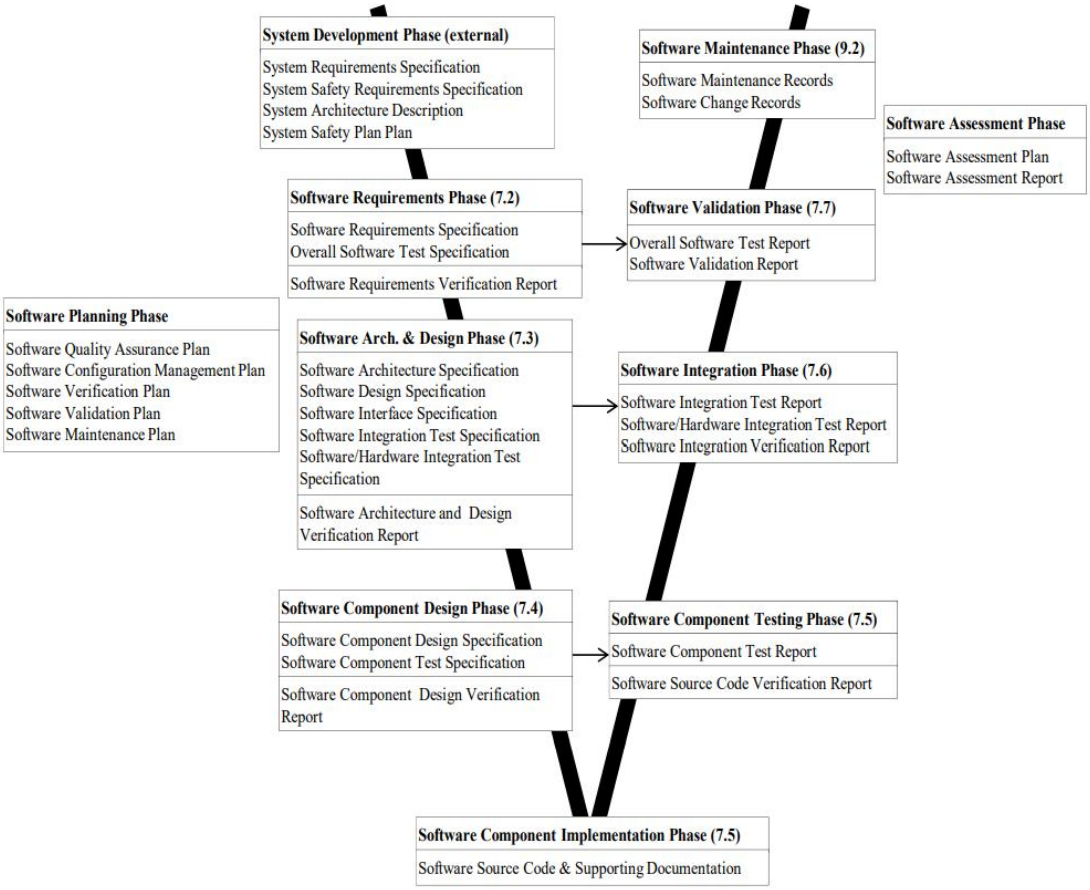


Figura 15 – Modelo de projeto em V [8].

A norma define, também, regras de implementação para cada fase do projeto do software, de acordo com o nível SIL, e que podem ser classificadas como obrigatórias (M), fortemente recomendadas (HR), recomendadas (R) e não obrigatórias (NR). Um possível exemplo são os requisitos do software apresentados na Tabela 3, entre outras especificações sobre as regras de implementação, compiladores, linguagens de programação, entre outros [8].

Tabela 3 – Requisitos do *software* [8].

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Formal Methods (based on a mathematical approach)	D.28	-	R	R	HR	HR
2. Modelling	Table A.17	R	R	R	HR	HR
3. Structured methodology	D.52	R	R	R	HR	HR
4. Decision Tables	D.13	R	R	R	HR	HR
Requirements: 1) The Software Requirements Specification shall include a description of the problem in natural language and any necessary formal or semiformal notation. 2) The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used.						

As ferramentas utilizadas para desenvolver sistemas críticos influenciam diretamente o seu funcionamento, pelo que um erro da ferramenta pode influenciar diretamente o comportamento do sistema crítico. A norma EN 50128 classifica as ferramentas da seguinte forma [8]:

- Ferramenta classe T1: Não tem qualquer influência no resultado do código executável e nos dados do sistema crítico. Como exemplo, uma ferramenta da classe T1 inclui um editor de texto ou um requisito ou ferramenta de apoio à conceção sem capacidades de geração automática de código;
- Ferramenta classe T2: Este tipo de ferramenta é utilizado para fins de teste e verificação do design ou código executável, mas não cria diretamente erros no executável responsável pelo sistema crítico. Como exemplo, uma ferramenta da classe T2 pode ser uma ferramenta de análise estática ou um gerador de testes;
- Ferramenta classe T3: Gera resultados que podem afetar direta ou indiretamente o código executável incluindo os dados relacionados com a segurança. Uma ferramenta da classe T3 pode incluir um compilador de código fonte, um compilador de dados/algoritmos, um compilador de otimização, entre outros.

Quando um processo manual é substituído pela utilização de uma ferramenta, a prova da integridade dessa ferramenta passa pelas mesmas etapas de validação como se de um processo manual se tratasse. As etapas de validação da integridade da ferramenta podem ser substituídas se for apresentada argumentação sobre a integridade dos resultados da ferramenta e o nível de integridade do resultado não for diminuído pela substituição.

2.4. TRABALHOS REALIZADOS NA ÁREA

Ainda que a norma EN 50128 apresente recomendações acerca das metodologias utilizadas na conceção de *software,* não impõem a obrigatoriedade de utilização de metodologias ou ferramentas específicas. As ferramentas e as metodologias são escolhidas de acordo com a preferência dos desenvolvedores do *software,* desde que cumpram os SIL. Contudo, cada ferramenta terá prós e contras que, consequentemente, irão facilitar ou dificultar o trabalho dos desenvolvedores. Para colmatar estas dificuldades são desenvolvidas ferramentas com o intuito de auxiliar o processo de desenvolvimento e implementação do *software,* tal como é proposto nesta tese. De seguida são apresentados dois projetos que possuem características que se enquadram no que foi descrito anteriormente.

No estudo *Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra* é apresentada uma ferramenta que foi desenvolvida com o intuito de automatizar a geração de tabelas de controlo, segundo o princípio centralizado [32]. Esta ferramenta, foi desenvolvida através da *Framework .NET* da *Microsoft.* Permite que o utilizador esquematize a disposição dos carris, de acordo com o *layout* desejado. A ferramenta é responsável por introduzir todos os elementos ferroviários necessários para o encravamento dos itinerários criados. Depois de modelar a estrutura ferroviária, as tabelas de controlo dessa ferrovia, são geradas automaticamente pela ferramenta. Esta ferramenta permite poupar recursos humanos e tempo no desenvolvimento de um SSF.

No estudo *Railway Interlocking Design Support Tools: AutoCAD Plugin based approach* é apresentado o desenvolvimento de um módulo que é integrado na ferramenta RailCOMPLETE [18]. A ferramenta RailCOMPLETE é um plug-in para AutoCAD, idealizada para o desenvolvimento de sistemas ferroviários [33]. O objetivo do módulo é desenvolver uma ferramenta capaz de criar e editar as especificações para o sistema de encravamento, segundo as especificações do *railway Markup Language* (railML) [34]. O railML, desenvolvido através da linguagem *eXtensible Markup Language* (XML), tem como objetivo simplificar a transferência de dados entre vários sistemas ferroviários através da especificação de uma estrutura de dados comum. Portanto, o railML é uma mais valia para os fabricantes de sistemas ferroviários, porque agiliza a transição de informação entre os vários sistemas desenvolvidos por um qualquer programa, Figura 16 [27].

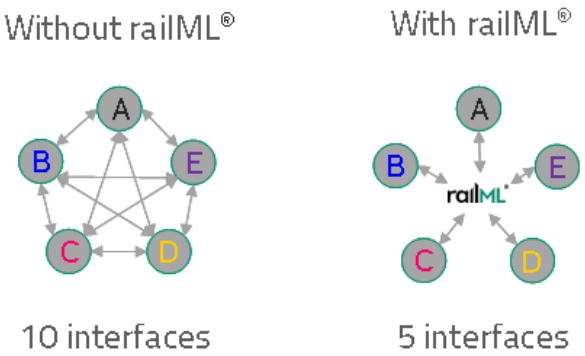


Figura 16 – Troca de informação recorrendo ao railML[34].

Ao longo deste capítulo foram abordados vários conceitos sobre a ferrovia e o desenvolvimento de sistemas ferroviários, de forma a elucidar o leitor sobre conceitos que estão por de detrás da necessidade de desenvolver o projeto apresentado nesta tese. No próximo capítulo é abordada a ferramenta que se pretende desenvolver.

3. DESCRIÇÃO DO PROJETO

O encravamento eletrónico impulsionou o desenvolvimento de sistemas ferroviários. Os programas utilizados no desenvolvimento destes sistemas variam consoante os fabricantes, porém devem cumprir todas as normas aplicadas ao desenvolvimento destes sistemas.

Ao longo deste capítulo são abordados os programas utilizados, pela uTRP, para desenvolver os SSF. Pretende-se, também, enquadrar o leitor com a necessidade de desenvolver o projeto proposto nesta tese. Como foi referido anteriormente, é necessário desenvolver uma ferramenta capaz de integrar os modelos dos SSF autonomamente nos controladores, agilizando a integração do sistema modelado, o que torna o processo mais eficiente. Inicialmente, é apresentado o programa SCADE Suite utilizado na fase de modelação. De seguida, é apresentado o programa SILworX utilizado na fase de implementação. Por último, é explicada a ferramenta que se pretende desenvolver e qual o seu enquadramento com os dois programas, de forma a evidenciar a sua utilidade e apresentar alguns objetivos mais concretos.

3.1. SCADE SUITE

O SCADE Suite é um programa idealizado para desenvolver *software* dedicado a sistemas de segurança crítica. Como exemplo, é possível destacar a sua utilização no desenvolvimento

de sistemas para as áreas do setor ferroviário, automóvel e aeronáutico [35]. Este tipo de sistema é normalmente classificado como sistema de tempo real, devido à necessidade de reagirem rapidamente a qualquer interação inerente ao sistema. As linguagens de programação *Signal*, *Esterel* e *Lustre* foram idealizadas para serem utilizadas no desenvolvimento de sistemas de tempo real. Estas linguagens, e outras que não foram mencionadas, estão por detrás das funcionalidades fornecidas pelo SCADE Suite [36]. No SCADE Suite, a modulação do *software* é baseada em módulos. O desenvolvimento modelar permite que cada módulo seja validado antes de ser integrado no sistema final. Esta estrutura modelar é uma mais valia pois permite que o código por detrás desse módulo possa ser reutilizado facilmente, o que permite poupar tempo e recursos.

O SCADE Suite possui várias ferramentas tais como um editor gráfico, um simulador, ferramentas de validação e um gerador de código automático, como é possível verificar na Figura 17. O desenvolvimento dos módulos é realizado no editor gráfico e existe a possibilidade de programar de forma gráfica (diagramas de blocos) ou textual, ambas baseadas na linguagem *Lustre*, o que possibilita a conversão da programação gráfica em textual. Em cada módulo podem ser criados operadores, designados por funções noutras linguagens de programação, consoante as necessidades dos programadores. O programador pode recorrer a vários operadores disponibilizados pela ferramenta, tais como operadores matemáticos, lógicos, de comparação/decisão, *arrays*, entre outros. Através do SCADE Suite é possível verificar se existem erros na programação. Como exemplo, deteta se existem variáveis que não foram inicializadas ou que não são utilizadas, a coerência dos tipos de dados, a falta de definições, entre outros. Existe a possibilidade de simular o modelo desenvolvido ciclicamente ou de forma continua e observar o estado das variáveis ao longo deste processo. O modelo desenvolvido no SCADE Suite pode ser convertido para a linguagem de programação C através da ferramenta *qualified code generator* (KCG), sem existir a necessidade de validar o código gerado, isto porque a ferramenta é certificada pela empresa TÜV SÜD de acordo com a norma EN 50128 até SIL4 [37]. O KCG é uma ferramenta imprescindível que agiliza a transição entre a fase de modelação e implementação de um sistema.

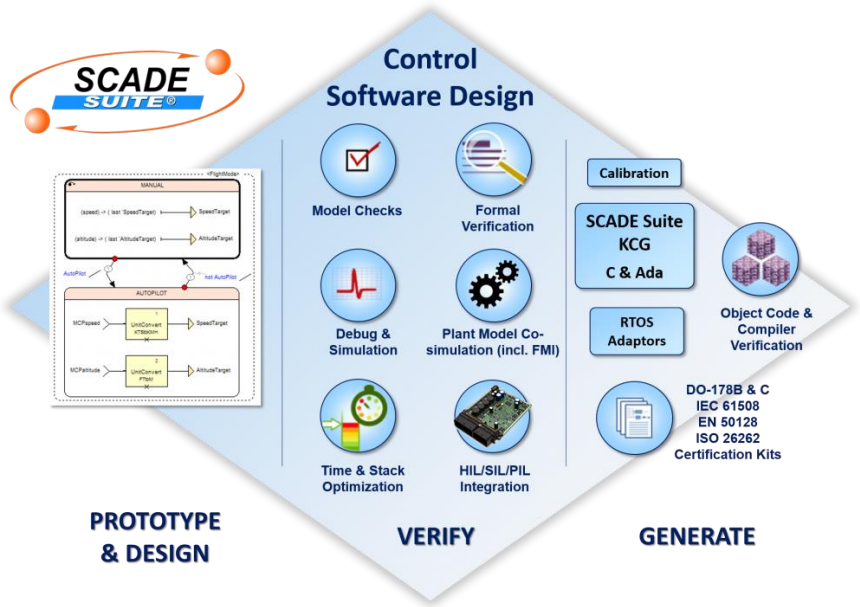


Figura 17 – Ferramentas do SCADE Suite [38].

3.1.1. SCADE *Application Programming Interface*

É possível manipular o conteúdo dos projetos desenvolvidos no SCADE através de comandos de leitura/escrita disponibilizados pelas *Application Programming Interface* (API) para as linguagens de programação Java, Python e *Tool Command Language* (TCL). A principal diferença entre estas API é que o código desenvolvido através da API do TCL e Python pode ser testado diretamente no ambiente do SCADE. A API do Python pode ser utilizada como um *Plug-in* para o *software* Python, enquanto que a API do Java é um *Plug-in* dedicado exclusivamente para o *software* Eclipse.

A informação por detrás dos ficheiros do SCADE é estruturada através da notação *Unified Modeling Language* (UML). O SCADE disponibiliza meta modelos UML, Figura 18, que contêm informação sobre a organização da estrutura dos dados. Estes dados são manipulados através de comandos de escrita/leitura das API, o que permite desenvolver ferramentas de integração.

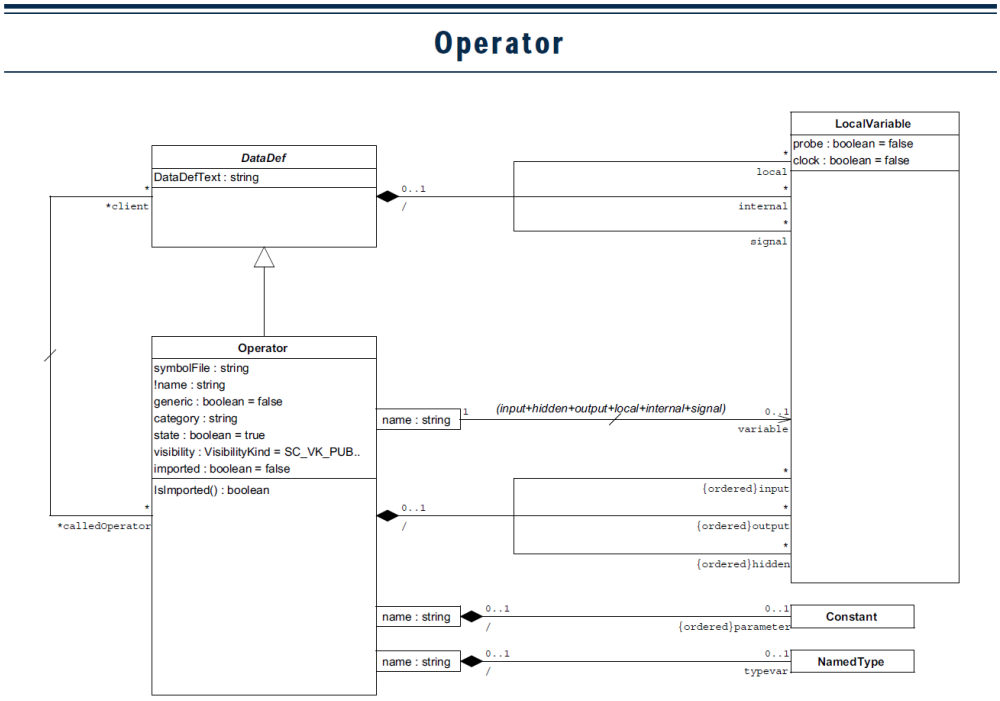


Figura 18 – Exemplo de meta modelo do SCADE Suite [39].

3.2. SILworX

O SILworX é um programa desenvolvido exclusivamente para programar os PLC da marca HIMA. Dentro do leque de PLC desta marca destacam-se para o setor ferroviário os PLC HIMax e o HIMatrix, ambos certificados pela empresa TÜV SÜD de acordo com as normas EN 50126, EN 50128 e EN 50129 até SIL 4 [40],[41], [42]. Os PLC da HIMA possuem um sistema operativo (OS) proprietário. Só os equipamentos que possuem a versão 7 ou superior desse OS, podem ser programados através do SILworX. Os equipamentos com versões mais antigas são programados através do programa ELOP II Factory [43].

Através do SILworX é possível programar recorrendo a diagramas de blocos ou de forma textual. Os diagramas de blocos desta ferramenta são baseados na linguagem de programação C++. Assim sendo, é necessário readaptar o código gerado pelo KCG de acordo com as especificações do SILworX. Esta readaptação implica que as variáveis, *inputs* e *outputs*, das funções desenvolvidas no SCADE Suite sejam importadas corretamente para SILworX [44]. É neste processo que surge a necessidade de desenvolver o projeto apresentado nesta tese. O modelo desenvolvido através do SCADE Suite é novamente validado no SILworX, antes de ser integrado nos PLC, de acordo com as normas EN 50128 [45].

3.2.1. HIMA HARDWARE

Os autómatos da HIMA são utilizados por vários fabricantes de soluções ferroviárias assim como a EFACEC, COLAS Rail, Mipro, Movares, entre outros [40]. A EFACEC utiliza os autómatos da HIMatrix, por exemplo, nos armários de controlo, Figura 5, para implementar os SSF.

Existem vários modelos do PLC HIMatrix como por exemplo o F20, F30, F35, F60 etc. [46]. A diversificação de modelos permite que se opte pela solução mais adequada para o projeto em causa, consoante o número de agulhas e sinais de um determinado projeto. Vários modelos garantem o seu funcionamento entre -25ºC e 70ºC, uma mais valia na implementação de sistemas ferroviários em locais onde as condições climatéricas possam ser mais adversas. Como exemplo, o HIMatrix F35 03 possui 24 *inputs* digitais, 8 *inputs* analógicos, 8 *outputs* digitais monitorizados através de *watchdog*, 2 contadores, 4 conectores RJ-45 (*Ethernet*) e 3 conectores D-sub de 9 pinos (*Fieldbus*) [47]. Através das interfaces de *Ethernet* e *Fieldbus* podem ser estabelecidos vários protocolos de comunicação para integrar os diferentes equipamentos da HIMA. Os protocolos de comunicação estão detalhados em [48].

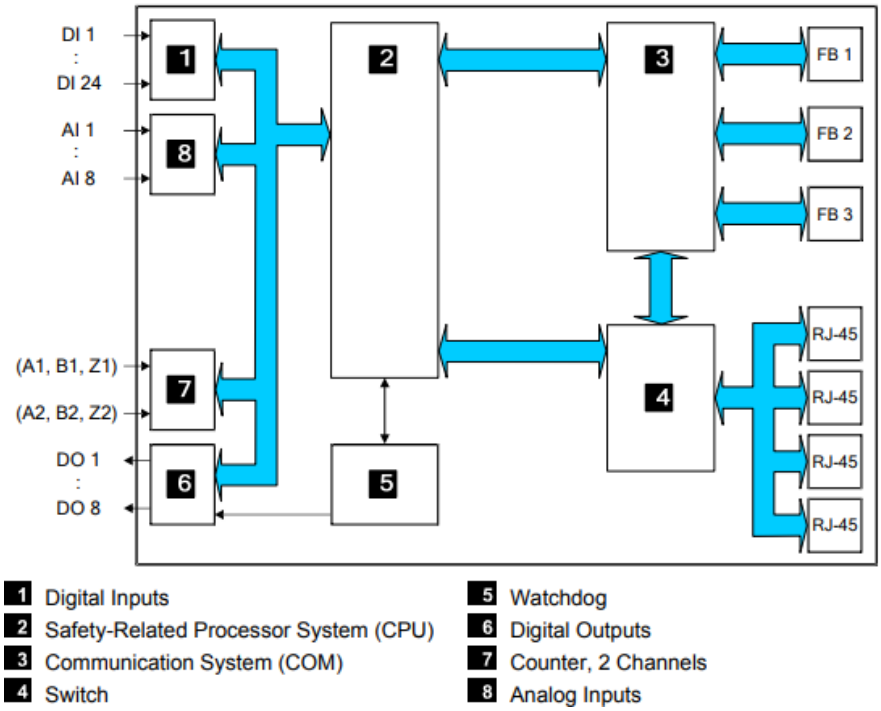


Figura 19 – Diagrama de Blocos do HIMatrix F35 03 [47]

3.3. DESCRIÇÃO DETALHADA DO PROJETO

Nos tópicos anteriores foram apresentados os programas utilizados na fase de modelação e implementação com o intuito de reforçar a necessidade de desenvolver o projeto apresentado nesta tese. Através dos tópicos anteriores é percetível que grande parte da modelação do sistema é realizada no SCADE Suite. A modelação do sistema de acordo com as características do SILworX implica que o programador manipule vários dados do projeto desenvolvido no SCADE Suite. Esta manipulação e outras tarefas que o programador tem de realizar são abordadas mais à frente neste tópico. De uma maneira geral, neste projeto é desenvolvida uma ferramenta com o intuito de eliminar a etapa de manipulação manual do programador, representada na Figura 20, como elo entre o SCADE Suite e o SILworX. A partir deste momento, pretende-se introduzir conceitos que estão por de trás do funcionamento do KATOS.

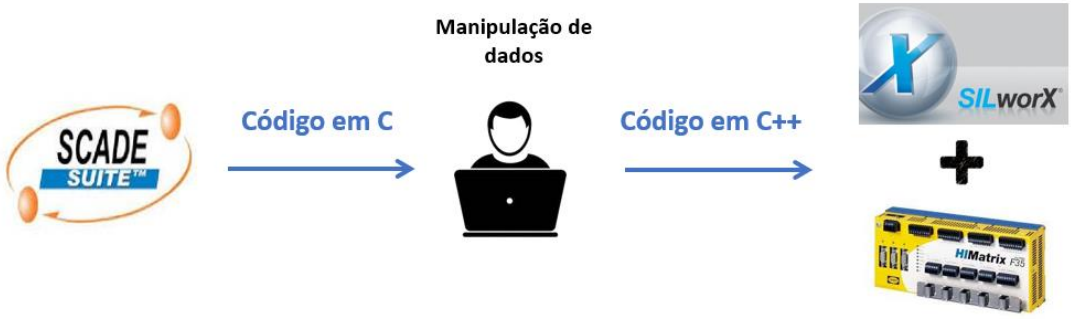


Figura 20 – Processo de desenvolvimento de um SSF.

3.3.1. ESTRUTURA DO PROJETO SCADE SUITE

Neste subtópico é apresentada a estrutura de um projeto desenvolvido no SCADE Suite para que posteriormente seja possível elucidar o leitor sobre o funcionamento do KATOS.

Na Figura 21 está representada a estrutura de um projeto de um SSF desenvolvido no SCADE Suite. Dentro dos retângulos vermelhos estão representados os modelos e dentro dos retângulos azuis os *packages* que pertencem ao primeiro modelo.

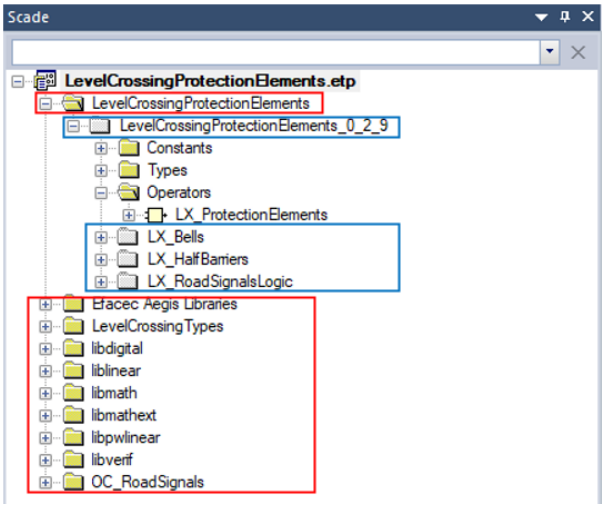


Figura 21 – Estrutura de um Projeto SCADE Suite

É possível verificar que o primeiro package possui três subdiretórios e três packages. No diretório *Constants* estão definidas variáveis constantes, o tipo de dado (inteiro, *boolean*, etc.) e o seu valor. No diretório *Types* são apresentadas estruturas, os campos de cada estrutura e o tipo de dado de cada campo. Através do diretório *Operators* é possível aceder às funções/operadores desenvolvidas para controlar os SSF. Estas funções não são discutidas neste documento, embora sejam responsáveis pelo encravamento dos vários elementos ferroviários. Todavia, será descrita a dependência entre os *Inputs* e *Outputs* (IO), variáveis utilizadas para configurar as funções de encravamento, e os vários modelos que fazem parte do projeto.

À medida que são configuradas as funções, os IO utilizados ficam disponíveis no diretório *Interface*, como é representado na Figura 22. Estes IO são variáveis do sistema e, portanto, a sua inicialização obriga à definição do tipo de dado.

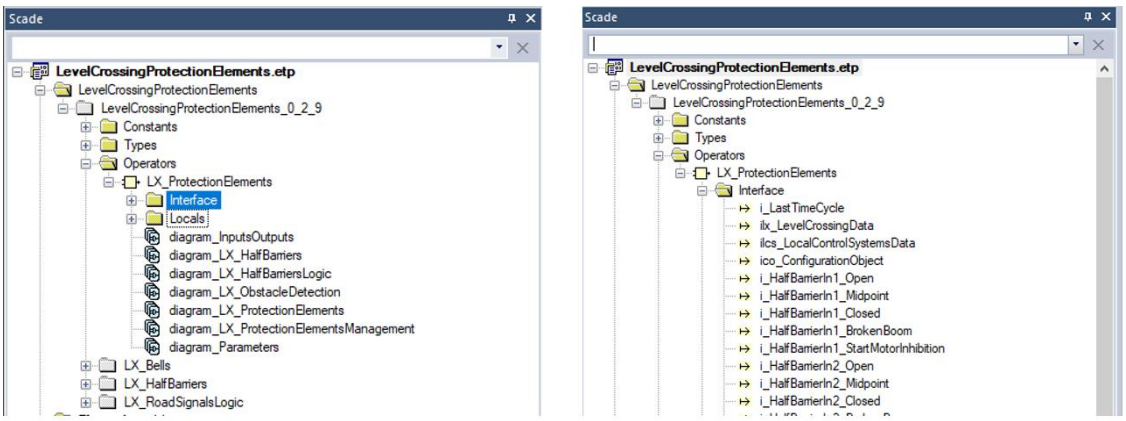


Figura 22 – *Inputs* e *Outputs* de uma função.

Os tipos de dados podem ser tipos básicos ou compostos (estruturas, *array*). Quando se define um tipo de dado para uma determinada variável, esta pode ser associada a qualquer estrutura presente num dos modelos do projeto. O mesmo acontece, por exemplo, ao definir o tamanho de um *array*, no qual pode ser utilizada uma qualquer constante definida em qualquer um dos modelos do projeto. O tipo de dado de um campo de uma estrutura pode também ser um tipo básico ou composto. Na Figura 23 estão representados vários campos de uma estrutura. Nos retângulos vermelho e azul pretende-se destacar um tipo de dado composto. No retângulo vermelho, o tipo de dado é um *array* do tipo "T_PE_BellControl_Parameters" (estrutura) com tamanho "K_NrOfBells" (constante do tipo inteiro). No retângulo azul, o tipo de dado é "T_PE_RoadSignal_Logic_Parameters" (estrutura).

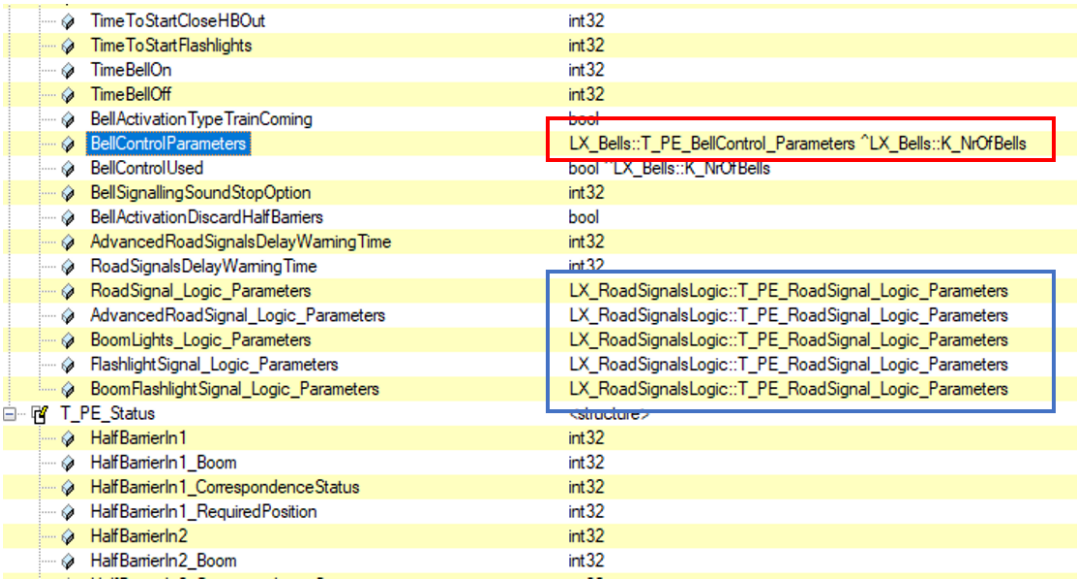


Figura 23 – Estrutura de dados.

É, portanto, percetível que qualquer campo de uma estrutura pode depender de outras estruturas definidas em qualquer uma das pastas *Types* assinaladas na Figura 24. Um dos procedimentos a realizar pelo programador para modelar o SILworX de acordo com o projeto do SCADE Suite é desfragmentar recursivamente os IO até ao seu tipo básico.

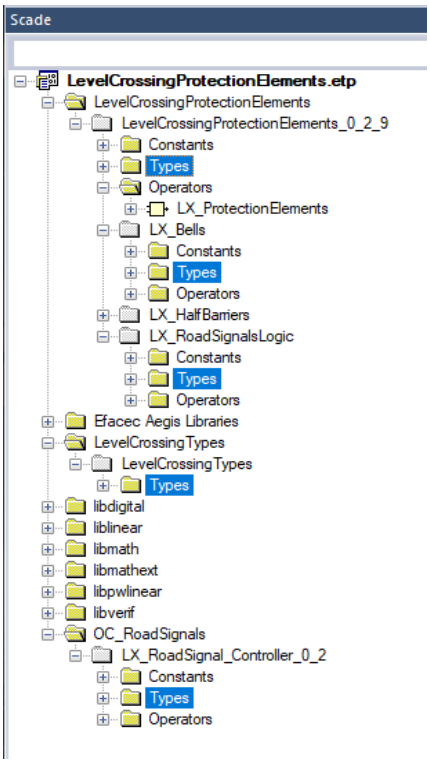


Figura 24 – Subdiretórios dos vários modelos que contêm estruturas.

De seguida é apresentado um exemplo da desfragmentação de uma hipotética função que possui apenas dois IO, Tabela 4.

Tabela 4 – IO utilizados no exemplo.

IO	
Nome da Variável	**Tipo de dado**
Var1	int32
Var2	Estrutura_N1

A Var1 é do tipo básico e a Var2 é do tipo composto, sendo que depende da Estrutura_N1. O Campo5 da Estrutura_N1 é também do tipo composto, pelo que depende da Estrutura_N2. As estruturas estão representadas na Tabela 5.

Tabela 5 – Estrutura utilizada no exemplo.

Estrutura_N1	
Campos da Estrutura	**Tipo de dado**
Campo1	int32
Campo2	bool
Campo3	int32
Campo4	bool
Campo5	Estrutura_N2

Estrutura_N2	
Campos da Estrutura	**Tipo de dado**
Campo1	int32 ^3
Campo2	bool

O processo de desfragmentação consiste em encontrar todas as dependências de um IO até que este não dependa de uma variável de tipo de dado composto. O objetivo é obter o nome final do IO depois de desfragmentado. Na Tabela 6 encontra-se representada a desfragmentação dos dados da Tabela 4.

Tabela 6 – Exemplo de Desfragmentação IO.

Desfragmentação IO			
Nome Var	**Tipo Básico**	**Tipo Composto**	**Nome Final**
Var1	int32		Var1
Var2		Estrutura_N1	
Var2.Campo1	int32		Var2.Campo1
Var2.Campo2	bool		Var2.Campo2
Var2.Campo3	int32		Var2.Campo3
Var2.Campo4	bool		Var2.Campo4
Var2.Campo5		Estrutura_N2	
Var2.Campo5.Campo1		int32 ^3	
Var2.Campo5.Campo1[0]	int32		Var2.Campo5.Campo1[0]
Var2.Campo5.Campo1[1]	int32		Var2.Campo5.Campo1[1]
Var2.Campo5.Campo1[2]	int32		Var2.Campo5.Campo1[2]
Var2.Campo5.campo2	bool		Var2.Campo5.Campo2

É necessário realizar este processo para todos os IO de uma qualquer função que será implementada no SILworX. Neste caso, só existiam 2 IO e as estruturas apresentadas como exemplo eram bastante simples. No entanto, deram origem a 9 variáveis. É percetível que numa função onde possam existir dezenas de IO, o processo de desfragmentação é bastante complexo. Esta desfragmentação é realizada manualmente e é uma tarefa morosa que pode originar erros por parte do programador. Erros estes que se vão repercutir ao longo do processo de integração do modelo de acordo com o SILworX, o que origina um processo de verificação mais prolongado.

Automatizar o processo de desfragmentação dos IO é um dos objetivos principais do KATOS, assim como os que serão apresentados no tópico 3.3.2. Essa automatização só é possível devido às API do SCADE.

3.3.2. REQUISITOS DO KATOS

Neste tópico são especificados quais os requisitos a que o KATOS deve obedecer para que as funções que controlam os elementos ferroviários, desenvolvidas no SCADE Suite, sejam modeladas autonomamente de acordo com as características do SILworX. Como já foi referido anteriormente, o modelo desenvolvido no SCADE Suite pode ser convertido para a

linguagem de programação C através da ferramenta KCG. No fim desta conversão são gerados vários ficheiros com esse código, os quais, daqui em diante, serão designados como ficheiros KCG.

O KATOS deve ser capaz de realizar a desfragmentação dos IO, apresentada em 3.3.1, autonomamente, independentemente dos tipos de dados configurados no SCADE Suite. Os dados obtidos na desfragmentação vão ser utilizados para editar os ficheiros KCG e criar tabelas para o SILworX. Contudo, é necessário efetuar um processamento diferente para cada situação.

Os ficheiros KCG, assim como outros três ficheiros afetos ao SILworX, têm de ser editados, sendo, portanto, necessário que o KATOS interprete o conteúdo desses ficheiros e execute as alterações autonomamente. A manipulação destes ficheiros não é abordada devido a termos de confidencialidade. No entanto, fica a ideia que para além de ser necessário editar os ficheiros, alguns serão renomeados e apagados.

O KATOS deve gerar várias tabelas em ficheiros *comma-separet values* (CSV). Deve ser criada uma tabela por cada *array* ou estrutura, identificado na desfragmentação, bem como duas outras tabelas através das quais é possível importar todas as variáveis obtidas na desfragmentação para o SILworX. Uma destas tabelas deve conter os nomes e os tipos das variáveis desfragmentadas, enquanto a outra tabela, para além de conter os dados das variáveis desfragmentadas, deve também conter os dados de variáveis do tipo composto que originaram variáveis do tipo básico. Através desta última tabela é possível estabelecer dependências entre as tabelas dos *array* ou estruturas e as variáveis importadas.

Apesar de ser esperado que o KATOS realize todo esse processamento automaticamente, é necessário que o utilizador introduza alguns dados para iniciar o processo. Os dados a introduzir são os seguintes:

- Projeto desenvolvido no SCADE SUITE (caminho e nome do ficheiro);
- Package que possui o operador/função convertido com o KCG;
- Operador;
- Prefixo utilizado pelo KCG (necessário para a manipulação dos ficheiros KCG);
- Indicação do diretório que contém os ficheiros KCG;
- Valor de uma variável afeta ao SILworX (necessária para manipulação dos ficheiros KCG e tabelas);

- Indicação do diretório que irá conter os ficheiros gerados pelo KATOS.

Para que o KATOS seja uma solução robusta, devem ser introduzidos vários processos de verificação ao longo do programa. Através destas verificações será possível apresentar ao utilizador alguns dados acerca da geração ou sobre possíveis erros que possam ocorrer.

De modo a facilitar a interação do utilizador com o KATOS, a solução deve possuir um *graphical user interface* (GUI) através do qual será possível introduzir os dados necessários à geração e visualizar as informações que permitem atestar o funcionamento do KATOS.

Ao longo deste capítulo foi evidenciada a necessidade de desenvolver o KATOS e foram apresentados os requisitos impostos. Através da análise destes requisitos é possível verificar que o KATOS pode ser classificado como uma ferramenta de classe T3, pelo que os processos de verificação introduzidos ao longo do seu funcionamento são imprescindíveis para que eventualmente seja possível certificar a ferramenta. No próximo capítulo será abordada a estrutura do *software* por detrás do KATOS.

4. PLANIFICAÇÃO DO *SOFTWARE*

Neste capítulo é abordada a estrutura do KATOS. Não será apresentado qualquer código desenvolvido para esta ferramenta devido a termos de confidencialidade. No entanto, pretende-se elucidar o leitor sobre o seu funcionamento através de diagramas e fluxogramas. Inicialmente, é apresentada a arquitetura geral do KATOS e de seguida é descrito o seu funcionamento.

4.1. ARQUITETURA GERAL DO *SOFTWARE*

Tal como referido anteriormente, o KATOS possui um GUI no qual são introduzidos os parâmetros necessários à adaptação do projeto desenvolvido no SCADE Suite segundo as características do SILworX. Os parâmetros introduzidos pelo utilizador são designados por *inputs* do KATOS. Alguns dos *inputs* do KATOS podem ser pré-preenchidos, com informação disponibilizada no projeto SCADE SUITE.

De modo a compreender a estrutura do *software* por detrás do KATOS pode-se recorrer à Figura 25. De uma forma geral o KATOS é dividido em três processos. Um dos processos é afeto ao GUI, o qual, para além de ser responsável por toda a interação do KATOS com o

utilizador, é responsável por validar os *inputs* introduzidos que serão argumentos dos outros processos. Consoante a validação desses *inputs* é possível executar os processos. Assim que o processo responsável pelo pré-preenchimento dos *inputs* do KATOS termine é gerado um ficheiro, ficheiro este que é interpretado pelo processo do GUI que preenche automaticamente os respetivos campos. Por último, o processo afeto à manipulação de dados é responsável por realizar a maior parte das necessidades sobre a edição e criação de ficheiros, apresentadas em 3.3.2. Novamente, assim que este processo termine, é gerado um ficheiro que será interpretado pelo GUI, através do qual é possível apresentar ao utilizador várias informações acerca da conversão gerada.

Os programas utilizados para desenvolver o GUI e os outros dois processos principais, manipulação de dados e pré-preenchimento, são diferentes, o que suscitou a necessidade de desenvolver ficheiros de verificação para atestar o seu funcionamento. Existe um ficheiro executável tanto para o processo de manipulação de dados, como para o de pré-preenchimento. Através do GUI é possível validar os argumentos utilizados para executar estes processos, o que permite acautelar o seu bom funcionamento. Os programas utilizados para o desenvolvimento do KATOS, os ficheiros de verificação e os avisos apresentados ao utilizador são abordados no capítulo 5.

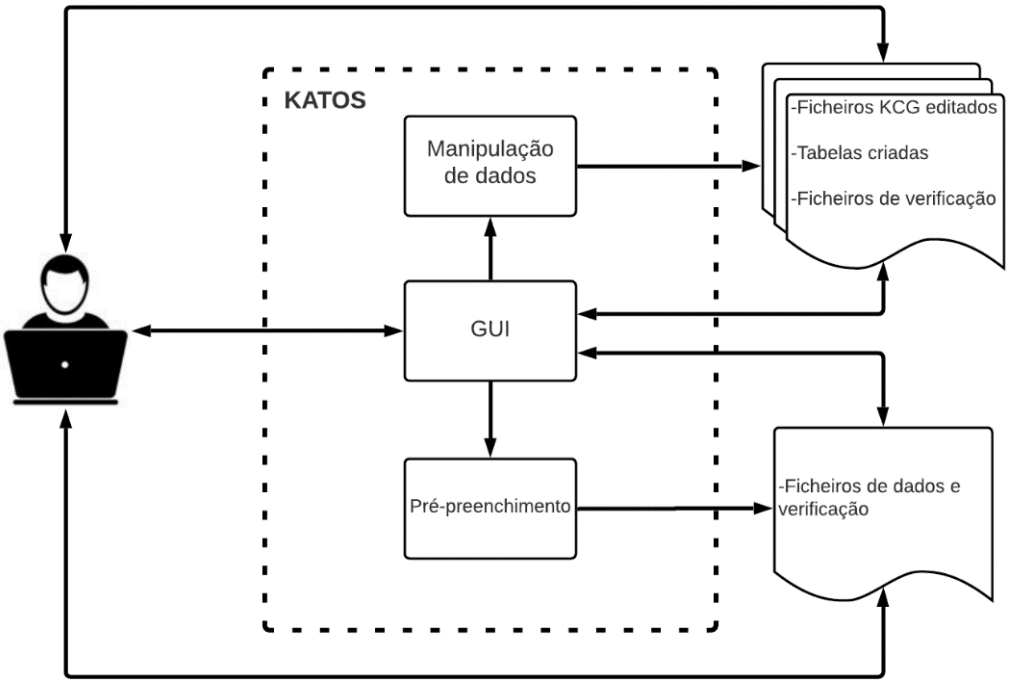


Figura 25 – Arquitetura geral do *Software* do KATOS

4.2. PRINCÍPIO DE FUNCIONAMENTO DO *SOFTWARE*

No tópico anterior foi apresentada a arquitetura geral do *software*. De modo a compreender a interação entre os processos do KATOS, estes serão aprofundados nos próximos subtópicos.

4.2.1. PROCESSO DO GUI

Este subtópico é dedicado ao processo do GUI. O seu funcionamento encontra-se representado no fluxograma da Figura 26.

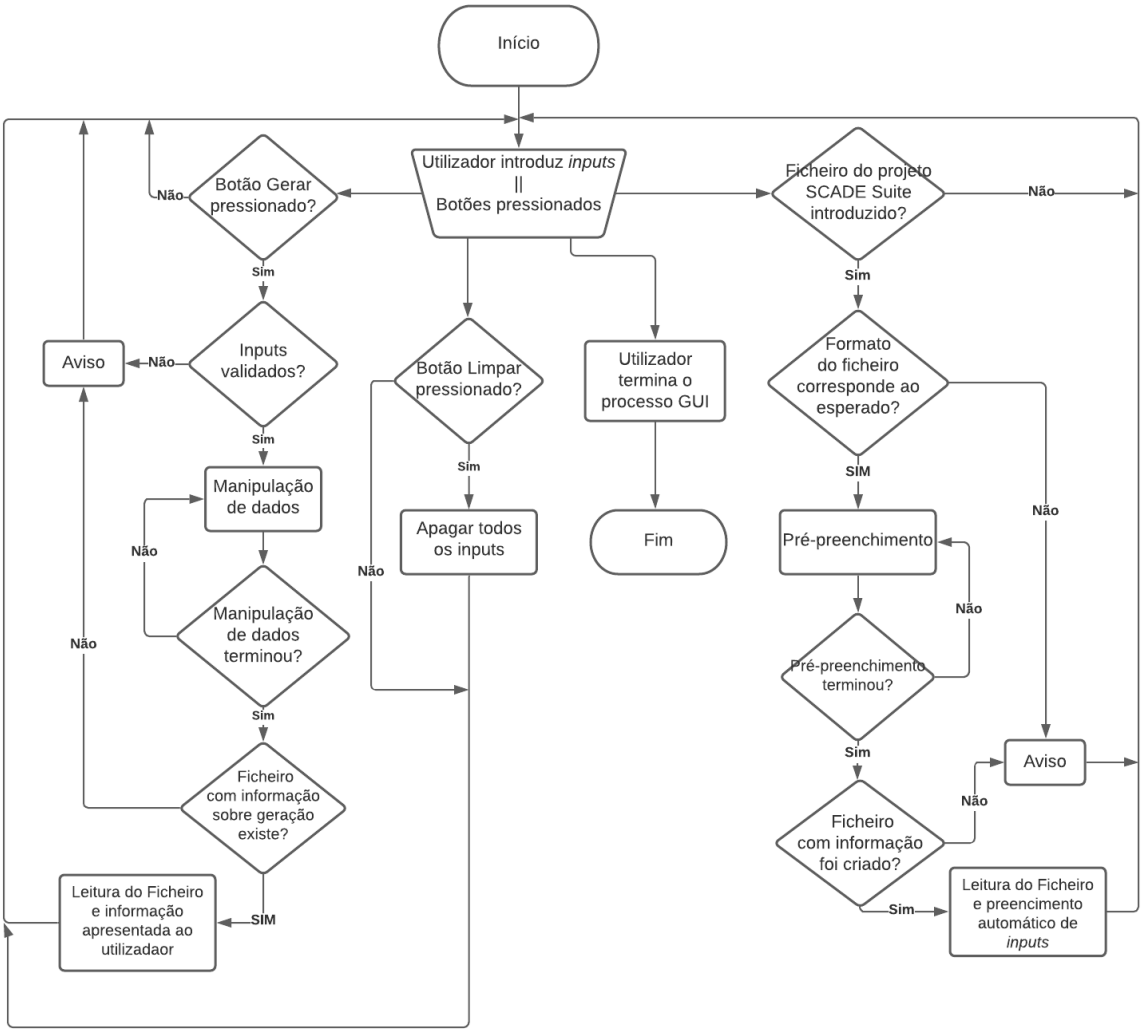


Figura 26 – Fluxograma do processo GUI.

Como o GUI depende da interação do utilizador para que sejam realizadas ações, o fluxograma só apresenta uma estrutura sequencial a partir do bloco que representa uma operação manual.

De uma maneira geral, este processo é responsável por:

- Validar o projeto SCADE Suite e, consequentemente, preencher alguns dos *inputs* do KATOS;
- Validar os *inputs* do KATOS e, subsequentemente, dar início ao processo de conversão das configurações segundo as especificações do SILworX;
- Apresentar avisos ao utilizador (antes e após a execução dos processos principais).

4.2.1. PROCESSO DE PRÉ-PREENCHIMENTO

O processo de pré-preenchimento encontra-se representado no fluxograma da Figura 27.

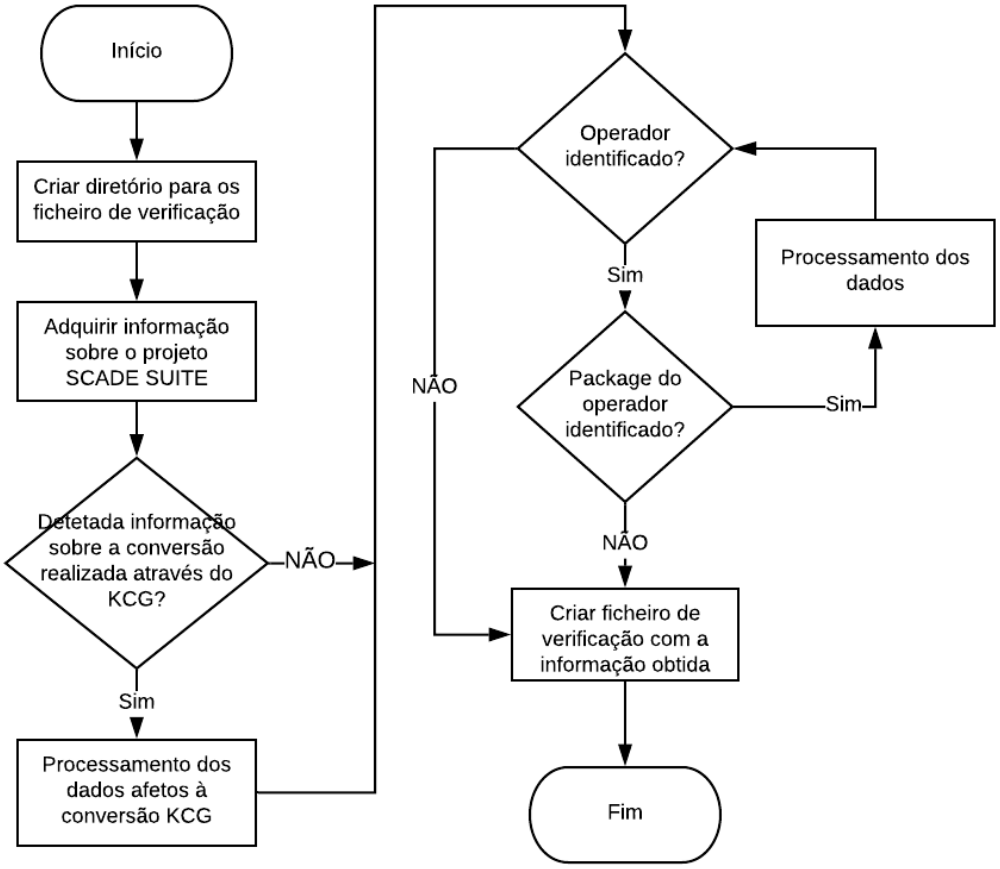


Figura 27 – Fluxograma do processo pré-preenchimento.

Só é expectável que o KATOS seja utilizado depois do projeto desenvolvido no SCADE Suite ser convertido para a linguagem de programação C, através do KCG. Através do projeto SCADE Suite é possível obter informação acerca da geração do KCG, o que permite preencher automaticamente os seguintes *inputs* do KATOS:

- Package que possui o operador/função;
- Operador;
- Prefixo (necessário para a manipulação dos ficheiros KCG).

Se não for encontrada nenhuma informação a respeito dos dados afetos à geração do KCG, não será gerado qualquer erro ou aviso visto que estes dados podem já não estar presentes no projeto do SCADE Suite. O utilizador poderá posteriormente alterar os *inputs* introduzidos automaticamente, pois podem ter sido alterados indevidamente. Caso não seja encontrado nenhum operador no projeto do SCADE SUITE é apresentado um erro, visto que não será possível encontrar os IO do operador. Em caso da ocorrência de algum erro, previsto ou não, este é identificado e é reportado no ficheiro de verificação para que seja apresentado ao utilizador. Todavia, este processo não se encontra representado no fluxograma da Figura 27 porque se tornaria demasiado extenso e dificultava a sua compreensão.

Este processo será responsável por:

- Recolher informação sobre dados utilizados na geração do KCG;
- Recolher informação acerca dos operadores e respetivas packages;
- Validação dos dados apresentados pelo projeto SCADE Suite;
- Criar um ficheiro de verificação, devidamente estruturado, que contenha informação acerca dos dados obtidos ou possíveis erros.

4.2.2. PROCESSO DE MANIPULAÇÃO DE DADOS

Este subtópico é dedicado ao processo de manipulação de dados, o seu funcionamento encontra-se representado no fluxograma da Figura 28.

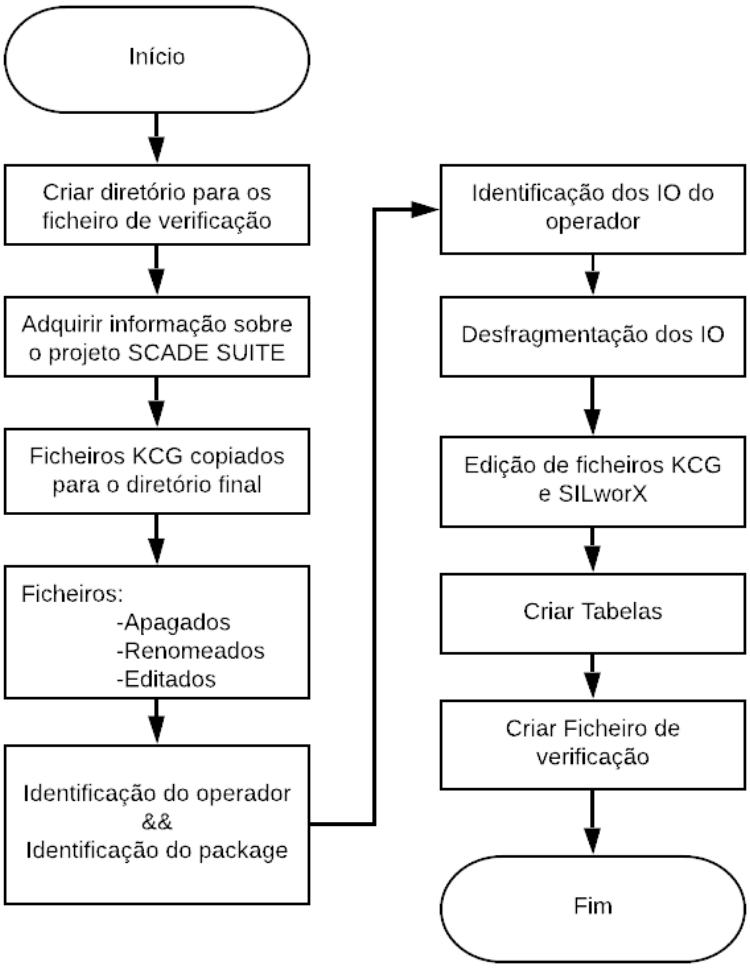


Figura 28 – Fluxograma do processo manipulação de dados.

Neste tópico só é abordado com algum detalhe a função de desfragmentação dos IO. As funções que permitem identificar os dados do SCADE Suite e as diversas verificações ao longo do processo não são aprofundadas. O fluxograma da Figura 28 aparenta ser bastante básico, todavia é meramente representativo e não reflete a complexidade do *software* desenvolvido.

Para não tornar o fluxograma da Figura 28 ilegível e facilitar a compreensão da função de desfragmentação dos IO, esta é descrita através do fluxograma da Figura 29 e Figura 30.

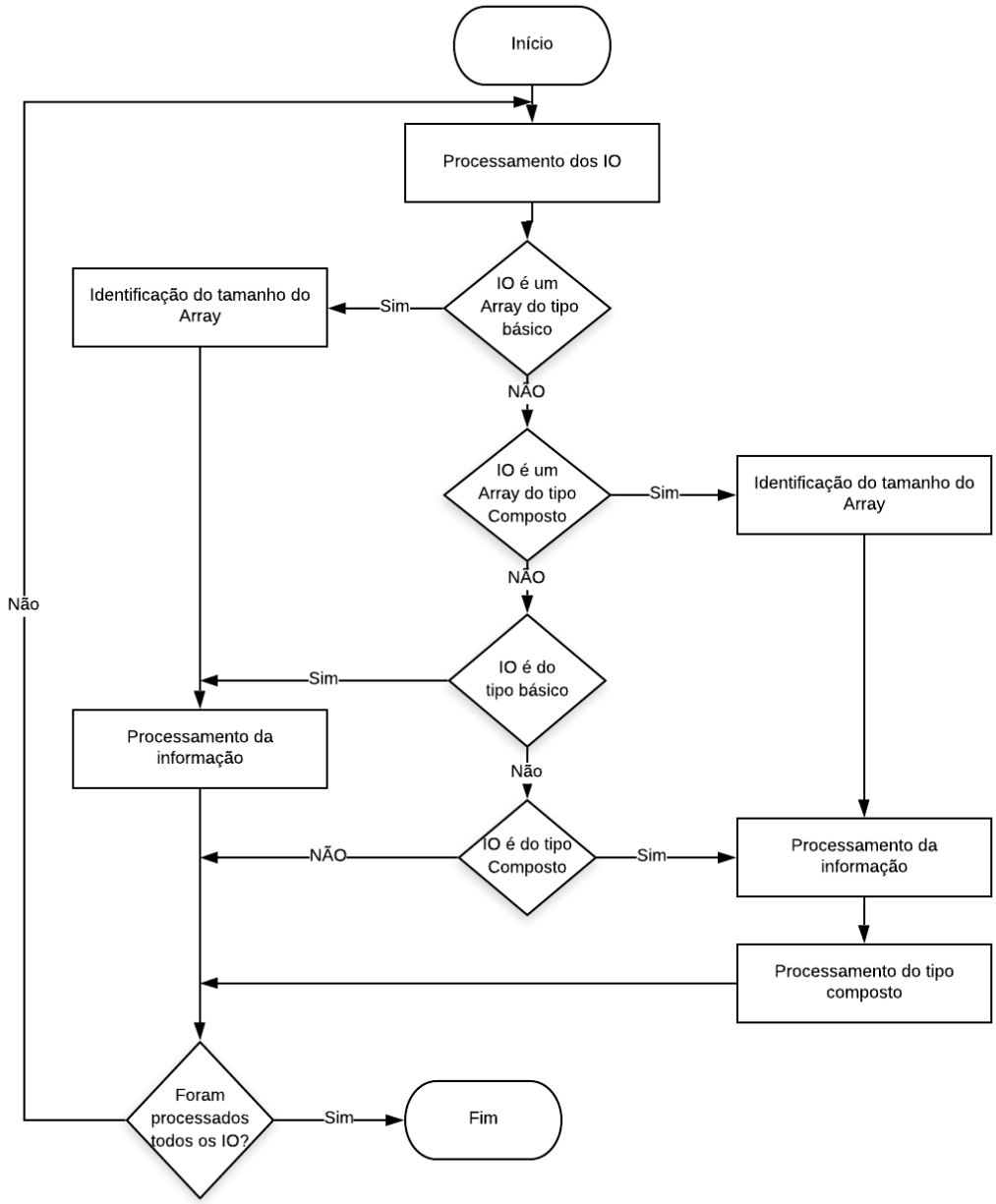


Figura 29 – Fluxograma da função desfragmentação dos IO

O fluxograma da Figura 29 retrata o processo de desfragmentação de todos os IO do operador selecionado. Através do processamento destes IO é possível identificar o tipo de dado de cada um. Consoante o tipo de dado é realizado um processamento que irá auxiliar o processo de criação de tabelas e edição de ficheiros. Quando é identificado um IO do tipo composto é necessário recorrer à função processamento do tipo composto, representada na Figura 30.

Apesar dos fluxogramas serem bastante idênticos é necessário referir que a identificação dos campos da estrutura é bastante mais complexa do que a identificação dos IO do operador.

No fluxograma da Figura 30 é percetível que o processamento do tipo composto só termina quando todos os campos de uma estrutura forem processados. Se algum dos campos da estrutura for do tipo composto é necessário recorrer novamente a esta função.

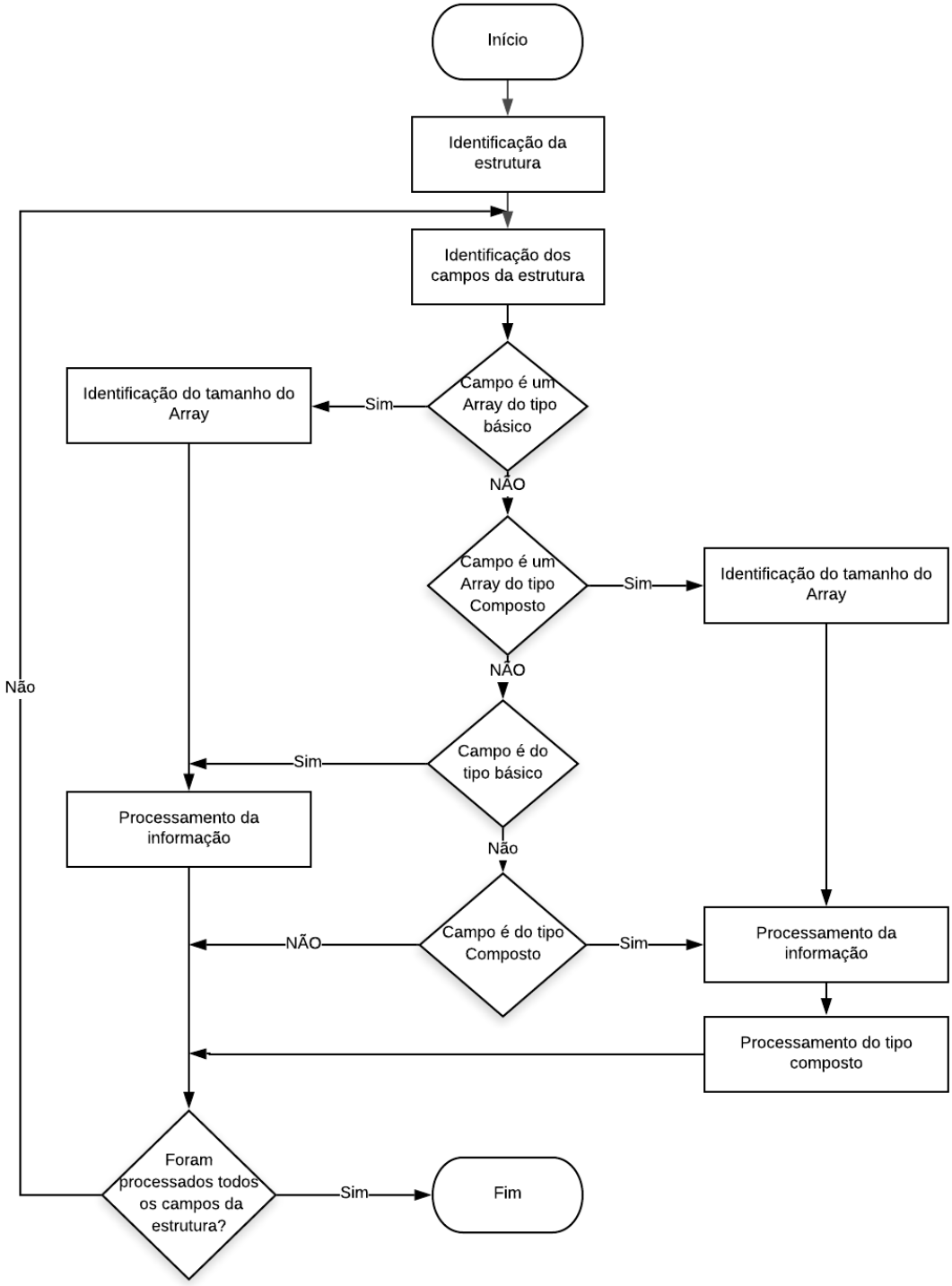


Figura 30 – Fluxograma da função processamento do tipo composto.

De uma maneira geral este processo será responsável por:

- Identificar corretamente o operador e respetiva package;
- Desfragmentar os IO desse operador;
- Editar ficheiros KCG e SILworX;
- Criar as tabelas necessárias;
- Criar um ficheiro de verificação, devidamente estruturado, que contenha informação acerca da manipulação de dados ou possíveis erros.

De mencionar que em caso da ocorrência de algum erro, previsto ou não, este é identificado e é reportado no ficheiro de verificação para que seja apresentado ao utilizador. Todavia este processo não se encontra representado em nenhum dos fluxogramas anteriores porque os tornaria demasiado extensos dificultava a sua compreensão.

Este capítulo pretende elucidar o leitor para o funcionamento do KATOS para que de seguida, no capítulo 5, entenda as decisões tomadas, os programas utilizados, os ficheiros de verificação e os avisos apresentados ao utilizador.

5. IMPLEMENTAÇÃO

Neste capítulo é apresentada a conceção do KATOS. Primeiramente, é abordada a API do SCADE utilizada para interpretar o conteúdo do projeto SCADE Suite e que permite desenvolver o processo que diz respeito à manipulação de dados. De seguida, é apresentado o desenvolvimento da interface gráfica do KATOS e alguns cuidados a ter na geração dos ficheiros executáveis afetos ao processo de manipulação de dados e pré-preenchimento. São abordadas as verificações realizadas pelo processo do GUI, antes de executar os ficheiros executáveis, e os ficheiros de verificação gerados pelos ficheiros executáveis. Por último, é exposto o desenvolvimento do ficheiro de instalação do KATOS.

5.1. MANIPULAÇÃO DO PROJETO SCADE SUITE

O primeiro passo no desenvolvimento deste projeto consiste em selecionar a API que será utilizada para manipular o conteúdo do projeto desenvolvido no SCADE Suite. Das API mencionadas em 3.1.1, optou-se por utilizar a API do Java. Esta escolha não se deve a factos técnicos. Comparativamente com as linguagens de programação das restantes API, a linguagem Java é a mais utilizada por parte da uTRP e poderá ser favorável manter essa conformidade.

Como já foi referido anteriormente, a API do Java é um *Plug-in* dedicado exclusivamente para o *software* Eclipse. O fabricante do SCADE Suite disponibiliza os procedimentos que devem ser realizados para configurar corretamente o *Plug-in* no Eclipse [49].

Após garantir a correta configuração do *Plug-in* é possível dar os primeiros passos na programação. Em primeiro lugar, é necessário garantir que é possível adquirir informação acerca do projeto SCADE Suite. A partir do momento em que é possível aceder ao conteúdo do projeto SCADE SUITE e identificar o *Package*, operador e respetivos IO, é possível desenvolver sequencialmente o código responsável pelo processo de manipulação de dados, apresentado em 4.2.2.

Assim que foi garantido o correto funcionamento do processo de manipulação de dados, deu-se início ao desenvolvimento do GUI.

5.2. INTERFACE GRÁFICA DO KATOS

Um dos principais objetivos do KATOS é que seja uma ferramenta intuitiva e *user-friendly*. Daí a necessidade de desenvolver um GUI responsável por toda a interação do KATOS com o utilizador. Neste projeto, a implementação do GUI torna-se essencial a partir do momento em que é garantido o correto funcionamento do processo de manipulação de dados.

Existem vários softwares dedicados ao desenvolvimento de GUI. De seguida são apresentados o *Swing* e o *Windows Presentation Foundation* (WPF). O *Swing* é uma API que pertence ao *Java Foundation Classes* (JFC) e permite desenvolver interfaces gráficas através da linguagem de programação Java. O *Swing* tem por base algumas das características do *Abstract Window Toolkit* (AWT). Contudo, as aplicações desenvolvidas através do Swing podem ser interpretadas por qualquer OS, o que não se verifica com o AWT [50], [51]. O WPF faz parte da *Framework .NET* desenvolvida pela Microsoft. Esta ferramenta é dedicada ao desenvolvimento de interfaces gráficas. No WPF a aparência/estilo da aplicação é criada através da formatação *Extensible Application Markup Language* (XAML), uma variação do XML, e o comportamento da aplicação é programado através C#. Apesar de existir uma separação entre o código afeto à aparência e à logica, estes estão sempre associados. Como exemplo, um evento de um botão é definido no XAML, todavia a função que controla esse evento é programada através de C#. A única contrapartida desta ferramenta é que as interfaces desenvolvidas só operam no OS Windows [52], [53].

É possível configurar o ambiente do Eclipse de modo a desenvolver um GUI através da API do *Swing* [54]. Uma vez que o processo de manipulação de dados foi desenvolvido em Java, inicialmente optou-se por desenvolver o GUI através do *Swing*. Isto permite realizar o *debug* do GUI e executar o processo de manipulação de dados sem a necessidade de gerar ficheiros executáveis, visto que é utilizado o compilador de Java para ambos os processos.

A uTRP irá desenvolver a ferramenta RAS através do WPF. O KATOS eventualmente será integrado nesta ferramenta, pelo que pode ser vantajoso, para esta integração, desenvolver o processo do GUI através do WPF. Apesar de os projetos desenvolvidos através do WPF operarem exclusivamente no OS Windows, este não é um fator preponderante visto que é o sistema operativo utilizado pela equipa da uTRP.

A necessidade de desenvolver o processo de pré-preenchimento surgiu na fase de estruturação do GUI, não só para facilitar a utilização do KATOS, mas também para prevenir que o utilizador insira parâmetros incorretos. Embora possa parecer inútil desenvolver o GUI através do *Swing* sabendo à partida que será utilizado o WPF, isto permite testar o funcionamento do GUI, apresentado em 4.2.1, sem necessidade de gerar ficheiros executáveis sempre que seja efetuada alguma alteração no processo de pré-preenchimento ou manipulação de dados. Basicamente, é vantajoso testar a interação entre o GUI e os outros dois processos somente através do Eclipse, sem existir a necessidade de recorrer a outro *software.*

O desenvolvimento do GUI através do WPF implica a ambientação com o *software* Visual Studio 2019 e que a programação seja efetuada em C#. A partir do momento que foi possível validar o correto funcionamento dos processos de pré-preenchimento e manipulação de dados, no *Swing*, foram dados os primeiros passos no desenvolvimento do GUI através do WPF. Depois de modelar toda a interface, recorrendo ao WPF, surge a necessidade de gerar ficheiros executáveis dos dois processos para testar a interação com o GUI.

Neste ponto do projeto foi possível identificar que ao gerar os ficheiros executáveis *Java Archive* (JAR), a opção que permite controlar a integração das bibliotecas utilizadas, funções da API do SCADE e outras, tem um grande influencia no desempenho do processo [55], [56]. Se for selecionada a opção em que o ficheiro JAR irá conter o código desenvolvido e as bibliotecas utilizadas, verifica-se que o tempo de execução dos processos é muito superior ao verificado em modo de *debug* no Eclipse. A opção que garante um tempo de execução

semelhante ao verificado no Eclipse é aquela em que o ficheiro JAR irá conter apenas o código desenvolvido e as bibliotecas são disponibilizadas num diretório dedicado, no mesmo caminho do ficheiro JAR.

5.3. VERIFICAÇÕES E RECOLHA DE DADOS DOS FICHEIROS JAR

O processo do GUI é responsável por validar os *inputs* do KATOS, apresentados em 3.3.2. Isto permite acautelar o funcionamento dos ficheiros executáveis. De seguida são apresentadas as validações de cada *input*:

- **Projeto SCADE Suite**:
  - É verificada a extensão do ficheiro, se for validado é iniciado o processo de pré-preenchimento;
  - Se o ficheiro não for validado é apresentada uma mensagem de aviso ao utilizador.
- ***Package*:**
  - *Input* preenchido automaticamente, contudo o utilizador pode selecionar qualquer package identificada pelo processo de pré-preenchimento;
  - *Input* proveniente do ficheiro de pré-preenchimento é automaticamente validado;
  - Se não for selecionado nenhum *package* e for dado início ao processo de manipulação de dados é apresentada uma mensagem de aviso ao utilizador.
- **Operador:**
  - *Input* preenchido automaticamente, contudo o utilizador pode selecionar os operadores identificados no processo de pré-preenchimento que pertençam à package selecionada;
  - *Input* proveniente do ficheiro de pré-preenchimento é automaticamente validado;
  - Se não for selecionado nenhum operador e for dado início ao processo de manipulação de dados é apresentada uma mensagem de aviso ao utilizador.
- **Diretório que contém os ficheiros KCG:**
  - É verificado se o utilizador possui permissões de leitura e escrita sobre o diretório introduzido;
  - Se não existir pelo menos um ficheiro com a extensão ‘.C’ ou ‘.H’ é apresentado um aviso ao utilizador, isto porque a geração do KCG gera obrigatoriamente estes ficheiros;
  - Ao validar este *input* é possível determinar o número de ficheiros que o processo de manipulação de dados deve gerar para que após a execução do processo seja comparado com número de ficheiros efetivamente gerado.

- **Prefixo:**
  - *Input* preenchido automaticamente;
  - O nome dos ficheiros KCG, conteúdo do diretório identificado no *input* anterior, normalmente começa pelo prefixo utilizado na geração do KCG, contudo não é obrigatório. Se algum dos ficheiros não possuir o prefixo identificado pelo processo de pré-preenchimento, o utilizador tem de introduzir um novo prefixo, manualmente, e confirmar esta alteração. A única restrição à alteração deste *input* é que não pode conter espaços, pelo que o processo do GUI não permite a sua utilização.
- **Valor de uma variável afeta ao SILworX:**
  - Esta variável é do tipo inteiro de 32-bit com sinal, no entanto, é validado se o número introduzido está compreendido entre 1 e 2147483647 (2^{31}-1);
  - Ao preencher este *input* o processo do GUI só permite a introdução de caracteres numéricos.
- **Diretório que irá conter os ficheiros gerados pelo KATOS:**
  - É verificado se o utilizador possui permissões de leitura e escrita sobre o diretório introduzido;
  - É verificado se os três ficheiros afetos ao SILworX estão presentes neste diretório. É da responsabilidade do utilizador fornecer estes três ficheiros. Se algum dos ficheiros não for identificado é apresentada uma mensagem de aviso ao utilizador;
  - Quando o utilizador dá início à execução do processo de manipulação de dados, é verificado se existe mais algum ficheiro neste diretório para além dos ficheiros afetos ao SILworX. Caso exista, o utilizador tem de confirmar que pretende prosseguir. Nesta etapa é também verificado o conteúdo dos ficheiros SILworX. Caso os ficheiros já tenham sido editados pelo menos uma vez, pelo processo de manipulação de dados, o utilizador tem de confirmar que pretende prosseguir.

Como foi mencionado no tópico 4.1, no fim da execução do processo de manipulação de dados e pré-preenchimento são gerados ficheiros que contêm informação acerca da execução destes processos.

Ambos os processos registam informação acerca da sua execução num ficheiro log. À medida que os processos iniciam ou completam determinadas tarefas, esse ficheiro é editado e é guardada essa informação. Em caso de alguma falha, o utilizador pode identificar que

função é que levou a terminar o processo. Aquando do desenvolvimento do GUI, através do WPF, este ficheiro foi bastante útil para realizar *debug* aos ficheiros JAR.

Para além do ficheiro log, ambos os processos geram ficheiros formatados em XML. Optou-se por utilizar o formato XML para estruturar a informação obtida em ambos os processos e facilitar a interpretação destes dados pelo processo do GUI.

O formato XML permite estruturar a informação através de *tags*, similar à formatação *HyperText Markup Language* (HTML). A denominação e conteúdo destas *tags* fica ao critério do utilizador, ao contrário do que acontece no HTML. Estas particularidades permitem que sejam criadas *tags e sub-tags* com conteúdo relativo a ambos os processos. Quando o processo do GUI interpretar o ficheiro XML, este irá identificar a informação através da designação das *tags* e *sub-tags*.

No caso do processo de pré-preenchimento, o ficheiro XML gerado deve conter informação acerca de:

- Todos os *Package* do módulo principal do SCADE Suite;
- Todos os operadores de cada package;
- O prefixo, *Package* e operador utilizado na geração do KCG.

No que diz respeito ao processo de manipulação de dados, o ficheiro XML contém informação acerca de:

- Todos os *inputs* do KATOS, utilizados para executar o processo de manipulação de dados;
- Número de ficheiros editados;
- Número de ficheiros editados sem conteúdo, caso existam (se existirem ficheiros sem conteúdo, estes serão listados);
- Validação da manipulação dos ficheiros afetos ao SILworX, nomeadamente se determinadas seções do código foram alteradas;
- Validação da criação de tabelas para todos os tipos compostos (caso exista um tipo de dado composto não identificado, para o qual não existe uma tabela, este é listado);
- Confirmação da manipulação de dados.

Caso ocorra algum erro durante a execução de qualquer processo, este é imediatamente terminado e é criado o ficheiro XML com apenas uma *tag* que contém informação acerca

desse erro. De seguida, são expostos alguns exemplos dos erros que podem ser apresentados ao utilizador:

- Não foi encontrado nenhum operador no projeto SCADE Suite;
- Pasta que contém os ficheiros KCG não foi encontrada;
- Erros durante a leitura ou escrita dos ficheiros KCG, tabelas ou ficheiro log;
- Ficheiros afetos ao SILworX não identificados ou duplicados (devem existir apenas 3 ficheiros, cada um com formatos diferentes);

É expectável que alguns destes erros nunca ocorram, porque já foram validados no processo do GUI.

5.4. FICHEIRO DE INSTALAÇÃO

Depois de garantir o correto funcionamento e interação entre o GUI e os processos de pré-preenchimento e manipulação de dados, surge a necessidade de gerar um ficheiro de instalação do KATOS. Para este efeito foram analisados vários s*oftware* que permitem desenvolver este tipo de ficheiros, tais como o *Inno Setup*, *Nullsoft Scriptable Install System*, *IExpress* e o *Advanced Installer*.

A necessidade de converter o processo do GUI num ficheiro executável suscitou a necessidade de aprofundar as funcionalidades da API Microsoft *Visual Studio Installer Projects*. Através desta API, concebida pela Microsoft, é possível gerar ficheiros de instalação através do Visual Studio. A utilização desta API para gerar os ficheiros de instalação é significativamente mais vantajosa, visto que não existe a necessidade de instalar novos *software* e gera o ficheiro executável de qualquer programa desenvolvido no *Visual Studio*, como é o caso do GUI. Através desta API é possível definir o diretório de instalação do KATOS, integrar as dependências do GUI, tais como os ficheiros JAR e as suas bibliotecas, entre outras funcionalidades [57], [58].

Os primeiros testes do KATOS, após a instalação, revelam que existe alguma incompatibilidade entre o código testado no ambiente do *Visual Studio* e o ficheiro executável, responsável pelo processo do GUI. Nomeadamente, ao realizar a depuração do processo do GUI, no ambiente do *Visual Studio*, os ficheiros JAR são executados sem qualquer problema, o que não se verificou quando se testou a ferramenta após a instalação. De referir que para executar ficheiros JAR é necessário que o utilizador possua instalado o

Java *Runtime Environment* (JRE). A solução encontrada passou por alterar os parâmetros do comando que permite executar os ficheiros JAR. Inicialmente, os ficheiros eram executados com o comando *'ficheiro.jar "arg1" "arg2"'* e a solução foi alterar o comando para *'java -jar ficheiro.jar "arg1" "arg2"'*. A única diferença entre estes comandos é que no segundo é especificado que o ficheiro.jar deve ser executado através do JRE. Ambos os comandos funcionam corretamente quando executados na *Command Shell* (CMD). No entanto, o primeiro comando possivelmente identifica a extensão do ficheiro e executa-o através do JRE, o interpretador do ficheiro executável pode não ter o mesmo comportamento o que poderá ser a origem desta incompatibilidade.

Em suma, no presente capítulo é detalhado o desenvolvimento do KATOS nomeadamente os *software* utilizados para desenvolver os diferentes processos e a sua integração. De seguida, no capítulo 6, é apresentada a solução desenvolvida.

6. RESULTADOS

Neste capítulo é demonstrado o funcionamento do KATOS, são apresentadas várias comparações entre os dados presentes no projeto SCADE Suite e os dados obtidos pelo KATOS e são apresentadas algumas das funcionalidades apresentadas no capítulo 5. São descritos os testes que permitem validar o KATOS e é reforçada a sua utilidade.

Assim que o utilizador instalar, seguindo os passos apresentados aquando da execução do ficheiro de instalação, e executar o KATOS será apresentada a interface gráfica representada na Figura 31.

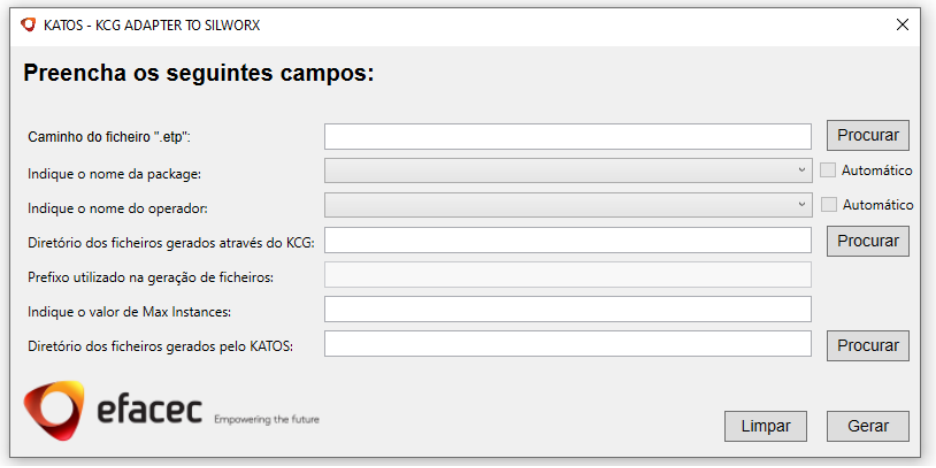


Figura 31 – Interface gráfica do KATOS.

Para todos os *inputs* que seja necessário introduzir um caminho, de um ficheiro ou diretório, é disponibilizado o botão "Procurar", representado na Figura 31, que permite ao utilizador navegar pelos arquivos do seu computador e selecionar pastas ou ficheiros. Existe também a possibilidade de o utilizador introduzir estes *inputs* ao realizar *drag and drop* ou introduzir manualmente o caminho que pretende. Na Figura 32 está representada a função disponibilizada pelo botão "Procurar". De notar que quando o *input* é o ficheiro do projeto SCADE Suite, só são apresentados ficheiros, e quando o *input* é um diretório, o utilizador só pode selecionar diretórios.

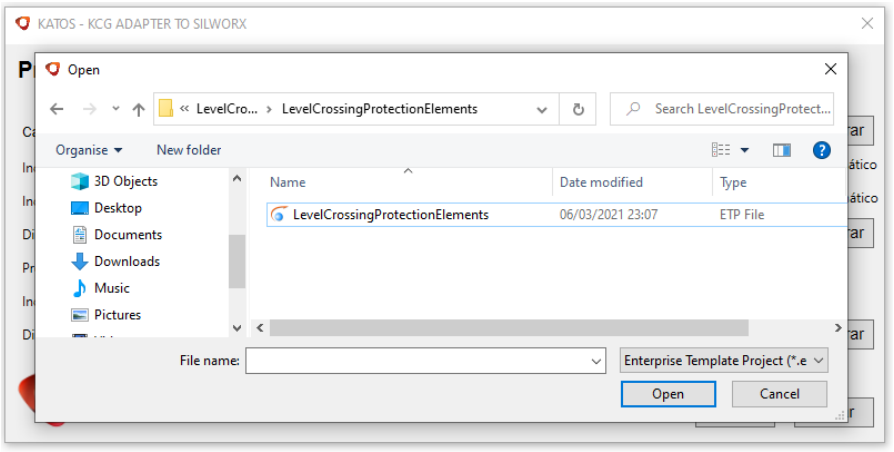


Figura 32 – Exemplo de seleção de ficheiro ou diretório.

O conteúdo do ficheiro XML gerado após o processo de pré-preenchimento encontra-se representado na Figura 33.

```
<?xml version="1.0" ?>
<KcgBuild_Package_Operator>
<Automatico>
    <Prefixo type="boolean">true</Prefixo>
    <Prefixo type="texto">PE</Prefixo>
    <PackOpe type="boolean">true</PackOpe>
    <PackOpe type="PackText">LevelCrossingProtectionElements_0_2_9</PackOpe>
    <PackOpe type="OpeText">LX_ProtectionElements</PackOpe>
</Automatico>

<Package name="LX_HalfBarriers">
    <Operator name="LX_HalfBarriers_Command"></Operator>
    <Operator name="LX_InputManagement"></Operator>
    <Operator name="LX_HalfBarrier_Status"></Operator>
    <Operator name="LX_TransitionDetector_FSM"></Operator>
    <Operator name="LX_InputTransitionDetector"></Operator>
    <Operator name="LX_HalfBarrierBoom_Status"></Operator>
    <Operator name="LX_ObstacleDetector_Status"></Operator>
    <Operator name="LX_HalfBarriers_CorrespondenceStatus"></Operator>
    <Operator name="LX_HalfBarriers_Control"></Operator>
    <Operator name="LX_HalfBarriersIn_RequiredPosition"></Operator>
    <Operator name="LX_HalfBarriers_InhibitMotor"></Operator>
    <Operator name="LX_HalfBarriers_CommandInterface"></Operator>
    <Operator name="LX_HalfBarriersOut_RequiredPosition"></Operator>
```

Figura 33 – Ficheiro XML gerado pelo processo de pré-preenchimento.

Na Figura 34 estão representados os campos preenchidos automaticamente após a interpretação do conteúdo do ficheiro XML. É possível verificar a possibilidade de aceder às configurações utilizadas na última geração do KGC. Visto que a informação acerca da última geração do KCG pode ser alterada, ou o utilizador pode pretender realizar a manipulação de dados sobre outro operador, que não o identificado, o ficheiro XML contém informação sobre os restantes *package*. Na Figura 35, à direita está representado um *package* e os seus operadores (SCADE Suite) e à esquerda está representada a sua listagem (KATOS).

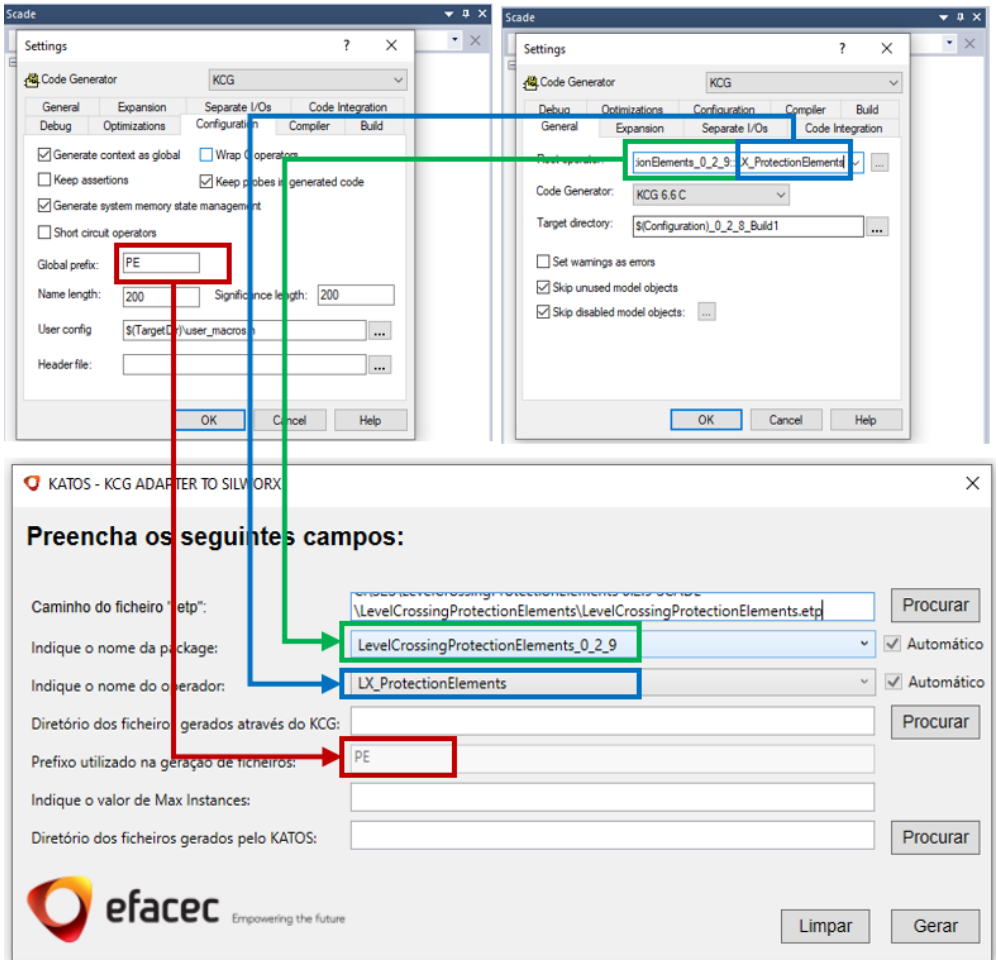


Figura 34 – Demonstração da aquisição dos dados utilizados na geração do KCG.

Figura 35 – Demonstração dos operadores identificados pelo KATOS.

Algumas das verificações e avisos introduzidos em 5.3 estão representados na Figura 36 e Figura 37 e Figura 38.

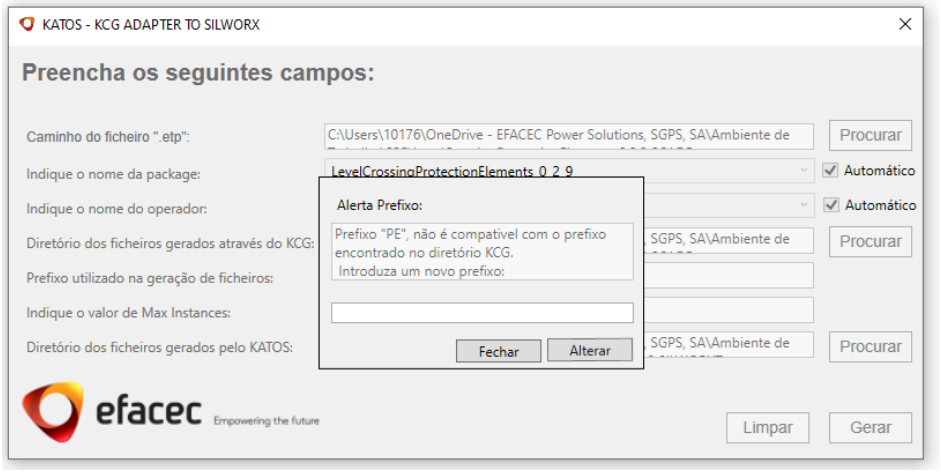


Figura 36 – Demonstração de prefixo incompatível.

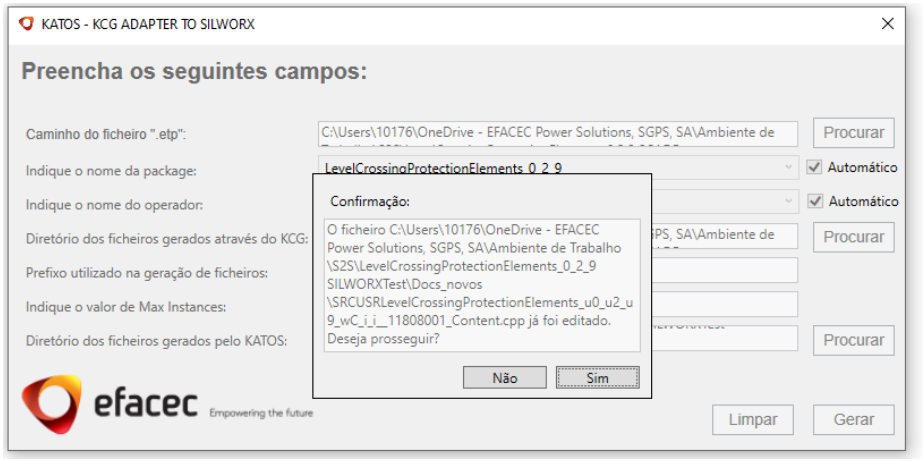


Figura 37 – Demonstração da verificação do conteúdo do Ficheiro do SILworX.

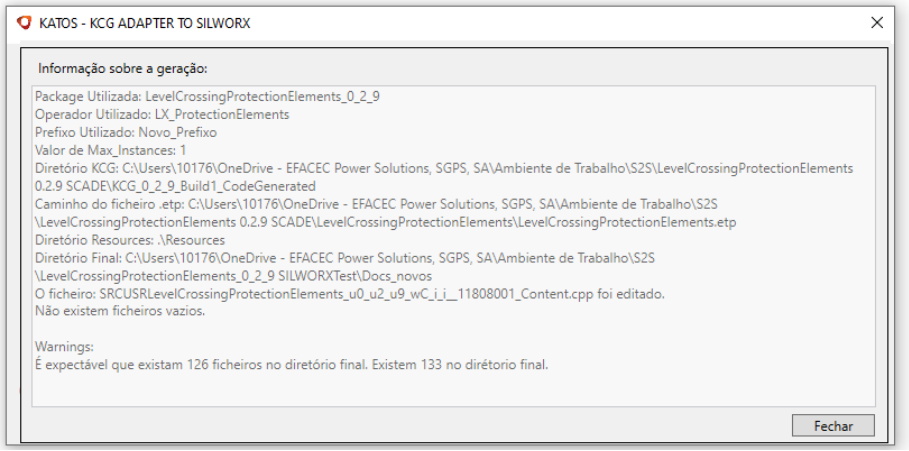


Figura 38 – Demonstração do aviso acerca dos ficheiros expectáveis.

Na Figura 39 está representado o relatório apresentado ao utilizador acerca da geração do KATOS. Através deste relatório é possível identificar que foram manipulados 126 ficheiros KCG inclusive os ficheiros afetos ao SILworX. As tabelas geradas pelo KATOS, Figura 40, não fazem parte dos 126 ficheiros, sendo geradas num diretório à parte.

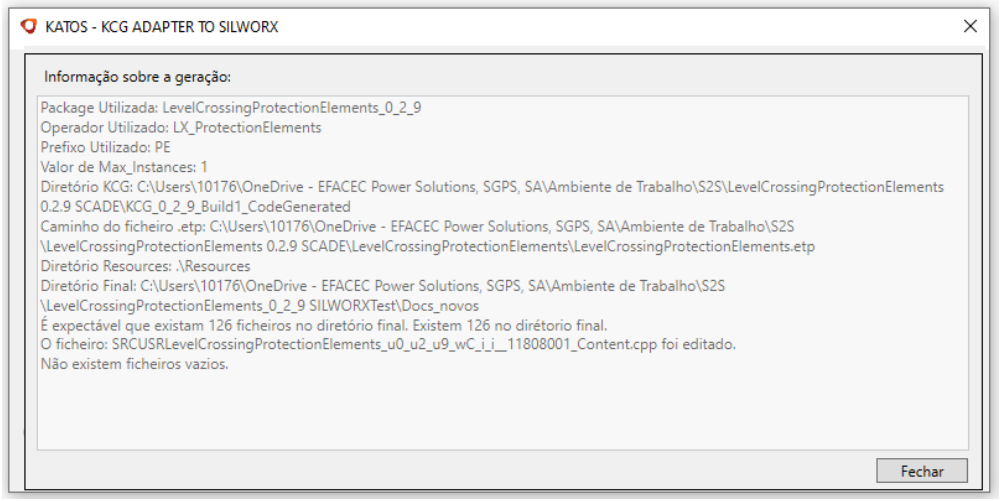


Figura 39 – Relatório da geração do KATOS.

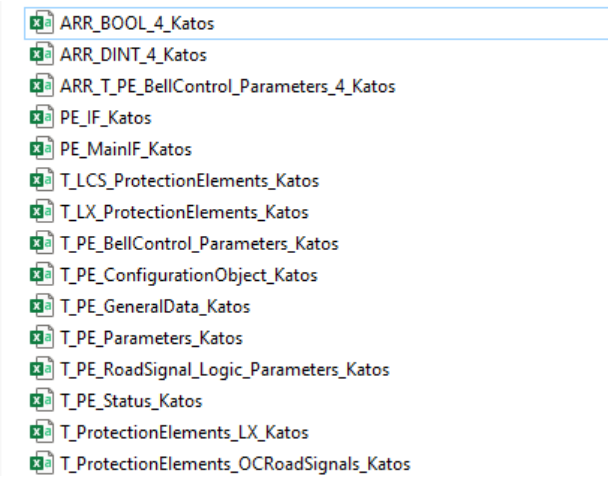


Figura 40 – Exemplo das tabelas criadas pelo KATOS.

No projeto SCADE Suite, utilizado como exemplo para a criação da Figura 39 e Figura 40, o operador selecionado pelo utilizador possui 49 *inputs* e 35 *outputs.* A sua desfragmentação resultou em 160 *inputs* e 112 *outputs*. É assim percetível que a manipulação de 126 ficheiros, a criação de tabelas, o processo de desfragmentação, entre outras tarefas, é um processo moroso e complexo. Um programador que esteja habituado a realizar as tarefas do KATOS manualmente demora cerca de 2 dias de trabalho a completá-las. Esta estimativa foi apresentada pela uTRP e varia consoante a dimensão do projeto em causa.

Tendo como referência modelos do SCADE Suite que já tinham sido integrados nos controladores e que se sabiam estarem corretos, foi realizada uma comparação entre os ficheiros existentes e os ficheiros gerados pelo KATOS, não tendo sido encontrada nenhuma discrepância. Para além disso, foi possível validar o funcionamento do controlador utilizando os ficheiros gerados pelo KATOS. Esta validação é realizada através de uma ferramenta já existente, que compara o comportamento funcional dos modelos e o obtido após o processo de integração, de forma a verificar a sua equivalência.

No computo geral, o KATOS resulta numa melhoria de processo significativa, sendo possível integrar autonomamente os modelos configurados através do SCADE Suite nos controladores, numa questão de segundos.

7. CONCLUSÕES

Nesta tese é apresentado o desenvolvimento de uma ferramenta que permite automatizar a integração de modelos de SSF num controlador. Neste capítulo são apresentadas as considerações finais acerca do trabalho desenvolvido.

Numa fase inicial, foram analisados alguns conceitos que dizem respeito ao setor ferroviário, aos princípios e metodologias de base de um SSF e aos avanços tecnológicos que, indiretamente, originaram a necessidade de desenvolver o projeto apresentado nesta tese. Sendo os SSF sistemas de segurança critica, nesta tese são apresentadas várias normas a ter em conta para implementação prática de um SSF. Do cumprimento dos requisitos impostos nestas normas depende o nível SIL do sistema, procurando-se atingir o mais elevado possível com o intuito de diminuir a probabilidade de ocorrência de situações perigosas. Os sistemas implementados pela EFACEC são certificados com SIL 4, o que permite garantir com grande fiabilidade a prevenção contra colisões e descarrilamentos.

O encravamento eletrónico impulsionou o desenvolvimento de SSF controlados através de PLC ou outras unidades de controlo. Todo o código desenvolvido para estas unidades de controlo é desenvolvido de acordo com a norma EN 50128, implicando que o *software* utilizado para implementar estes sistemas seja devidamente certificado. As disparidades técnicas entre o código gerado pelo KCG e a importação de código no SILworX originam a

necessidade de desenvolver o KATOS. A análise dos requisitos impostos à integração do projeto modelado através do SCADE Suite no SILworX permitiu identificar quais seriam os *inputs* do KATOS e estabelecer especificações para a configuração das gerações autónomas. Estas especificações são intrínsecas ao código desenvolvido e, portanto, não foram apresentadas neste documento.

Automatizar este processo só foi possível devido à utilização da API Java do SCADE desenvolvida para o *software* Eclipse. Através do Eclipse foi possível tirar partido da API Java do SCADE e desenvolver todo o código que respeita à integração autónoma dos modelos do SCADE Suite nos controladores, de acordo com o SILworX. As API disponibilizadas pelo SCADE favorecem significativamente os utilizadores deste *software*, como, por exemplo, os fabricantes de sistemas de segurança critica para o setor ferroviário, automóvel, aeronáutico e outros. Isto porque através da API é possível desenvolver ferramentas integrativas, como é o caso do KATOS, que resultam em melhorias de processo significativas, originando um retorno financeiro considerável, por via da poupança de tempo e recursos.

A utilização do WPF permitiu desenvolver uma interface gráfica intuitiva, responsável por toda a interação do KATOS com o utilizador. Nesta interface são introduzidos e validados os *inputs* do KATOS, são apresentados avisos e é apresentada informação acerca da geração efetuada pelo KATOS, através da interpretação de ficheiros XML que contêm informação acerca dos processos que manipulam o conteúdo do SCADE Suite.

De uma maneira geral, através do projeto apresentado nesta tese, foi desenvolvida uma ferramenta capaz de interpretar o conteúdo de projetos modelados no SCADE Suite e integrar autonomamente estes modelos nos controladores, de acordo com o SILworX. São criadas todas as tabelas necessárias, é manipulado o conteúdo dos ficheiros KCG e dos ficheiros afetos ao SILworX, são apresentados avisos e informação acerca da conversão. O KATOS permite agilizar o processo de integração o que torna o processo mais eficiente comparativamente com a integração manual. Ao longo deste capítulo foi possível identificar que os objetivos propostos nesta tese foram alcançados, com a exceção da integração do KATOS no RAS, visto a ferramenta ainda não estar desenvolvida.

Como trabalho futuro seria interessante implementar as seguintes melhorias:

- Utilização de *threads* em *background* para controlar o estado dos processos dos ficheiros JAR, libertando a *thread* da interface gráfica e impedir o bloqueio durante a execução destes ficheiros;
- Desenvolver a documentação necessária que permita validar a integridade do KATOS e dar início ao processo de certificação, isto permitirá substituir as etapas de validação da integridade dos resultados da ferramenta, como se de um processo manual se tratasse. Visto que o KATOS substitui um processo manual e se enquadra numa ferramenta de classe T3;
- Integrar o KATOS na ferramenta RAS.

Referências Documentais

[1] EFACEC, “Quem Somos | EFACEC.” Disponível em: https://www.efacec.pt/quem-somos/ (Acedido em: Março 29, 2021).

[2] EFACEC, “Estrutura Societária | EFACEC.” Disponível em: https://www.efacec.pt/estrutura-societaria/ (Acedido em: Março 30, 2021).

[3] EFACEC, “Transportes - Ferrovias, Rodovias e Sistemas Industriais | EFACEC.” Disponível em: https://www.efacec.pt/transportes-rodoviarios-ferroviarios/ (Acedido em: Abril 01, 2021).

[4] J. Pereira et al., “Relatório E Contas,” 2019, [Em linha]. Disponível em: https://www.EFACEC.pt/informacao-financeira/.

[5] “AEGIS by EFACEC® - Rail Signaling System: EFACEC Engenharia e Sistemas,.” Disponível em: https://www.virtualmarket.innotrans.com/en/AEGIS-by-EFACEC®-Rail-Signaling-System,p1517796 (Acedido em: Abril 01, 2021).

[6] “TRPNews XSafe.” http://newsletter-trp.EFACEC.com/TRPNews9/index_files/Page315.htm (Acedido em: Abril 01, 2021).

[7] “Safety and functional safety | IEC.” Disponível em: https://www.iec.ch/safety (Acedido em: Abril 13, 2021).

[8] E. Standard, “En 50128,” 2001.

[9] “Liverpool and Manchester Railway | Science and Industry Museum.” Disponível em: https://www.scienceandindustrymuseum.org.uk/objects-and-stories/making-the-liverpool-and-manchester-railway (Acedido em: Abril 14, 2021).

[10] “Statistics | Eurostat.” Disponível em: https://ec.europa.eu/eurostat/databrowser/view/rail_if_line_tr/default/bar?lang=en (Acedido em: Abril 15, 2021).

[11] I. Nacional e R. F. Rodrigues, “Mudança de bitola,” 2001.

[12] “Relatório Especial: Rede ferroviária de alta velocidade na Europa.” Disponível em: https://op.europa.eu/webpub/eca/special-reports/high-speed-rail-19-2018/pt/ (Acedido em: Abril 15, 2021).

[13] J. un Yang, S. Chen, P. ng Qin, e F. Lu, “The Effects of Subway Expansion on Traffic Conditions Evidence from Beijing”, 2015. [Em linha]. Disponível em: https://www.jstor.org/stable/resrep15032?seq=1#metadata_info_tab_contents. (Acedido em: Abril 15, 2021).

[14] S. Li, Y. Liu, A. O. Purevjav, e L. Yang, “Does subway expansion improve air quality?,” Journal of Environmental Economics and Management, vol. 96, pp. 213–235, 2019, doi: 10.1016/j.jeem.2019.05.005.

[15] UNIFE, “GLOBAL RAIL MARKET GROWS DESPITE COVID-19.” Disponível em: https://www.unife.org/news/209-unife-world-rail-market-study-launch.html (Acedido em: Abril 16, 2021).

[16] Eurostat, “Railway accidents 2019 - Statistics Explained.” Disponível em: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=File:Railway_accidents_2019.png (Acedido em: Abril 15, 2021).

[17] G. Theeg e S. Vlasenko, Railway Signalling and Interlocking. 2019.

[18] S. A. Khan, “Railway Interlocking Design Support Tools AutoCAD Plugin based approach,” University of Oslo Department of Informatics, 2016.

[19] L. WILLGRESS, “Victorian signal boxes will be retired as new technology is introduced nationwide.” Disponível em: https://www.dailymail.co.uk/news/article-3456101/Farewell-signal-box-computer-technology-takes-control-Britain-s-150-year-old-rail-network.html (Acedido em: Abril 19, 2021).

[20] “EFACEC wins one of the most important level crossing competitions in Europe | EFACEC.” Disponível em: https://www.EFACEC.pt/en/EFACEC-wins-one-of-the-most-important-level-crossing-competitions-in-europe/ (Acedido em: Abril 20, 2021).

[21] Jörn Pachl, “Railway Signalling Principles,” p. 83, 2020, [Em linha]. Disponível em: https://www.researchgate.net/publication/341277248_Railway_Signalling_Principles.

[22] P. Kawalec e R. ˙ Zysko, “Modern methods in railway interlocking algorithms design,” Microprocessors and Microsystems, vol. 44, pp. 38–46, 2015, doi: 10.1016/j.micpro.2015.11.018.

[23] “Understanding Safety Integrity Level IEC 61511 | Instrumentation Tools.” Disponível em: https://instrumentationtools.com/understanding-safety-integrity-level-iec-61511/ (Acedido em: Abril 30, 2021).

[24] P. S. Services, "Products Solutions Services Functional Safety in practice". Disponível em: https://portal.endress.com/dla/5001068/0812/000/00/WP01032F00EN0116.pdf.(Acedido: Abil 30, 2021.)

[25] J. Masters e J. Britton, “Paper How To Achieve En 50128 Compliance”, PRQA - Programming Research, 2011.

[26] T. British Standards Institution, BSI Standards Publication Railway Applications-The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS). 2017.

[27] A. Nash, D. Huerlimann, J. Schuette, and V. P. Krauss, “RailML-A standard data interface for railroad applications.” [Em linha]. Disponível em: www.railml.org. (Acedido em: Maio 12, 2021.)

[28] “Railway applications-Communication, signalling and processing systems-Safety related electronic systems for signalling,” 2018. [Em linha]. Disponível em:https://infostore.saiglobal.com/preview/366980870223.pdf?sku=859695_SAIG_NSAI_NSAI_2695621. (Acedido em: Maio 14, 2021.)

[29] I. Standard, “EN 50159 Railway applications-Communication, signalling and processing systems-Safety-related communication in transmission systems,” 2010. [Em linha]. Disponível em: https://infostore.saiglobal.com/preview/98700340660.pdf?sku=874837_SAIG_NSAI_NSAI_2079785. (Acedido em: Maio 14, 2021)

[30] “Training EN 50126, EN 50129, EN 50128 for the rail sector| TÜV SÜD.” Disponível em: https://www.tuvsud.com/en/industries/infrastructure-and-rail/rail/functional-safety-training/en5012x-training (Acedido em: Abril 28, 2021).

[31] J. Boulanger, CONTROL, SYSTEMS AND INDUSTRIAL ENGINEERING SERIES CENELEC 50128 and IEC 62279 Standards. 2015.

[32] U. Yildirim, M. S. Durmuş, e M. T. Söylemez, “Automatic interlocking table generation for railway stations using symbolic Algebra,” IFAC Proceedings Volumes (IFAC-PapersOnline), vol. 45, no. 24, pp. 171–176, 2012, doi: 10.3182/20120912-3-BG-2031.00035.

[33] “RailCOMPLETE.” Disponível em: https://www.railcomplete.com/en/ (Acedido em: Maio 01, 2021).

[34] “Introduction - railML.org (EN).” Disponível em: https://www.railml.org/en/introduction.html (Acedido em: Maio 12, 2021).

[35] “Ansys SCADE Suite | Model-Based Design.” Disponível em: https://www.ansys.com/products/embedded-software/ansys-scade-suite (Acedido em: Maio 03, 2021).

[36] J. L. Colaço, B. Pagano, e M. Pouzet, “SCADE 6: A formal language for embedded critical software development,” Proceedings - 11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, vol. 2018-Janua, pp. 1–11, 2018, doi: 10.1109/TASE.2017.8285623.

[37] “Report to the Certificate Code Generator SCADE Suite KCG 6.6.” Acedido em: Maio 05, 2021. [Em linha]. Disponível em: https://www.cadfem-cis.ru/media/tuv_certificate_kcg_6.6_report.pdf

[38] “SCADE Suite - Software Testing Tools Guide.” Disponível em: http://www.testingtoolsguide.net/tools/scade-suite/ (Acedido em: Maio 03, 2021).

[39] ANSYS, “SCADE Metamodels.”

[40] HIMA, “SafeRail: Lean safety technology for the rail sector.” Disponível em: https://www.hima.com/en/about-hima/news/article/saferail-lean-safety-technology-for-the-rail-sector (Acedido em: Maio 06, 2021).

[41] T. SÜD, “HIMatrix CENELEC SIL 4.” Disponível em: https://sds-automatyka.pl/wp-content/uploads/Cert_TUEV_HIMatrix_CENELEC_Z10_12_05_19183_047_D_D.pdf (Acedido em: Maio 06, 2021).

[42] T. SÜD, “HIMaX CENELEC SIL 4.” Disponível em: http://www.netmuh.com.tr/media/files/cert_tuev_himax_cenelec_z10_11_02_19183_007_d_d92397.pdf (Acedido em: Maio 06, 2021).

[43] Hima, “HIMatrix Safety-Related Controller F2 DO 8 01 Manual,” 2010. [Em linha]. Disponível em: http://www.netmuh.com.tr/media/products/f2-do-8-01-manual_katalog63994.pdf (Acedido em: Maio 06, 2021.)

[44] “SILworX.” Disponível em: https://www.hima.com/en/products-services/silworx (Acedido em: Maio 19, 2021).

[45] “SILworX Certificate.” Disponível em: https://fs-products.tuvasi.com/files/certificates/certificates_asi/2015/EZ/968_EZ_487_04_15/968_EZ_487_04_15_en_el.pdf (Acedido em: Maio 19, 2021).

[46] “Detailed Information HIMatrix.” Disponível em: https://www.hima.com/en/products-services/himatrix/detailed-information-himatrix (Acedido em: Maio 20, 2021).

[47] S. Controller e E. Hi, "HIMatrix- F35 03 Manual", [Em linha]. Disponível em: https://sds-automatyka.pl/wp-content/uploads/HI_800_477_E_HIMatrix_F35_03.pdf.(Acedido em: Maio 22, 2021).

[48] Hima, "SILworX Communication Manual", [Em linha]. Disponível em: https://sds-automatyka.pl/wpcontent/uploads/HI_801_101_E_SILworX_Communication_Manual.pdf. (Acedido em: Maio 22, 2021).

[49] Esterel Technologies SAS, “SCADE Java/Tcl API Guide”, Esterel Technologies, 2018.

[50] “Java Swing Tutorial - javatpoint.” Disponível em: https://www.javatpoint.com/java-swing (Acedido em: Maio 28, 2021).

[51] R. E. Dave Wood, Marc Loy, “Java Swing”, O’Reilly , 1998.

[52] “What is Windows Presentation Foundation - WPF .NET | Microsoft Docs.” Disponível em: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-5.0 (Acedido em: Maio 28, 2021).

[53] “Windows in WPF overview - WPF .NET | Microsoft Docs.” Disponível em: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/windows/?view=netdesktop-5.0 (Acedido em: Maio 28, 2021).

[54] “Swing Tutorial.” Disponível em: http://web.cs.ucla.edu/~miryung/teaching/EE379K-Spring2014/SwingTutorial.html (Acedido em: Maio 24, 2021).

[55] “eclipse - What is the difference between runnable jar library handling options? - Stack Overflow.” Disponível em: https://stackoverflow.com/questions/8302894/what-is-the-difference-between-runnable-jar-library-handling-options (Acedido em: Maio 25, 2021).

[56] “Export Eclipse Java Project as Runnable JAR | AlertSite Documentation.” Disponível em: https://support.smartbear.com/alertsite/docs/monitors/web/selenium/export-eclipse-java-project-as-runnable-jar.html (Acedido em: Maio 25, 2021).

[57] “Deploy a Visual C++ Application By Using a Setup Project | Microsoft Docs.” Disponível em: https://docs.microsoft.com/en-us/cpp/windows/walkthrough-deploying-a-visual-cpp-application-by-using-a-setup-project?view=msvc-160 (Acedido em: Maio 26, 2021).

[58] “Visual Studio Installer Deployment | Microsoft Docs.” Disponível em: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/2kt85ked(v%3Dvs.100) (Acedido em: Maio 26, 2021).